MARCELO EDUARDO HOEPFNER
HELTON ALEXANDER MICHEL

PROJETO – SCOS
(SISTEMA CONTROLE DE ORDENS DE SERVIÇO)

CURITIBA
2006

MARCELO EDUARDO HOEPFNER

HELTON ALEXANDER MICHEL

ESCOLA TÉCNICA DA UFPR
BIBLIOTECA

PROJETO – SCOS

(SISTEMA CONTROLE DE ORDENS DE SERVIÇO)

Projeto Final apresentado como requisito parcial à obtenção do grau de Tecnólogo em Informática, Curso de Tecnologia em Informática, Escola Técnica da Universidade Federal do Paraná.

Orientadores:
Prof.º Mario de Paula Soares Filho
Prof.º Irapuru Haruo Florido

MG
005.1
H695

CURITIBA

2006

MARCELO EDUARDO HOEPFNER

HELTON ALEXANDER MICHEL

PROJETO – SCOS
(SISTEMA CONTROLE DE ORDENS DE SERVIÇO)

Professor Mario de Paula Soares

Orientador

Professor Irapuru Haruo Florido

Orientador

Membro da Banca Examinadora

Membro da Banca Examinadora

Aos nossos familiares,

com todo nosso amor.

AGRADECIMENTOS

Ao prof. Mario, pela compreensão e paciência e pela sua orientação.

Ao prof. Irapuru, por sua colaboração e orientação.

Ao prof. Dieval pela imensa colaboração em nossos diagramas e nas correções efetuadas em nosso banco de dados, e pela sua orientação.

Aos amigos e profissionais exemplares Emilio Córdova, Antonio Adir Vaz Junior, pelas brilhantes idéias e pelas contribuições com seus conhecimentos e experiências profissionais, que não esqueceremos pelo resto de nossas vidas.

A Nossa família que soube nos entender quando não podíamos dar-lhes atenção, pois estávamos muito atarefados com nosso projeto, e nem mesmo assim não esqueceram que estávamos longe mais ao mesmo tempo perto.

"O único lugar onde o sucesso vem
antes do trabalho é no dicionário."

Albert Einstein

RESUMO

O projeto descrito neste trabalho propõe demonstrar os passos para o desenvolvimento de um software que assista os profissionais da empresa Oliprinter Informática Ltda®. no cumprimento do seu ofício. No processo de elaboração do sistema, foram utilizados os conceitos de análise orientada a objetos, baseado nos padrões propostos pela UML (Unified Modeling Language – Linguagem de Modelagem Unificada). A implementação é realizada com a ferramenta de desenvolvimento Borland Delphi 7, software que utiliza a linguagem pascal orientada a objetos. O armazenamento dos dados é feito em um banco de dados relacional PostgreSQL 8.0 (Ansi-92), que se comunica de forma integral com o modelo orientado à objetos. Destaca-se como característica principal a possibilidade de reutilização do conteúdo deste trabalho.

Para tanto, aproveitamos a real necessidade de uma loja de informática, de automatizar o seu controle interno de ordens de serviço.

LISTA DE SIGLAS

ANSI – American National Standard Institute

DER – Diagrama Entidade Relacionamento.

OO – Orientado a Objeto.

OS – Ordem de Serviço.

UML – Unified Modeling Language

SCOS – Sistema de Controle de Ordem de Serviço

LISTA DE SÍMBOLOS

® - Marca registrada.

SUMÁRIO

1 Introdução ____ *15*

1.1 Cenário Atual ____ **16**

1.2 Definiação do problema ____ **18**

1.3 Soluação desenvolvida ____ **19**

2 Metodologia ____ *20*

2.1 UML (Unified Modeling Language) ____ **21**

3 Ferramentas utilizadas ____ *22*

3.1 DBDesigner 4 ____ **22**

3.2 Delphi 7 ____ **22**

3.3 Postgre SQL 8.0 ____ **23**

3.4 Poseidon for uml 3.2.1 Community Edition ____ **23**

3.5 PostgreSQL Manager 2 ____ **24**

3.6 Microsoft Project 2003 ____ **24**

3.7 SmartDraw 7 ____ **24**

4 Dicionário de dados ____ *25*

5 Camada de persistência ____ *35*

6 Cronograma ____ *52*

7 Diagramas de Casos de Uso ____ *53*

7.1.1 Caso de uso – Administrador: ____ 53

7.1.2 Caso de uso – Recepcionista: ____ 58

7.1.3 Caso de uso – Técnico: ____ 69

7.1.4 Caso de uso – Almoxarife: ____ 73

7.1.5 Caso de uso – Gerente: ____ 78

8 Diagrama de Classes ____ *84*

9 Diagramas de Sequência ____ *85*

10 Diagrama de Colaboração ____ *116*

11 Diagrama de Entidade Relacionamento ____ *147*

12 Diagrama de Estados ____ *148*

13 Diagramas de Telas ____ *149*

14 Telas do Sistema ____ *150*

15 Script de Criação do Banco de Dados ____ *178*

16 Instalação do Sistema ____ *185*

17 Implementações Futuras ____ *186*

18 Ambiente de Produção ____ *187*

18.1 Servidor ____ **187**

18.2 Cliente ____ **187**

19 Conclusão ____ *188*

20 Referências Bibliográficas ____ *189*

1 Anexos ____ *190*

2 Código Fonte ____ *190*

2.1 Classes ____ **190**

2.1.1 unit TUsuario; ____ 190

2.1.2 unit TAcesso; ____ 195

2.1.3 unit TCaixa; ____ 197

2.1.4 unit TCidade; ____ 202

2.1.5 unit TContatos; ____ 205

2.1.6 unit TContrato; ____ 208

2.1.7 unit TEquipamento; 212

2.1.8 unit TFones; 215

2.1.9 unit TGrupo; 219

2.1.10 unit TItensOrdemDeServico; 222

2.1.11 unit TLogOS; 226

2.1.12 unit TMenu; 228

2.1.13 unit TOrdemDeServico; 232

2.1.14 unit TPeca; 238

2.1.15 unit TPessoa; 241

2.1.16 unit TSubItensOrdemDeServico; 245

2.1.17 unit TTipoDeEquipamento; 250

2.1.18 unit TTipoDePeca; 253

2.1.19 unit TTipoDocumento; 256

2.1.20 unit TTipoMov; 259

2.1.21 unit TUf; 262

2.2 Fontes do Sistema 265

2.2.1 unit uV_OS; 265

2.2.2 unit u_L_V_OS; 270

2.2.3 unit uApres; 272

2.2.4 unit uCAltera_Orcamento; 273

2.2.5 unit uCCidades; 284

2.2.6 unit uCClientes; 286

2.2.7 unit uCContratos; 297

2.2.8 unit uCEquipamentos; 300

2.2.9 unit uCEstados; 302

2.2.10 unit uCFornecedores; 304

2.2.11 unit uCGrupoDeUsuarios; 315

2.2.12 unit uCOrdemServico; 317

2.2.13 unit uCOS_AprovOrcamento; 328

2.2.14 unit uCOS_Atualizar; 331

2.2.15 unit uCOS_Fechar; 338

2.2.16 unit uCPecas; 344

2.2.17 unit uCTDocumentos; 346

2.2.18 unit uCTEquipamentos; 348

2.2.19 unit uCTPecas; 350

2.2.20 unit uCUsuarios; 352

2.2.21 unit uDm_Base; 355

2.2.22 unit uLCidades; 356

2.2.23 unit uLClientes; 358

2.2.24 unit uLContratos; 360

2.2.25 unit uLCPagar; 362

2.2.26 unit uLCReceber; 366

2.2.27 unit uLEquipamentos; 370

2.2.28 unit uLEstados; 373

2.2.29 unit uLFornecedores; 375

2.2.30 unit uLGrupoDeUsuarios; 377

2.2.31 unit uListaCaixa; 379

2.2.32 unit uLLiberaOrcar; 382

2.2.33 unit uLogin; 385

2.2.34 unit uLOrcamentoRecusado; 388

2.2.35 unit uLOrcar; 390

2.2.36 unit uLOrcarAltera; 392

2.2.37 unit uLOS_Atualiza; 399

2.2.38 unit uLOS_Cancelar; 401

2.2.39 unit uLOS_Fechar; 405

2.2.40 unit uLPecas; 407

2.2.41 unit uLTDocumentos; 410

2.2.42 unit uLTEquipamentos; 412

2.2.43 unit uLTPecas; 414

2.2.44 unit uLUsuarios; 416

2.2.45 unit uMain; 419

2.2.46 unit uMovCaixa; 430

2.2.47 unit uMovCaixaCReceber; 434

2.2.48 unit uPesqCliente; 438

2.2.49 unit uPesqContratos; 441

2.2.50 unit uPesqEquipamento; 443

2.2.51 unit uPesqFornecedor; 445

2.2.52 unit uPesqPecas; 447

2.2.53 unit uRelatCaixa; 449

2.2.54 unit uRelatEstoque; 451

2.2.55 unit uRelatOS; 452

2.2.56 unit uTAcesso; 455

2.2.57 unit uTela_ListaPadrao; 459

1 INTRODUÇÃO

A Oliprinter Informática, empresa sediada em Curitiba-Pr é uma empresa credenciada pela OLIVETTI DO BRASIL S/A, no segmento de Assistência Técnica, Venda de Acessórios e Venda de Equipamentos.

Conta para isso, com profissionais atuantes no mercado a mais de 20 anos, com treinamento técnico de fabricantes tradicionais no mercado, tais como: Olivetti, HP. Bematech, Lexmark, Epson, Canon.

A Oliprinter Informática, atua com a Assistência Técnica de Microcomputadores, Impressoras (Matriciais, Jato de Tinta, Multifuncionais, Laser, Etc...), Scanners, Monitores, Fax, Copiadoras Analógicas e Digitais, Calculadoras e Máquinas de Escrever. Atendimentos ao cliente no balcão ou On-Site.

Trabalha com a venda de materiais de consumo, acessórios e periféricos para os produtos atendidos. Executa serviços de cabeamento extruturado, instalação e configuração de switches e roteadores.

Busca ampliar o mercado de atuação oferecendo aos clientes a especialização técnica em produtos e serviços.

1.1 CENÁRIO ATUAL

Atualmente, a empresa Oliprinter Informática realiza o cadastro de suas ordens de serviço através de uma planilha numerada e controlada, que esta no formato .DOC e é aberta no Microsoft Word. Onde nesta planilha, são digitados alguns dados como: a data de entrada, os dados básicos do cliente, uma descrição simples do equipamento trazido para manutenção e o sintoma (indício do mau funcionamento) apresentado por este. Neste mesmo documento podem ser registradas também, informações que o cliente ou o funcionário que está iniciando a ordem de serviço, julguem relevantes, como por exemplo, se uma impressora foi trazida pelo cliente com ou sem os cartuchos de tinta ou se o cliente trouxe algum periférico ou cabo junto ao gabinete. Após o cadastro, o equipamento recebe uma etiqueta com o número da ordem de serviço aberta.

Em seguida o equipamento é posto eu uma estante de "entrada" (novas ordens de serviço), a pessoa que tem a responsabilidade para que os equipamentos sejam avaliados é o gerente da empresa, sendo necessários diversos aciones aos técnicos durante o dia, para que o equipamento seja avaliado.

Após a avaliação e a definição do problema apresentado pelo equipamento (diagnóstico), o técnico comunica ao gerente, o valor do orçamento é estipulado e repassado ao cliente por telefone. O cliente pode aprovar ou não o orçamento e, nem sempre, essa definição é feita no momento em que lhe é informado o orçamento para a realização do serviço e conseqüente solução do problema.

Caso o orçamento seja aprovado, o técnico é verbalmente comunicado para que o serviço seja executado. Caso o orçamento seja reprovado, o técnico também é comunicado para que equipamento seja preparado para ser entregue ao cliente, sem que nenhuma alteração seja realizada.

No momento em que o equipamento fica pronto para ser entregue, independente da execução ou não do serviço, o técnico comunica verbalmente ao gerente a situação do equipamento. O gerente faz o contato telefônico com o cliente para avisá-lo sobre disponibilidade do equipamento para retirada.

1.2 DEFINIÇÃO DO PROBLEMA

A dificuldade de controlar, procurar e armazenar as ordens de serviços (informações) impossibilita um controle a nível gerencial, tais como: maior rapidez, maior produtividade, maior lucratividade e maior satisfação do cliente.

1.3 SOLUÇÃO DESENVOLVIDA

O sistema SCOS – Sistema de Controle de Ordens de Serviço foi desenvolvido para centralizar, automatizar, controlar e auditar todo o fluxo de informações geradas por uma ordem de serviço, agilizando os processos envolvidos neste fluxo e disponibilizando essas informações para o operacional e gerencia.

2 METODOLOGIA

Para o desenvolvimento do sistema SCOS, foi necessário identificar as possibilidades e recursos para sua implantação. No desenvolvimento do trabalho acadêmico utilizamos o sistema operacional Windows XP® pois os equipamentos no cliente são providos por este sistema operacional, a plataforma de desenvolvimento utilizada foi o Borland Delphi® 7 conforme solicitação do cliente, e o SGBD que pode estar localizado em um equipamento linux ou windows foi o Postgre SQL 8. Foi utilizado o paradigma de desenvolvimento orientado a objetos (OO), pois sua maior vantagem é a manutenção do código e a implementação de novos módulos no sistema, que inevitavelmente são necessários após a implantação e utilização.

BIBLIOTECA DA ESCOLA TÉCNICA DA UFPR

2.1 UML (UNIFIED MODELING LANGUAGE)

A UML é uma linguagem padronizada que utiliza recursos gráficos para representar situações reais, sob diferentes perspectivas, no processo de desenvolvimento de um sistema orientado a objeto. Cada uma das perspectivas pode ser apresentada através de diagramas que utilizam recursos oferecidos pela UML.

Basicamente, são representados os itens do sistema e como se relacionam, na seqüência que cada situação ou perspectiva exige.

Algumas das vantagens de um sistema modelado sob as regras e notações da UML são as seguintes:

- Facilidade de visualização de um processo ou fluxo do sistema, por serem representados graficamente.
- Padronização dos modelos, graças às regras pré-estipuladas da linguagem.
- Pode ser facilmente compreendido por qualquer desenvolvedor. Isto é possível por ser uma linguagem que gera uma documentação detalhada e padronizada.

3 FERRAMENTAS UTILIZADAS

3.1 DBDESIGNER 4

O dbdesigner® 4 é uma ferramenta para modelagem de banco de dados, open source, que gera o esquema das estruturas da modelagem. É uma ferramenta livre que possibilita a visualização gráfica do projeto do banco de dados (der), a modelagem e o controle dos dados nele existente. Pode ser utilizado tanto em ambiente windows quanto em linux. Foi escolhido por ter sido muito citado como uma ferramenta eficiente durante as pesquisas iniciais para o projeto.

3.2 DELPHI 7

O delphi® é um ambiente de desenvolvimento de aplicações, orientado a objeto, que permite o desenvolvimento de aplicações para os sistemas operacionais windows, com pouca codificação.

O delphi tem ferramentas de desenvolvimento, como templates e experts de aplicações e formulários, que aumentam muito a produtividade, facilitando a programação da aplicação.

3.3 POSTGRE SQL 8.0

O postgresql® é um SGBD (sistema gerenciador de banco de dados) objeto-relacional de código aberto, com mais de 15 anos de desenvolvimento. É extremamente robusto e confiável, além de ser extremamente flexível e rico em recursos. Ele é considerado objeto-relacional por implementar, além das características de um sgbd relacional, algumas características de orientação a objetos, como herança e tipos personalizados. A equipe de desenvolvimento do postgresql sempre teve uma grande preocupação em manter a compatibilidade com os padrões sql92/sql99.

3.4 POSEIDON FOR UML 3.2.1 COMMUNITY EDITION

A ferramenta Poseidon® é uma ferramenta para modelagem uml:

Com ele é possível modelar:

– Diagramas de classe;

– Diagramas de estado;

– Diagramas de atividade;

– Diagramas de Caso de Uso;

– Diagramas de Colaboração;

– Diagramas de Sequência, com algumas limitações.

• É uma extensão do ArgoUML;

• Efetua a geração de código das classes para delphi;

3.5 POSTGRESQL MANAGER 2

É uma ferramenta de desempenho elevado para a administração e o desenvolvimento da base de dados de PostgreSQL®. Trabalha com quaisquer versões de PostgreSQL até 8.0 e suporta todas as características mais atrasadas de PostgreSQL including Tablespaces, nomes do argumento nas funções, e assim por diante. Oferece a abundância de ferramentas poderosas para usuários experientes tais como o desenhador visual da base de dados, o construtor visual da query, e o editor poderoso do BLOB satisfazem a todas suas necessidades.

3.6 MICROSOFT PROJECT 2003

Programa de gerenciamento de projetos principal da Microsoft. Com ferramentas familiares e fáceis de serem usadas, permitindo que você gerencie projetos independentemente de sua área de trabalho.Essa solução, permite que as organizações alinhem iniciativas comerciais, projetos e recursos para melhores resultados comerciais. Com o uso de recursos de relatório e análise flexíveis, as organizações têm as informações acionáveis que ajudarão a otimizar recursos, priorizar tarefas e alinhar os projetos como um portfólio aos objetivos gerais dos negócios.

3.7 SMARTDRAW 7

O Smart Draw é uma ferramenta poderosa para fazer diagramas. Ele traz um banco com elementos gráficos e importa dados de programas como Visio, AutoCAD e 3D Studio. O programa ainda traz assistentes que ajudam a fazer rapidamente diagramas complicados. Está disponível em várias edições, indicadas para diferentes aplicações

4 DICIONÁRIO DE DADOS

Tabela acesso		
Campo	Tipo	Descrição
id_usuario	INTEGER	id do usuario
id_menu	VARCHAR(50)	id do menu
nivel	INTEGER	nivel de acesso

Tabela caixa		
Campo	Tipo	Descrição
id_lcto	SERIAL	id do lancamento
id_pessoa	INTEGER	id da pessoa
id_tipomov	INTEGER	id do tipo do movimento
num_doc	INTEGER	numero da os ou da NF ou do RECIBO
num_parcela	INTEGER	numero da parcela
data_emissao	DATE	data da emissao
data_vcto	DATE	data do vencimento
data_prorrog	DATE	data da prorogação
data_pgto	DATE	data pagamento
tipo_pgto	INTEGER	1= dinheiro 2= cheque 3= cartao
valor_apagar	INTEGER	valor a pagar
valor_juro	INTEGER	valor dos juros
valor_pago	INTEGER	valor pago

situacao	INTEGER	cadastrada ou baixada
obs	VARCHAR(100)	observações

Tabela cidade

Campo	Tipo	Descrição
id_cidade	INTEGER	id da cidade
id_uf	INTEGER	id da uf
descricao	VARCHAR(50)	nome da cidade

Tabela contatos

Campo	Tipo	Descrição
id_pessoa	INTEGER	id da pessoa
nome	VARCHAR(30))	nome do contato
dia_nasc	VARCHAR(2)	dia de nascimento
mes_nasc	VARCHAR(2)	mes de nascimento
cargo	VARCHAR(30)	cargo
setor	VARCHAR(30)	setor
email	VARCHAR(100)	email

Tabela contrato		
Campo	Tipo	Descrição
id_contrato	SERIAL	id do contrato
id_pessoa	INTEGER	id da pessoa
data_inicio_contrato	DATE	data de cadastro do contrato
data_fim_vigencia	DATE	data de inicio da vigencia do contrato
data_inicio_vigencia	DATE	data do fim da vigencia do contrato
data_bloqueio	DATE	data em que foi bloqueado o contrato
valor_mensal	NUMERIC	valor mensal cobrado pelo contrato
dia_vcto	INTEGER	dia do vencimento do contrato
dias_prorog	INTEGER	dias de prorogação apos o vencimento do contrato
bloqueado	INTEGER	se o contrato encontrase bloqueado

Tabela equipamento		
Campo	Tipo	Descrição
id_equipamento	INTEGER	id do equipamento
id_tipo_equipamento	INTEGER	id do tipo de equipamento
id_contrato	INTEGER	id do contrato
descricao	VARCHAR(50)	descrição do equipamento
numero_serie	VARCHAR(50)	numero de serie do equipamento

Tabela grupo		
Campo	Tipo	Descrição
id_grupo	INTEGER	id do grupo de usuarios
nomegrupo	VARCHAR(30)	nome do grupo de usuarios

Tabela item_os		
Campo	Tipo	Descrição
id_os	INTEGER	id da ordem de serviço
id_item_os	INTEGER	id do item da ordem de serviço
id_equipamento	INTEGER	id do equipamento
id_contrato	INTEGER	id_contrato
tipo_atendimento	INTEGER	tipo do atendimento 0= avulso 1= contrato 2= garantia
desc_equip	TEXT	descrição do equipamento
desc_prob	TEXT	descrição do problema relatado pelo cliente

Tabela item_os_mov		
Campo	Tipo	Descrição
id_sitem	INTEGER	id do subitem
id_item_os	INTEGER	id do item da ordem de serviço
id_os	INTEGER	id da ordem de serviço
id_contrato	INTEGER	id do contrato
id_peca	INTEGER	id da peça
qtde_peca	INTEGER	quantidade de peças
descricao	TEXT	descrição
cobrar	INTEGER	se vai ser cobrado do cliente ou não

Tabela logs_os		
Campo	Tipo	Descrição
id_log	SERIAL	id do log
id_os	INTEGER	id da ordem de serviço
id_usuario	INTEGER	id do usuario
acao	VARCHAR(30)	acao efetuada na os
data	DATE	data que foi efetuada a acao

Tabela menu		
Campo	Tipo	Descrição
id_menu	VARCHAR(50)	id do menu
tipo_menu	INTEGER	se é menu ou submenu
descricao	VARCHAR(30)	descrição
id_modulo	INTEGER	id do modulo
sequencia	INTEGER	sequencia
menupai	INTEGER	menu pai se sub menu

Tabela os		
Campo	Tipo	Descrição
id_os	SERIAL	numero da ordem de serviço
id_pessoa	INTEGER	id da pessoa
data_cad	DATE	data de cadastro da ordem de serviço
data_alt	DATE	data que sofreu a ultima alteração
data_fechou	DATE	data que fechou a ordem de serviço
data_canc	DATE	data que cancelou a ordem de serviço
status	TEXT	status da ordem de serviço
valor_pecas	NUMERIC	valor total das peças da ordem de serviço
valor_orcado	NUMERIC	valor total do orçamento

valor_cobrado	NUMERIC	valor cobrado do cliente
valor_mao_obra	NUMERIC	valor da mão de obra
valor_pecas_ncobrado	NUMERIC	valor de peças não cobrado
valor_desconto_mo	NUMERIC	valor do desconto da mão de obra
valor_desconto_pecas	NUMERIC	valor do desconto das peças

Tabela peca

Campo	Tipo	Descrição
id_peca	INTEGER	id da peça
id_tipo_peca	INTEGER	id do tipo de peça
descricao	VARCHAR(50)	descrição da peça
quantidade	INTEGER	quantidade em estoque
valor_unitario	FLOAT	valor unitario da peça

Tabela Pessoa

Campo	Tipo	Descrição
id_pessoa	SERIAL	id da pessoa
id_uf	INTEGER	id da uf
id_cidade	INTEGER	id da cidade
nome	VARCHAR(40)	razao social ou nome
endereco	VARCHAR(40)	endereço
bairro	VARCHAR(30)	bairro
cep	VARCHAR(9)	cep
tipo	VARCHAR(1)	tipo fisica / juridica

tipo_pessoa	VARCHAR(1)	C = Cliente e F= fornecedor
cgc_cnpj	VARCHAR(18)	documento cgc ou cnpj depende do tipo de pessoa
ie_rg	VARCHAR(18)	documento insc estadual ou rg depende do tipo de pessoa
contato	VARCHAR(30)	nome do responsavel na empresa
email	VARCHAR(100)	endereço de email
homepage	VARCHAR(100)	pagina de internet
obs	VARCHAR(TEXT)	obs que achar necessarias sobre o cliente

Tabela telefones

Campo	Tipo	Descrição
id_fone	SERIAL	id do telefone
nome	VARCHAR(30))	nome do contato
id_pessoa	INTEGER	id da pessoa
tipo_fone	INTEGER	celular / comercial / residencial / fax
telefone	VARCHAR(30)	numero do telefone

Tabela tipo_equipamento

Campo	Tipo	Descrição
id_tipo_equipamento	INTEGER	id do tipo de equipamento
descricao	VARCHAR(50)	descrição do tipo ex. monitor / gabinete / equip de fax

Tabela tipo_peca		
Campo	Tipo	Descrição
id_tipo_peca	INTEGER	id do tipo da peça
descricao	VARCHAR(50)	descrição do tipo de peça ex. memoria hd

Tabela tipomov		
Campo	Tipo	Descrição
id_tipomov	INTEGER	id do tipo de movimento
tipomov	VARCHAR(1)	entrada ou saida
descricao	VARCHAR(50)	descrição do movimento = pag de duplicata nº / recebimento de os nº

Tabela uf		
Campo	Tipo	Descrição
id_uf	INTEGER	id da unidade de federação
abreviada	VARCHAR(50)	uf abreviada ex. PR
descricao_uf	VARCHAR(50)	uf completa ex. Paraná

Tabela usuario		
Campo	Tipo	Descrição
id_usuario	INTEGER	id do usuario
id_grupo	INTEGER	id do grupo
nome	VARCHAR(30)	nome de login
nomecompleto	VARCHAR(50)	nome completo
senha	VARCHAR(30)	senha
status	INTEGER	ativo ou inativo
ramal	VARCHAR(30)	ramal
email	VARCHAR(100)	email
cargo	VARCHAR(50)	cargo
datacadastro	DATE	data de cadastro do usuario
datault acesso	DATE	data do ultimo acesso ao sistema

5 CAMADA DE PERSISTÊNCIA

Classe Tacesso (Tacesso.pas)		
Propriedade	Tabela	Campo
P_id_usuario	acesso	id_usuario
P_id_menu	acesso	id_menu
P_nivel	acesso	nivel

Manutenibilidade Dados

Gravar

Inserir

insert into acesso (id_usuario, id_menu, nivel) values (:p_Usu, :p_Menu, :p_Nivel)

Alterar

update acesso set Nivel = :p_Nivel where ((Id_Usuario = :p_Usu) And (Id_Menu = :p_Menu))

Classe Tcaixa (Tcaixa.pas)		
Propriedade	Tabela	Campo
P_id_lcto	caixa	id_lcto
P_id_pessoa	caixa	id_pessoa
P_id_tipomov	caixa	id_tipomov
P_num_doc	caixa	num_doc
P_num_parcela	caixa	num_parcela
P_data_emissao	caixa	data_emissao
P_data_vcto	caixa	data_vcto

P_data_prorrog	caixa	data_prorrog
P_data_pgto	caixa	data_pgto
P_tipo_pgto	caixa	tipo_pgto
P_valor_apagar	caixa	valor_apagar
P_valor_juro	caixa	valor_juro
P_valor_pago	caixa	valor_pago
P_situacao	caixa	situacao
P_obs	caixa	obs

Manutenibilidade Dados

Gravar

Inserir

Insert into caixa (num_doc, id_tipomov, id_pessoa, num_parcela, data_emissao, data_vcto, data_pgto, data_prorrog, tipo_pgto, valor_apagar, valor_juro, valor_pago, situacao, obs) Values (:P_num_doc, :P_id_tipomov, :P_id_pessoa, :P_num_parcela, :P_data_emissao, :P_data_vcto, :P_data_pgto, :P_data_prorrog, :P_tipo_pgto, :P_valor_apagar, :P_valor_juro, :P_valor_pago, :P_situacao, :P_obs)

Alterar

Update caixa set num_doc = :P_num_doc, id_tipomov = :P_id_tipomov, id_pessoa = :P_id_pessoa, num_parcela = :P_num_parcela, data_emissao = :P_data_emissao, data_vcto = :P_data_vcto, data_pgto = :P_data_pgto, data_prorrog = :P_data_prorrog, tipo_pgto = :P_tipo_pgto, valor_apagar = :P_valor_apagar, valor_juro = :P_valor_juro, valor_pago = :P_valor_pago, situacao = :P_situacao, obs = :P_obs where (id_lcto = :p_id_lcto)

Ler

select caixa.* from tipomov INNER JOIN caixa ON (tipomov.id_tipomov = caixa.id_tipomov) where (data_vcto >= :p_diaini) and (data_vcto <= :p_diafim) and (tipomov.tipomov = :p_tipomov) order by data_vcto, id_tipomov

Classe Tcidade (Tcidade.pas)		
Propriedade	Tabela	Campo
P_id_cidade	cidade	id_cidade
P_id_uf	cidade	id_uf
P_descricao	cidade	descricao

Manutenibilidade Dados

Gravar

Inserir

Insert into cidade (id_uf, descricao) Values (:P_id_uf, :p_descricao)

Alterar

Update cidade set descricao = :p_descricao, id_uf = :P_id_uf where (id_cidade = :p_id_cidade)

Ler

select * from cidade where ((id_cidade = :p_id_cidade) and (id_uf = :p_id_uf))

Classe Tcontatos (Tcontatos.pas)		
Propriedade	Tabela	Campo
P_id_pessoa	contatos	id_pessoa
P_nome	contatos	nome
P_dia_nasc	contatos	dia_nasc
P_mes_nasc	contatos	mes_nasc
P_cargo	contatos	cargo

P_setor	contatos	setor
P_email	contatos	email

Manutenibilidade Dados

Gravar

Inserir

Insert into contatos (id_pessoa, nome, dia_nasc, mes_nasc, cargo, setor, email) Values (:p_id_pessoa, :p_nome, :p_dia_nasc, :p_mes_nasc, :p_cargo, :p_setor, :p_email)

Alterar

Update contatos set dia_nasc = :p_dia_nasc, mes_nasc = :p_mes_nasc, cargo = :p_cargo, setor = :p_setor, email = :p_email where (id_pessoa = :p_id_pessoa) and (nome = :p_nome)

Ler

select * from contatos where (id_pessoa = :p_id_pessoa) order by id_pessoa, nome

Classe Tcontrato (Tcontrato.pas)		
Propriedade	Tabela	Campo
P_id_contrato	contrato	id_contrato
P_id_pessoa	contrato	id_pessoa
P_data_inicio_contrato	contrato	data_inicio_contrato
P_data_fim_vigencia	contrato	data_fim_vigencia
P_data_inicio_vigencia	contrato	data_inicio_vigencia
P_data_bloqueio	contrato	data_bloqueio
P_valor_mensal	contrato	valor_mensal
P_dia_vcto	contrato	dia_vcto

P_dias_prorog	contrato	dias_prorog
P_bloqueado	contrato	bloqueado

Manutenibilidade Dados

Gravar

Inserir

Insert into contrato (id_pessoa, data_inicio_contrato, data_fim_vigencia, data_inicio_vigencia, data_bloqueio, valor_mensal, dia_vcto, dias_prorog, bloqueado) Values (:P_id_pessoa, :P_data_inicio_contrato, :P_data_fim_vigencia, :P_data_inicio_vigencia, :P_data_bloqueio, :P_valor_mensal, :P_dia_vcto, :P_dias_prorog, :P_bloqueado)

Alterar

Update contrato set id_pessoa = :P_id_pessoa, data_inicio_contrato = :P_data_inicio_contrato, data_fim_vigencia = :P_data_fim_vigencia, data_inicio_vigencia = :P_data_inicio_vigencia, data_bloqueio = :P_data_bloqueio, valor_mensal = :P_valor_mensal, dia_vcto = :P_dia_vcto, dias_prorog = :P_dias_prorog, bloqueado = :P_bloqueado where (id_contrato = :p_id_contrato)

Ler

select * from contrato order by id_pessoa,id_contrato

Classe Tequipamento (Tequipamento.pas)		
Propriedade	Tabela	Campo
P_id_equipamento	equipamento	id_equipamento
P_id_tipo_equipamento	equipamento	id_tipo_equipamento
P_id_contrato	equipamento	id_contrato
P_descricao	equipamento	descricao
P_numero_serie	equipamento	numero_serie

Manutenibilidade Dados

Gravar

Inserir

Insert into equipamento (id_tipo_equipamento, id_contrato, descricao, numero_serie) Values (:p_id_tipo_equipamento, :p_id_contrato, :p_descricao, :p_numero_serie)

Alterar

Update equipamento set id_tipo_equipamento = :p_id_tipo_equipamento, id_contrato = :p_id_contrato, descricao = :p_descricao, numero_serie = :p_numero_serie where (id_equipamento = :p_id_equipamento)

Ler

select * from equipamento where (id_contrato = :p_id_contrato) order by id_contrato,id_equipamento

Classe Tgrupo (Tgrupo.pas)		
Propriedade	Tabela	Campo
P_id_grupo	grupo	id_grupo
P_nomegrupo	grupo	nomegrupo

Manutenibilidade Dados

Gravar

Inserir

Insert into grupo (nomegrupo) Values (:p_nomegrupo)

Alterar

Update grupo set nomegrupo = :p_nomegrupo where (Id_Grupo = :p_Id_grupo)

Ler

select * from grupo order by id_grupo, nomegrupo

Classe TItensOrdemDeServico (TItensOrdemDeServico.pas)		
Propriedade	Tabela	Campo
P_id_os	item_os	id_os
P_id_item_os	item_os	id_item_os
P_id_equipamento	item_os	id_equipamento
P_id_contrato	item_os	id_contrato
P_tipo_atendimento	item_os	tipo_atendimento
P_desc_equip	item_os	desc_equip
P_desc_prob	item_os	desc_prob

Manutenibilidade Dados

Gravar

Inserir

Insert into item_os (id_os, desc_equip, desc_prob, tipo_atendimento, id_contrato, id_equipamento)Values (:p_id_os, :p_desc_equip, :p_desc_prob, :P_tipo_atendimento, :p_id_contrato, :p_id_equipamento)

Alterar

Update item_os set id_os = :p_id_os, id_contrato = :p_id_contrato, id_equipamento = :p_id_equipamento, tipo_atendimento = :P_tipo_atendimento, desc_equip = :p_desc_equip, desc_prob = :p_desc_prob where (id_item_os = :p_id_item_os)

Excluir

delete from item_os where ((id_os = :p_id_os) and (id_item_os = :p_id_item_os))

Ler

select * from item_os where (id_os = :p_id_os) order by id_item_os, id_os

Classe TSubItensOrdemDeServico (TSubItensOrdemDeServico.pas)		
Propriedade	Tabela	Campo
P_id_sitem	item_os_mov	id_sitem
P_id_item_os	item_os_mov	id_item_os
P_id_os	item_os_mov	id_os
P_id_contrato	item_os_mov	id_contrato
P_id_peca	item_os_mov	id_peca
P_qtde_peca	item_os_mov	qtde_peca
P_descricao	item_os_mov	descricao
P_cobrar	item_os_mov	cobrar

Manutenibilidade Dados

Gravar

Inserir

Insert into item_os_mov (id_os, id_item_os, descricao, cobrar, id_peca, qtde_peca) Values (:p_id_os, :p_id_item_os, :p_descricao, 1, :p_id_peca, :p_qtde_peca)

Alterar

Update item_os_mov set id_peca = :p_id_peca, qtde_peca = :p_qtde_peca, descricao = :p_descricao where ((id_Sitem = :p_id_Sitem) and (id_os = :p_id_os) and (id_item_os = :p_id_item_os))

Excluir

delete from item_os_mov where ((id_Sitem = :p_id_Sitem) and (id_os = :p_id_os) and (id_item_os = :p_id_item_os))

Excluir

delete from item_os_mov where (id_item_os = :p_id_item_os) and (id_os = :p_id_os)

Ler

select * from item_os_mov where (id_item_os = :p_id_item_os) and (id_os = :p_id_os) order by id_item_os, id_sitem

Classe Tlogs_os (Tlogs_os.pas)		
Propriedade	Tabela	Campo
P_id_log	log_os	id_log
P_id_os	log_os	id_os
P_id_usuario	log_os	id_usuario
P_acao	log_os	acao
P_data	log_os	data
Classe Tmenu (Tmenu.pas)		
Propriedade	Tabela	Campo
P_id_menu	menu	id_menu
P_tipo_menu	menu	tipo_menu
P_descricao	menu	descricao
P_id_modulo	menu	id_modulo
P_sequencia	menu	sequencia
P_menupai	menu	menupai

Manutenibilidade Dados

Gravar

Inserir

Insert into menu (Id_Menu, Tipo_Menu, Descricao, Sequencia, MenuPai, Id_Modulo) Values (:p_Id_Menu, :p_Tipo_Menu, :p_Descricao, :p_Sequencia, :p_MenuPai, :p_Id_Modulo)

Alterar

Update menu set Tipo_Menu = :p_Tipo_Menu, Sequencia = :p_Sequencia, MenuPai = :p_MenuPai, Descricao = :p_Descricao,Id_Modulo = :p_Id_Modulo where (Id_Menu = :p_Id_Menu)

Ler

select * from menu order by id_modulo, tipo_menu, sequencia

Classe Tos (Tos.pas)		
Propriedade	Tabela	Campo
P_id_os	os	id_os
P_id_pessoa	os	id_pessoa
P_data_cad	os	data_cad
P_data_alt	os	data_alt
P_data_fechou	os	data_fechou
P_data_canc	os	data_canc
P_status	os	status
P_valor_pecas	os	valor_pecas
P_valor_orcado	os	valor_orcado
P_valor_cobrado	os	valor_cobrado
P_valor_mao_obra	os	valor_mao_obra
P_valor_pecas_ncobrado	os	valor_pecas_ncobrado
P_valor_desconto_mo	os	valor_desconto_mo
P_valor_desconto_pecas	os	valor_desconto_pecas

Manutenibilidade Dados

Gravar

Inserir

Insert into os (id_pessoa, status, data_cad, valor_orcado, valor_cobrado, valor_mao_obra,valor_pecas, valor_pecas_ncobrado, valor_desconto_mo, valor_desconto_pecas) Values (:p_id_pessoa, :p_status, :p_data_cad, :p_valor_orcado, :p_valor_cobrado, :p_valor_mao_obra, :p_valor_pecas, :p_valor_pecas_ncobrado, :p_valor_desconto_mo, :p_valor_desconto_pecas)

Alterar

Update os set id_pessoa = :p_id_pessoa, status = :p_status, data_cad = :p_data_cad, data_alt = :p_data_alt, data_fechou = :p_data_fechou, data_canc = :p_data_canc, valor_orcado = :p_valor_orcado, valor_cobrado = :p_valor_cobrado, valor_mao_obra = :p_valor_mao_obra, valor_pecas = :p_valor_pecas, valor_pecas_ncobrado = :p_valor_pecas_ncobrado, valor_desconto_mo = :p_valor_desconto_mo, valor_desconto_pecas = :p_valor_desconto_pecas where (id_os = :p_id_os)

Excluir

delete from os where (id_os = :p_id_os)

Ler

select * from os where (id_os = :p_id_os) and (status = :p_status) order by id_os desc

Classe Tpeca (Tpeca.pas)		
Propriedade	Tabela	Campo
P_id_peca	peca	id_peca
P_id_tipo_peca	peca	id_tipo_peca
P_descricao	peca	descricao
P_quantidade	peca	quantidade
P_valor_unitario	peca	valor_unitario

Manutenibilidade Dados

Gravar

Inserir

Insert into peca (id_tipo_peca, descricao, quantidade, valor_unitario) Values (:p_id_tipo_peca, :p_descricao, :p_quantidade, :p_valor_unitario)

Alterar

Update peca set id_tipo_peca = :p_id_tipo_peca, descricao = :p_descricao, quantidade = :p_quantidade, valor_unitario = :p_valor_unitario where (id_Peca = :P_id_Peca)

Ler

select * from peca order by id_Peca, id_tipo_peca, descricao

Classe TPessoa (TPessoa .pas)		
Propriedade	Tabela	Campo
P_id_pessoa	pessoa	id_pessoa
P_id_uf	pessoa	id_uf
P_id_cidade	pessoa	id_cidade
P_nome	pessoa	nome
P_endereco	pessoa	endereco
P_bairro	pessoa	bairro
P_cep	pessoa	cep
P_tipo	pessoa	tipo
P_tipo_pessoa	pessoa	tipo_pessoa
P_cgc_cnpj	pessoa	cgc_cnpj
P_ie_rg	pessoa	ie_rg
P_contato	pessoa	contato
P_email	pessoa	email

P_homepage	pessoa	homepage
P_obs	pessoa	obs

Manutenibilidade Dados

Gravar

Inserir

Insert into pessoa (id_cidade, id_uf, nome, endereco, bairro, cep, ie_rg, cgc_cnpj, tipo_pessoa, tipo, contato, email, homepage, obs) Values (:p_id_cidade, :p_id_uf, :p_nome, :p_endereco, :p_bairro, :p_cep, :p_ie_rg, :p_cgc_cnpj, :p_tipo_pessoa, :p_tipo, :p_contato, :p_email, :p_homepage, :p_obs)

Alterar

Update pessoa set id_cidade = :p_id_cidade, id_uf = :p_id_uf, nome = :p_nome, endereco = :p_endereco, bairro = :p_bairro, cep = :p_cep, ie_rg = :p_ie_rg, cgc_cnpj = :p_cgc_cnpj, tipo_pessoa = :p_tipo_pessoa, tipo = :p_tipo, contato = :p_contato, email = :p_email, homepage = :p_homepage, obs = :p_obs where (id_pessoa = :p_id_pessoa)

Ler

select * from pessoa where (tipo = :p_tipo) order by id_pessoa, nome

Classe Ttelefones (Ttelefones.pas)		
Propriedade	Tabela	Campo
P_id_fone	telefones	id_fone
P_nome	telefones	nome
P_id_pessoa	telefones	id_pessoa
P_tipo_fone	telefones	tipo_fone
P_telefone	telefones	telefone

Manutenibilidade Dados

Gravar

Inserir

Insert into telefones (id_pessoa, nome, telefone, tipo_fone) Values

(:p_id_pessoa, :p_nome, :p_telefone, :p_tipo_fone)

Alterar

Update telefones set telefone = :p_telefone, tipo_fone = :p_tipo_fone where

((id_fone = :p_id_fone) and (id_pessoa = :p_id_pessoa) and (nome = :p_nome))

Excluir

delete from telefones where (id_fone = :p_id_fone) and (id_pessoa =

:p_id_pessoa) and (Nome = :p_nome)

Ler

select * from telefones where (id_pessoa = :p_id_pessoa) and (nome =

:p_nome) order by id_pessoa, nome

Classe Ttipo_equipamento (Ttipo_equipamento.pas)		
Propriedade	Tabela	Campo
P_id_tipo_equipamento	tipo_equipamento	id_tipo_equipamento
P_descricao	tipo_equipamento	descricao

Manutenibilidade Dados

Gravar

Inserir

Insert into tipo_equipamento (descricao) Values (:p_descricao)

Alterar

Update tipo_equipamento set descricao = :p_descricao where

(id_tipo_equipamento = :p_id_tipo_equipamento)

Ler

Select * from tipo_equipamento order by id_tipo_equipamento, descrição

Classe Ttipo_peca (Ttipo_peca.pas)		
Propriedade	Tabela	Campo
P_id_tipo_peca	tipo_peca	id_tipo_peca
P_descricao	tipo_peca	descricao

Manutenibilidade Dados

Gravar

Inserir

Insert into tipo_peca (descricao) Values (:p_descricao)

Alterar

Update tipo_peca set descricao = :p_descricao where (id_tipo_peca = :p_id_tipo_peca)

Ler

select * from tipo_peca order by id_tipo_peca, descricao

Classe Ttipomov (Ttipomov.pas)		
Propriedade	Tabela	Campo
P_id_tipomov	tipomov	id_tipomov
P_tipomov	tipomov	tipomov
P_descricao	tipomov	descricao

Manutenibilidade Dados

Gravar

Inserir

Insert into tipomov (tipomov, descricao) Values (:P_tipomov, :p_descricao)

Alterar

Update tipomov set descricao = :p_descricao, tipomov = :P_tipomov where (id_tipomov = :p_id_tipomov)

Ler

select * from tipomov order by id_tipomov,tipomov, descricao

Classe Tuf (Tuf.pas)		
Propriedade	Tabela	Campo
P_id_uf	uf	id_uf
P_abreviada	uf	abreviada
P_descricao_uf	uf	descricao_uf

Manutenibilidade Dados

Gravar

Inserir

Insert into uf (abreviada, descricao) Values (:P_abreviada, :p_descricao)

Alterar

Update uf set abreviada = :p_abreviada, descricao = :p_descricao where (id_uf = :p_id_uf)

Ler

select * from uf order by id_uf,abreviada, descricao

Classe Tusuario (Tusuario.pas)		
Propriedade	Tabela	Campo
P_id_usuario	usuario	id_usuario
P_id_grupo	usuario	id_grupo
P_nome	usuario	nome
P_nomecompleto	usuario	nomecompleto
P_senha	usuario	senha
P_status	usuario	status
P_ramal	usuario	ramal

BIBLIOTECA DA ESCOLA TÉCNICA DA UFPR

P_email	usuario	email
P_cargo	usuario	cargo
P_datacadastro	usuario	datacadastro
dataultacesso	usuario	dataultacesso

Manutenibilidade Dados

Gravar

Inserir

insert into usuario (Nome, NomeCompleto, Status, Id_Grupo, Ramal, EMail, Cargo, DataCadastro, DataUltAcesso, Senha) Values (:p_Nome, :p_NomeCompl, :p_Status, :p_Id_Grupo, :p_Ramal, :p_EMail, :p_Cargo, :p_DataCadastro, :p_DataUltAcesso, :p_senha)

Alterar

Update usuario set Nome = :p_Nome, NomeCompleto = :p_NomeCompl, Status = :p_Status, Id_Grupo = :p_Id_Grupo, Ramal = :p_Ramal, EMail =:p_EMail, Cargo = :p_Cargo, DataUltAcesso = :p_DataUltAcesso, Senha = :p_senha where Id_Usuario = :p_Id_Usuario

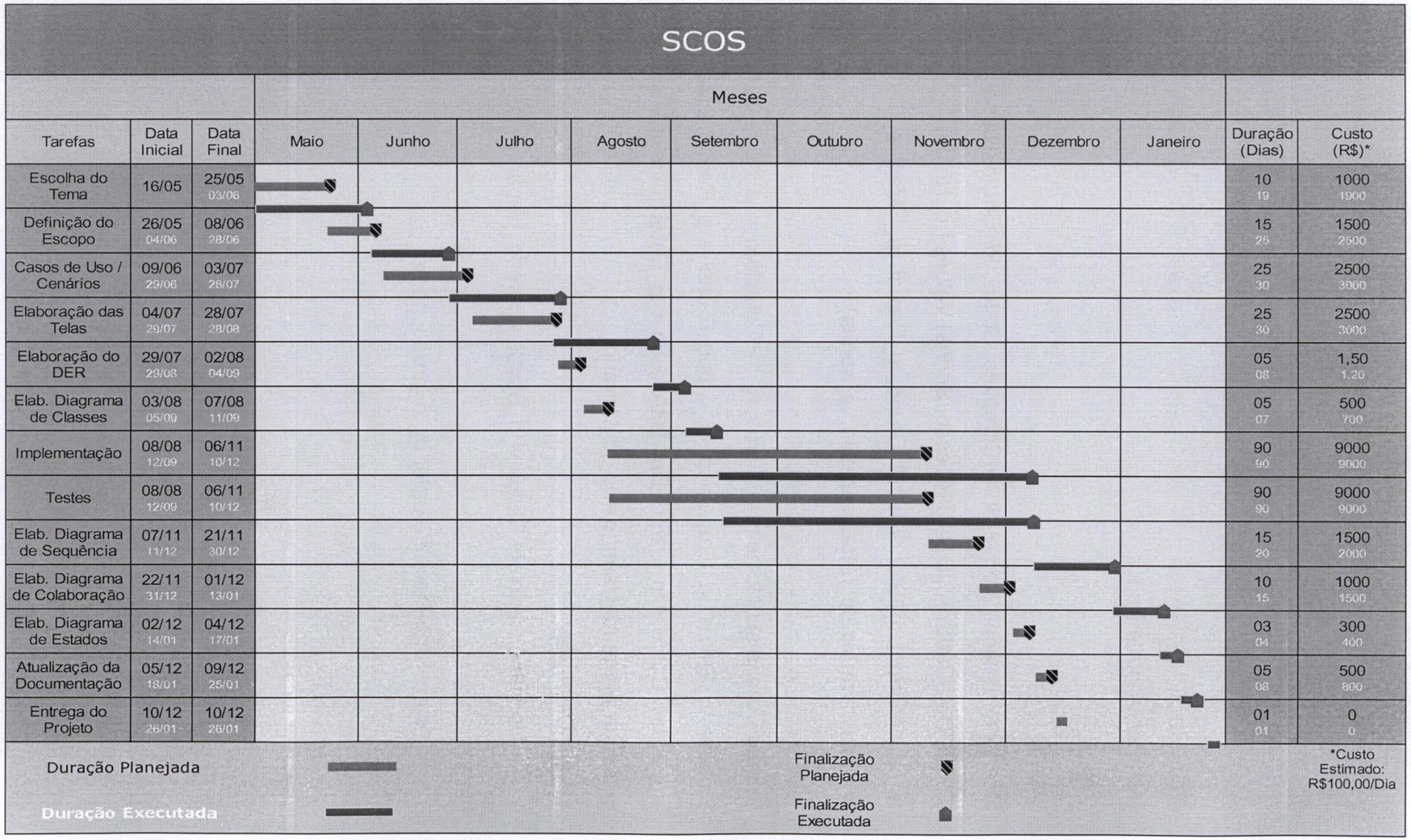
Ativar / Desativar

Update usuario set Status = :p_Status where Id_Usuario = :p_Id_Usuario

Ler

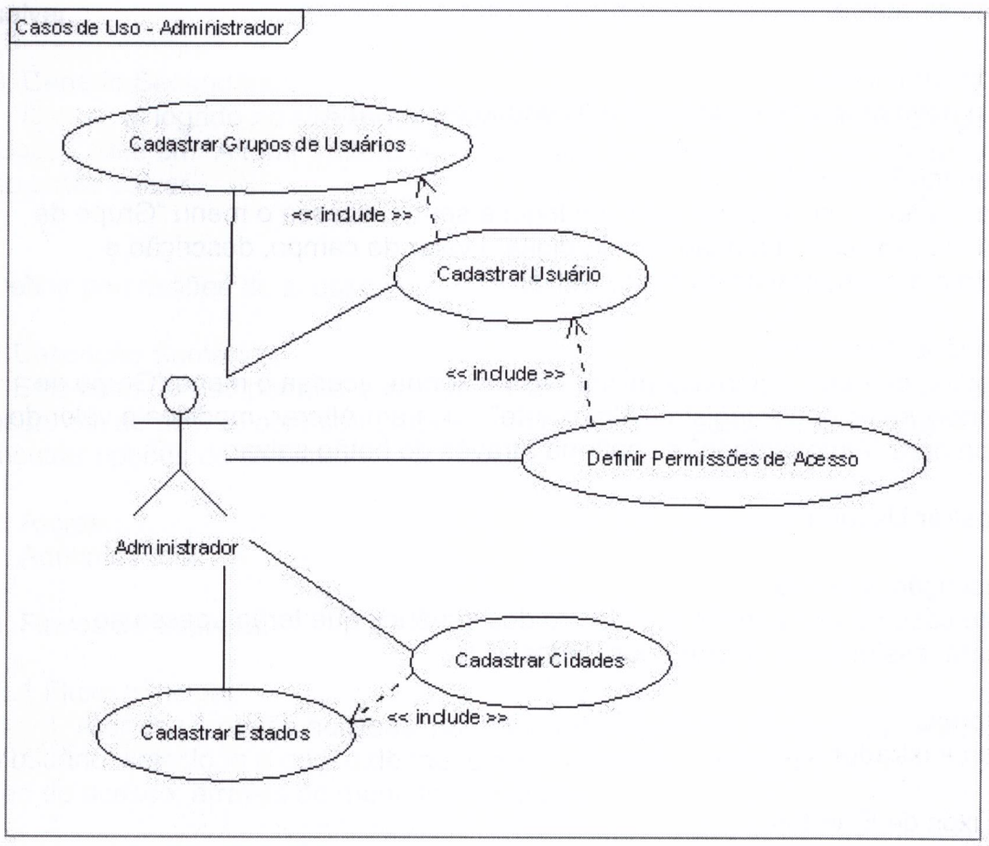
Select * from usuario order by id_grupo, nome

6 CRONOGRAMA



7 DIAGRAMAS DE CASOS DE USO

7.1.1 Caso de uso – Administrador:



Cadastrar Grupos de Usuários:

1.Descrição Sumária:
Este caso de uso permite o cadastro dos grupos de usuários que terão acesso ao sistema, para possibilitar a definição de perfis de acesso com diferentes características. Também permite que as informações do grupo sejam alteradas.

2. Atores:
Administrador.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator acessa a opção grupo de usuários, no menu Cadastros, usa a opção incluir, digita o nome e descrição do grupo de usuário e confirma.

2. Ator acessa a opção grupo de usuários, no menu Cadastros, seleciona um grupo, usa a opção alterar, altera os campos necessários e confirma.

3.2 Fluxo secundário:

1. Caso o ator não selecione um grupo e use a opção alterar, o sistema informa que deve ser selecionado um grupo.

5. Pré-Condições:

Não se aplica.

6. Pós-Condições:

Um novo grupo de usuários é adicionado ao sistema.

7. Cenário Principal:

Carlos, entra no sistema com seu login e senha, acessa o menu "Grupo de Usuários", executa a função Incluir, digita o valor do campo, descrição e confirma a ação através do botão Salvar.

8. Cenário Alternativo:

Carlos, entra no sistema com seu login e senha, acessa o menu "Grupo de Usuários, seleciona o registro "Almoxarife", clica em Alterar, modifica o valor do campo para "Almoxarifado" e confirma através do botão salvar.

Cadastrar Usuário:

1.Descrição Sumária:

Este caso de uso permite o cadastro dos usuários que terão acesso ao sistema, assim como alterar seus dados.

2. Atores:

Administrador.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção usuários, no menu Administração do Sistema, usa a opção incluir, define login, senha, escolhe o grupo do usuário, adiciona demais dados cadastrais e confirma.

2. Ator acessa a opção usuários, no menu Administração do Sistema, seleciona um registro de usuário, usa a opção alterar, altera os dados necessários e confirma.

3.2 Fluxo secundário:

1. Caso o ator não selecione um registro e use a opção alterar, o sistema informa que um registro deve ser selecionado.

2. Caso o ator não informe o login ou senha para o usuário, o sistema avisa para que sejam informados os respectivos dados.

5. Pré-Condições:

O grupo de usuário deve estar previamente cadastrado no sistema.

6. Pós-Condições:
Um novo usuário é adicionado ao sistema.

7. Cenário Principal:
Carlos, já logado no sistema, acessa o menu "Usuários", clica em "Incluir" e cadastra os dados do novo usuário "João", define seu login, senha, seleciona o grupo "Almoxarifado" como seu perfil de acesso e confirma através do botão salvar.

8. Cenário Secundário:
Carlos, já logado no sistema, acessa o menu "Usuários", seleciona o registro "João", clica em "Alterar", altera seu login para "JoãoPedro" e confirma através do botão salvar.

Definir permissões de acesso:

1.Descrição Sumária:
Este caso de uso permite que sejam definidas as permissões de acesso dos usuários, de acordo com seu grupo e perfil desejado, possibilitando ocultar / mostrar opções do sistema.

2. Atores:
Administrador.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator usa a opção acessos, no menu Administração do Sistema, escolhe o usuário, seleciona a opção de menu a ser alterado o acesso e determina o tipo do acesso, através do menu lateral direito.

3.2 Fluxo secundário:
1. Caso o ator selecione a opção de tipo de acesso, sem selecionar a opção de menu a ser alterada, o sistema informa que uma opção deve ser selecionada.

5. Pré-Condições:
O usuário deve estar previamente cadastrado no sistema.

6. Pós-Condições:
São alteradas as permissões de acesso de usuário.

7. Cenário Principal:
Carlos clica no menu "Acessos", escolhe o usuário "João", seleciona a opção "Pesquisar OS" e determina o acesso como "Completo".

Cadastrar Estados:

1.Descrição Sumária:
Este caso de uso permite o cadastro de Estados (UF) ao sistema, para ampliar a gama de Estados pré-existentes, caso haja necessidade. Também permite a alteração de um Estado já cadastrado.

2. Atores:
Administrador.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator acessa a opção Estados (UF), no menu Cadastros, usa a opção incluir, entra com a sigla e a descrição do Estado e confirma.
2. Ator acessa a opção Estados (UF), no menu Cadastros, seleciona um Estado (UF), usa a opção alterar, altera os campos necessários e confirma.

3.2 Fluxo secundário:
1. Caso o ator não informe a sigla do Estado, o sistema avisa que seja informada a sigla.
2. Caso o ator não selecione um Estado e use a opção alterar, o sistema informa que um Estado deve ser selecionado.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
Um novo Estado (UF) é adicionado ao sistema.

7. Cenário Principal:
Carlos acessa o menu Estados (UF), clica na opção "Incluir", digita "RS" no campo sigla e "Rio Grande do Sul" no campo descrição do Estado e confirma através do botão salvar.

8. Cenário Secundário:
Carlos acessa o menu Estados (UF), seleciona o registro "RS", clica na opção "Alterar", digita "RN" no campo sigla e "Rio Grande do Norte" no campo descrição do Estado e confirma através do botão salvar.

Cadastrar Cidades:

1.Descrição Sumária:
Este caso de uso permite o cadastro de Cidades no sistema, permitindo ampliar a gama de Cidades pré-cadastradas, caso seja necessário. Também permite a alteração de uma Cidade já cadastrada.

2. Atores:
Administrador.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção Cidades, no menu Cadastro, seleciona a opção incluir, seleciona o Estado (UF), informa o nome da Cidade e confirma.
2. Ator acessa a opção Cidades, no menu Cadastro, seleciona um registro de Cidade, seleciona a opção alterar, seleciona o Estado (UF), altera o nome da Cidade e confirma.

3.2 Fluxo secundário:

1. Caso o ator não selecione o Estado (UF), o sistema avisa que é necessário selecionar o Estado (UF).
2. Caso o ator não selecione um registro de Cidade, o sistema informa que um registro deve ser selecionado.

5. Pré-Condições:

O Estado (UF) deve estar previamente cadastrado no sistema.

6. Pós-Condições:

Uma nova Cidade é adicionada ao sistema.

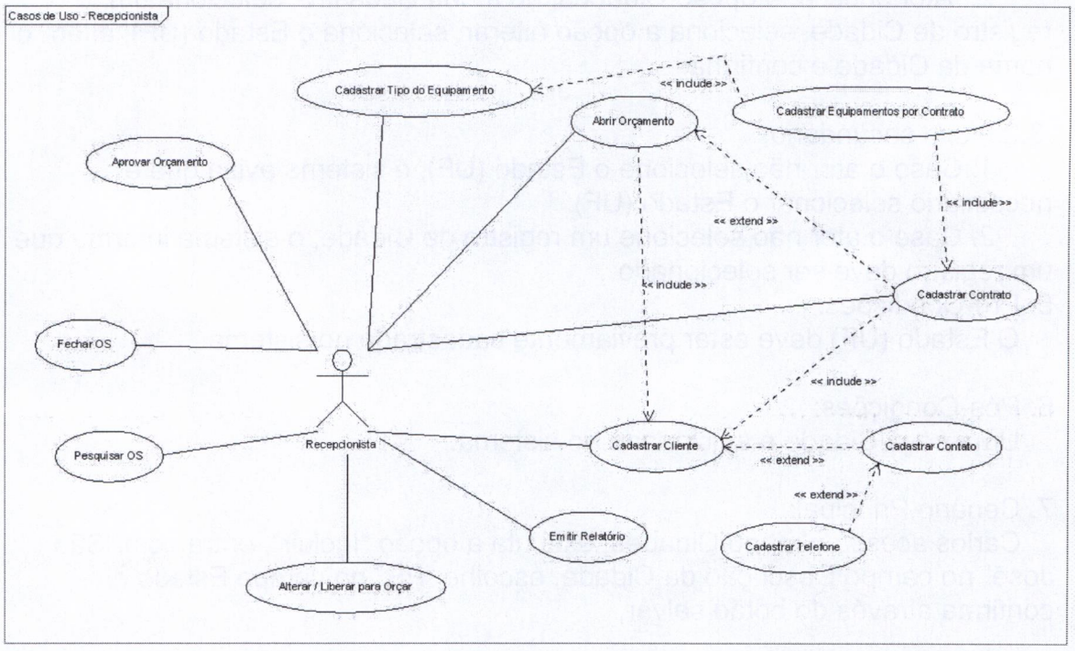
7. Cenário Principal:

Carlos acessa o menu Cidades, executa a opção "Incluir", entra com "São José" no campo Descrição da Cidade, escolhe "PR" no campo Estado e confirma através do botão salvar.

8. Cenário Secundário:

Carlos acessa o menu Cidades, seleciona o registro "São José", executa a opção Alterar, escolhe "SC" no campo Estado e confirma através do botão salvar.

7.1.2 Caso de uso – Recepcionista:



Cadastrar Cliente:

1.Descrição Sumária:
Este caso de uso permite o cadastro de clientes no sistema, para a abertura de orçamentos e ordens de serviço. Também permite a alteração de cadastro, inclusão de contatos e observações.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção clientes, no menu Cadastros, usa a opção incluir, preenche os campos de cadastro e confirma.
2. Ator seleciona a opção clientes, no menu Cadastros, seleciona um registro de Cliente, usa a opção alterar, altera os campos necessários e confirma.
3. Ator, dentro de um registro de Cliente, acessa a aba Observações, insere uma observação e confirma.

3.2 Fluxo secundário:
1. Caso o ator não preencha corretamente o campo CPF ou CNPJ, o sistema informa que o CPF / CNPJ está incorreto.
2. Caso o ator não selecione um registro de cliente ao usar a opção alterar, o sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
Um novo cliente é adicionado ao sistema.

7. Cenário Principal:
Vanessa, acessa a opção "Clientes", clica em "Incluir" adiciona os dados do cliente "Paulo Farias", escolhe a opção "Pessoa Física", entra com o CPF, RG e demais dados e confirma, através do botão salvar.

8. Cenário Secundário:
Vanessa, acessa a opção "Clientes", clica em "Incluir" adiciona os dados do cliente "Paulo Farias", escolhe a opção "Pessoa Física", entra com o CPF e ao acessar o campo RG, o sistema informa "CPF Inválido", então Vanessa verifica o número e o altera, entra com o RG e demais dados e confirma, através do botão salvar.

Cadastrar Contato:

1.Descrição Sumária:
Este caso de uso permite a inclusão e / ou alteração dos contatos de clientes previamente cadastrados no sistema.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator, dentro de um registro de Cliente, acessa a aba Contatos, usa a opção incluir, insere os dados do contato e confirma.
2. Ator, dentro de um registro de Cliente, acessa a aba Contatos, seleciona um registro de contato e usa a opção alterar dados, altera os campos necessários e confirma.

3.2 Fluxo secundário:
1. Caso o ator não preencha o campo nome e confirme a inclusão do contato, o sistema informa que o campo deve ser preenchido.

5. Pré-Condições:
O cliente deve estar pré-cadastrado.

6. Pós-Condições:
Um novo contato é adicionado ao sistema.

7. Cenário Principal:

Vanessa acessa o registro do cliente “Pedro Faria”, seleciona a aba “Contatos”, clica com o botão direito e seleciona a opção “Incluir”, adiciona os dados do contato e confirma com a opção “Salvar”.

8. Cenário Secundário:
Vanessa acessa o registro do cliente “Pedro Faria”, seleciona a aba “Contatos”, clica com o botão direito e seleciona a opção “Alterar”, altera o campo “e-mail” e confirma com a opção “Salvar”.

Cadastrar Telefone:

1.Descrição Sumária:
Este caso de uso permite que sejam cadastrados e / ou alterados os telefones, de contatos de um cliente pré-cadastrado no sistema.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator, dentro de um registro de Cliente, acessa a aba Contatos, seleciona um registro de contato, usa a opção alterar, digita o número e o tipo do telefone e confirma.

3.2 Fluxo secundário:
1. Caso o ator não selecione o tipo do telefone, o sistema informa que deve ser escolhido o tipo do telefone.

5. Pré-Condições:
O cliente e o contato devem estar previamente cadastrados.

6. Pós-Condições:
O telefone do contato é adicionado ao sistema.

7. Cenário Principal:
Vanessa acessa o menu clientes, seleciona o registro de “Pedro Faria”, clica em “Alterar”, seleciona a aba “Contatos, escolhe o contato “Roberto”, clica com o botão direito e seleciona “Alterar”, clica em “Alterar Telefones”, digita o número do telefone, escolhe a opção “Celular” e confirma através do botão salvar.

Cadastrar Contrato:

1.Descrição Sumária:
Este caso de uso permite o cadastro de contratos de prestação de serviço, para pessoas jurídicas, permitindo estipular a data de vigência, prescrição, valor a ser cobrado (no caso de pagamento pré-definido) e data do vencimento.

2. Atores:

Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção contrato, no menu Cadastros, usa a opção incluir, pesquisa o cliente pelo número do CPF e / ou Razão Social, confirma, insere os dados de contrato e confirma.

2. Ator acessa a opção contrato, no menu Cadastros, seleciona o registro de contrato, usa a opção alterar, altera os dados de contrato e confirma.

3.2 Fluxo secundário:

1. Ator insere data de término do contrato menor que a data de início e o sistema informa que a data final do contrato é menor que a inicial.

5. Pré-Condições:

O cliente deve estar previamente cadastrado.

6. Pós-Condições:

O contrato é cadastrado no sistema.

7. Cenário Principal:

Pedro Faria leva sua impressora para conserto e informa a recepcionista Vanessa, que o mesmo gostaria de executar o serviço por meio de um contrato de prestação de serviços com a empresa. Vanessa verifica se o mesmo possui cadastro de cliente (através do menu clientes), acessa o menu contratos e clica em "Incluir", digita a data de início e de fim de contrato e conclui através do botão salvar.

Cadastrar Tipo do Equipamento:

1. Descrição Sumária:

Este caso de uso permite o cadastro de tipos de equipamentos contratuais, com fins de se definir categorias de equipamentos de contrato.

2. Atores:

Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção Tipos de Equipamentos Contratuais, no menu Cadastros, usa a opção Incluir, digita a descrição do tipo e confirma.

2. Ator acessa a opção Tipos de Equipamentos Contratuais, no menu Cadastros, seleciona um tipo de equipamento e usa a opção alterar, digita a nova descrição do tipo e confirma.

3.2 Fluxo secundário:

1. Ator não digita uma descrição do tipo (inclusão / alteração). Sistema avisa que o campo Descrição do Tipo deve ser preenchido.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
O tipo de equipamento contratual é cadastrado.

7. Cenário Principal:
A recepcionista Vanessa, verifica que o equipamento o qual o cliente Paulo Faria solicitou reparo, é da categoria "Impressoras", porém, Vanessa, acessa o menu Tipos de Equipamentos Contratuais e verifica que o tipo não está cadastrado, então, clica na opção "Incluir", preenche o campo "Descrição do Tipo" com "Impressoras" e confirma com o botão salvar.

Cadastrar Equipamentos por Contato:

1.Descrição Sumária:
Este caso de uso permite que sejam cadastrados equipamentos num contrato de cliente já pré-cadastrado. Também permite a alteração dos equipamentos já cadastrados.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator acessa a opção Equipamentos >> Contratos, no menu Cadastros, seleciona o registro do contrato, usa a opção incluir, escolhe o tipo do equipamento, digita seu número de série, a descrição e confirma.
2. Ator acessa a opção Equipamentos >> Contratos, no menu Cadastros, seleciona o registro do contrato, usa a opção alterar, altera os dados do equipamento e confirma.

3.2 Fluxo secundário:
1. Ator não seleciona o tipo do equipamento. O sistema informa que um tipo de equipamento deve ser selecionado.
2. Ator não preenche o campo número de série. O sistema informa que o campo número de série dever ser preenchido.

5. Pré-Condições:
O contrato e o tipo do equipamento devem estar previamente cadastrados.

6. Pós-Condições:
Os equipamentos são cadastrados ao contrato.

7. Cenário Principal:
O cliente Pedro Faria, leva sua impressora para que seja verificado um problema de impressão e é atendido pela Recepcionista Vanessa. Pedro que gostaria de efetuar o serviço através de um contrato de prestação de serviços com a empresa. Vanessa, então, verifica se o cliente está cadastrado (no

opção clientes, no menu cadastros), efetua o cadastro do contrato (na opção contratos, no menu cadastros), seleciona a opção Equipamentos >> Contratos, no menu cadastros, seleciona o contrato do cliente Pedro Faria, clica na opção incluir, escolhe o tipo de equipamento "Impressora", preenche o número de série, a descrição e confirma através do botão salvar.

Abrir Orçamento:

1.Descrição Sumária:
Este caso de uso permite a abertura de um orçamento, que após aprovado, virá a tornar-se uma Ordem de Serviço. Nele, são inclusos os equipamentos que serão reparados e também há a possibilidade de inclusão de peças as quais o cliente deseje agregar ao seu equipamento.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção abrir orçamento, do menu OS – Ordens de Serviço, efetua a pesquisa do cliente pelo nome (razão social) ou CPF, seleciona o cliente, determina o tipo de atendimento (no caso avulso / garantia), insere a descrição do equipamento e do problema, insere as peças que serão agregadas ao serviço, o valor da mão de obra (caso necessário) e finaliza.
2. Ator seleciona a opção abrir orçamento, do menu OS – Ordens de Serviço, efetua a pesquisa do cliente pelo nome (razão social) ou CPF, seleciona o cliente, determina o tipo de atendimento (no caso contrato), seleciona o contrato, os equipamentos, insere a descrição do problema, insere as peças que serão agregadas ao serviço, o valor da mão de obra (caso necessário) e finaliza.

3.2 Fluxo secundário:
1. Ator seleciona a opção contrato, no tipo de atendimento e o sistema informa que o cliente não possui contratos ativos, devendo o ator, então, selecionar a opção avulso ou garantia.
2. Ator seleciona peça a ser agregada ao equipamento e não preenche o campo descrição. O sistema informa que o campo descrição deve ser preenchido.

5. Pré-Condições:
O cliente e o contrato (caso seja via contrato) devem estar previamente cadastrados.

6. Pós-Condições:
É aberto um orçamento de Ordem de Serviço.

7. Cenário Principal:

O cliente Pedro Faria chega com seu equipamento e solicita reparo. A recepcionista Vanessa, no menu "Abrir Orçamento", verifica se o mesmo possui cadastro e, constatando o cadastro existente, abre uma solicitação de orçamento para o cliente, escolhe o tipo de atendimento como "Avulso", inclui o equipamento e descrição dos problemas apresentados, selecionando a opção "Inserir Equipamento", acrescenta a peça "Gravador de CD", através da opção "Incluir", no menu de Peças dos Equipamentos da OS e finaliza.

8. Cenário Secundário:
O cliente Pedro Faria chega com seu equipamento e solicita reparo. A recepcionista Vanessa, no menu "Abrir Orçamento", verifica se o mesmo possui cadastro e, constatando o cadastro existente, abre uma solicitação de orçamento para o cliente. Pedro informa que possui um contrato de prestação de serviços com a Empresa, portanto, Vanessa escolhe o tipo de atendimento como "Contrato", aonde aparece que há um contrato ativo. Ela seleciona o contrato, confirma, seleciona o equipamento que será reparado, confirma, inclui a descrição dos problemas apresentados, selecionando a opção "Inserir Equipamento", e finaliza.

Alterar / Liberar para Orçar:

1.Descrição Sumária:
Este caso de uso permite verificar orçamentos pré-cadastrados, para fins de eventuais alterações. Também permite que sejam liberados os orçamentos que não necessitem de novas alterações ou cancelados os orçamentos não aprovados pelos clientes, antes de serem enviados ao técnico, para orçar.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Alterar / Liberar para Orçar, no menu OS – Ordens de Serviço, seleciona o registro do orçamento e usa a opção Liberar para Efetivação do Orçamento.
2. Ator seleciona a opção Alterar / Liberar para Orçar, no menu OS – Ordens de Serviço, seleciona o registro do orçamento e usa a opção Alterar Orçamento, insere / remove ou modifica os equipamentos / peças necessários e confirma.
3. . Ator seleciona a opção Alterar / Liberar para Orçar, no menu OS – Ordens de Serviço, seleciona o registro do orçamento e usa a opção Cancelar Orçamento, e confirma.

3.2 Fluxo secundário:
1. Caso o ator não selecione um registro para Liberar / Cancelar, o sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
A abertura de orçamento deve estar previamente cadastrada.

6. Pós-Condições:
A solicitação de orçamento é enviada para o técnico para efetivação.

7. Cenário Principal:
Paulo Faria, ao efetivar a abertura de orçamento, pela recepcionista Vanessa, é indagado se há necessidade de se incluir algum item e se está de acordo com o que foi solicitado. Após a confirmação de Paulo, ela, na opção "Alterar / Liberar para "Orçar", seleciona o registro da solicitação de orçamento e confirma a liberação através da opção "Liberar para Efetivação de Orçamento".

8. Cenário Secundário:
Paulo Faria, ao efetivar a abertura de orçamento, pela recepcionista Vanessa, é indagado se há necessidade de se incluir algum item e se está de acordo com o que foi solicitado. Paulo solicita que também seja incluída mais uma impressora. Vanessa, na opção "Alterar / Liberar para Orçar", seleciona o registro da solicitação de orçamento e clica na opção "Alterar", inclui o novo equipamento e finaliza.

Aprovar Orçamentos:

1.Descrição Sumária:
Este caso de uso é usado para aprovar os orçamentos solicitados e já efetivados pelo Técnico e que foram aprovados pelo cliente. Também permite que visualizar e cancelar orçamentos não aprovados.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Aprovação Orçamentos, no menu OS – Ordens de Serviço, seleciona o registro do orçamento, usa a opção Aprovar e confirma.
2. Ator seleciona a opção Aprovação Orçamentos, no menu OS – Ordens de Serviço, seleciona o registro do orçamento, usa a opção Não Aprovado e confirma.
3. Ator seleciona a opção Aprovação Orçamentos, no menu OS – Ordens de Serviço, seleciona o registro do orçamento, usa a opção visualizar.

3.2 Fluxo secundário:
1. Ator não seleciona o registro de orçamento para Aprovar / Não Aprovar / Visualizar. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
A solicitação de orçamento deve estar previamente cadastrada, liberada para orçamento e orçada pelo técnico.

6. Pós-Condições:

O orçamento é aprovado e é encaminhado ao Técnico a Ordem de Serviço para execução.

7. Cenário Principal:
Paulo Faria é contactado pela Recepcionista Vanessa, a qual lhe informa que o custo do serviço é de R$120,00. Paulo concorda com o valor e Vanessa confirma o orçamento através da opção "Aprovar Orçamento", no menu "Aprovação Orçamentos".

8. Cenário Secundário:
Paulo Faria é contactado pela Recepcionista Vanessa, a qual lhe informa que o custo do serviço é de R$120,00. Paulo não concorda com o valor e Vanessa cancela o orçamento através da opção "Não Aprovado", no menu "Aprovação Orçamentos".

Fechar OS:

1.Descrição Sumária:
Este caso de uso permite o fechamento de Ordens de Serviço já executadas e liberadas pelo Técnico, gerando um movimento de contas à receber.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Fechar Ordens de Serviço, no menu OS – Ordens de Serviço, seleciona o registro da OS e usa a opção Fechar OS, verifica os valores e confirma o fechamento.

3.2 Fluxo secundário:
1. Ator não seleciona o registro da OS ao usar a opção Fechar OS. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
A ordem de serviço deve estar com status "Liberada", ou seja, já executada pelo Técnico.

6. Pós-Condições:
A ordem de serviço é encerrada.

7. Cenário Principal:
Paulo Faria recebe um telefonema da Recepcionista Vanessa, avisando-o que seu equipamento está pronto para retirada. Paulo comparece à empresa e Vanessa acessa a Ordem de Serviço referente, através do menu Fechar Ordem de Serviço, clica na opção "Fechar OS", recebe o pagamento de Paulo e clica em "Fechar a Ordem de Serviço".

Pesquisar OS:

1.Descrição Sumária:
Este caso de uso permite que sejam pesquisadas Ordens de Serviço pelo seu Status, que podem ser: Todas, canceladas, orçamentos, aprovadas, em andamento, aguardando peça, liberadas, fechadas, orçar, orçado, recusada e devolver.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Pesquisar OS, no menu OS – Ordens de Serviço, seleciona o status de visualização, seleciona o registro da OS e usa a opção Visualizar.

3.2 Fluxo secundário:
1. Ator não seleciona um registro de OS e usa a opção visualizar. O sistema informa que um registro dever ser selecionado.

5. Pré-Condições:
Uma solicitação de orçamento ou uma Ordem de Serviço deve estar previamente cadastrada.

6. Pós-Condições:
A ordem de serviço / solicitação de orçamento é visualizada.

7. Cenário Principal:
Paulo Faria liga para a empresa e a recepcionista Vanessa o atende. Ele deseja saber o andamento de sua Ordem de Serviço. Vanessa acessa a opção Pesquisar OS, no menu OS – Ordens de Serviço, seleciona o status "Todas", seleciona o registro da OS e clica em visualizar. Informa Paulo que sua OS está fechada e que pode vir retirar o equipamento.

Emitir Relatório:

1.Descrição Sumária:
Este caso de uso permite a emissão de relatórios de Ordens de Serviço, por data, número ou cliente.

2. Atores:
Recepcionista.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona o status da Ordem serviço, usa a opção visualizar e confirma a impressão.

1. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona a aba Específicos, seleciona a condição de busca, usa a opção visualizar e confirma a impressão.

3.2 Fluxo secundário:

1. Ator ao usar a opção visualizar, não seleciona a condição de busca. O sistema avisa de deve ser seleciona uma condição.

5. Pré-Condições:

A Ordem de Serviço deve estar previamente cadastrada.

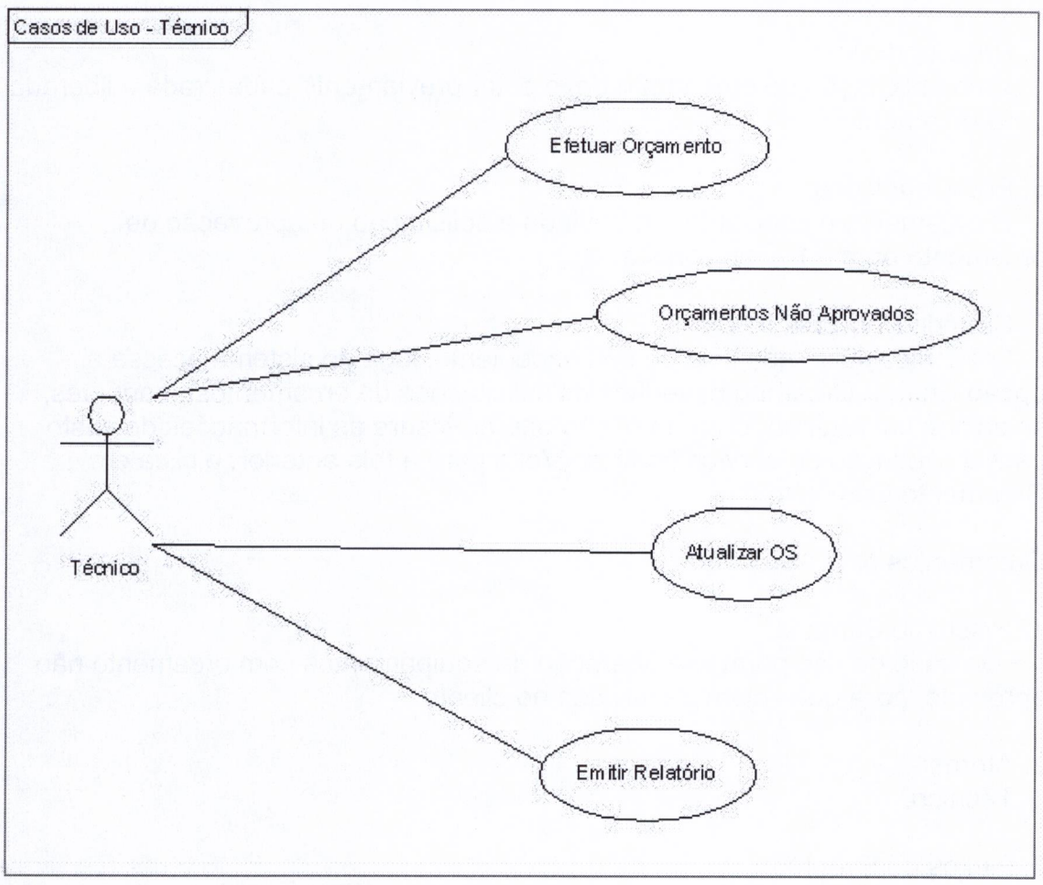
6. Pós-Condições:

São visualizadas e impressas as Ordens de Serviço.

7. Cenário Principal:

A recepcionista Vanessa, deseja visualizar e imprimir uma relação das Ordens de Serviço com status “Em Andamento”, para tal, acessa o menu Ordens de Serviço, seleciona “Em Andamento”, clica em Visualizar e clica no botão imprimir.

7.1.3 Caso de uso – Técnico:



Efetuar Orçamento:

1.Descrição Sumária:
Este caso de uso permite ao Técnico, verificar os custos da execução dos serviços de uma solicitação de orçamento para repassar à Recepcionista, a qual passará o valor ao cliente, para aprovação.

2. Atores:
Técnico.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Efetuar Orçamento, no menu OS – Ordens de Serviço, seleciona o registro da solicitação de orçamento, usa a opção alterar, seleciona o equipamento, inclui a descrição do problema e custo do serviço / peça e confirma, retornando à tela anterior, onde seleciona a opção Orçamento Completo.

3.2 Fluxo secundário:

1. Ator não seleciona um registro e usa a opção Alterar / Orçamento Completo. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:

Uma solicitação de orçamento deve estar previamente cadastrada e liberada para orçamento.

6. Pós-Condições:

O orçamento é concluído e é enviada a solicitação de aprovação de orçamento para a Recepcionista.

7. Cenário Principal:

O Técnico Júlio, no início de seu expediente, loga no sistema acessa a opção Efetuar Orçamento, verifica as solicitações de orçamentos pendentes, seleciona um registro, clica na opção alterar, insere as informações de custo para a execução do serviço finaliza. Volta para a tela anterior, e clica em Orçamento Completo.

Orçamentos Não Aprovados:

1.Descrição Sumária:

Este caso de uso permite a liberação de equipamentos com orçamento não aprovado, para que sejam devolvidos ao cliente.

2. Atores:

Técnico.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator seleciona opção Orçamentos Não Aprovados, no menu OS – Ordens de Serviço, seleciona o registro de orçamento e usa a opção Liberar Equipamento.

3.2 Fluxo secundário:

1. Ator não seleciona um registro e usa a opção Liberar Equipamento. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:

O orçamento deve ter sido efetivado e não aprovado.

6. Pós-Condições:

O orçamento é cancelado e o equipamento é liberado para devolução.

7. Cenário Principal:

O Técnico Júlio, deseja consultar se há algum equipamento que deva ser devolvido ao cliente, para tal, seleciona a opção Orçamentos Não Aprovados, no menu OS – Ordens de Serviço e clica em “Liberar Equipamento”. Após,

encaminha o equipamento à Recepcionista, que se encarregará de contactar o cliente para efetivar a devolução.

Atualizar OS:

1.Descrição Sumária:
Este caso de uso permite ao Técnico, o status da execução dos serviços numa Ordem de Serviço e atualizá-los, conforme o andamento do processo, assim como liberar uma ordem de serviço já concluída, para seu devido fechamento.

2. Atores:
Técnico.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Atualizar OS, no menu OS – Ordens de Serviço, seleciona o registro da OS, usa a opção alterar, inclui informações sobre o problema, adiciona / remove peças (conforme necessário), altera o status da OS e confirma.

3.2 Fluxo secundário:
1. Ator não seleciona um registro e usa a opção alterar. O sistema avisa que um registro deve ser selecionado.

5. Pré-Condições:
A solicitação de orçamento deve ter sido aprovada pelo cliente.

6. Pós-Condições:
O status da Ordem de Serviço é alterado.

7. Cenário Principal:
O Técnico Júlio, acessa a opção Atualizar Ordem de Serviço, verifica que há uma Ordem de Serviço Aprovada, seleciona o registro, clica em "Alterar", inclui informações sobre o serviço a ser executado, seleciona o status "Em Andamento" e confirma.

Emitir Relatório:

1.Descrição Sumária:
Este caso de uso permite a emissão de relatórios de Ordens de Serviço, por data, número ou cliente.

2. Atores:
Técnico.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona o status da Ordem serviço, usa a opção visualizar e confirma a impressão.

1. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona a aba Específicos, seleciona a condição de busca, usa a opção visualizar e confirma a impressão.

3.2 Fluxo secundário:

1. Ator ao usar a opção visualizar, não seleciona a condição de busca. O sistema avisa de deve ser seleciona uma condição.

5. Pré-Condições:

A Ordem de Serviço deve estar previamente cadastrada.

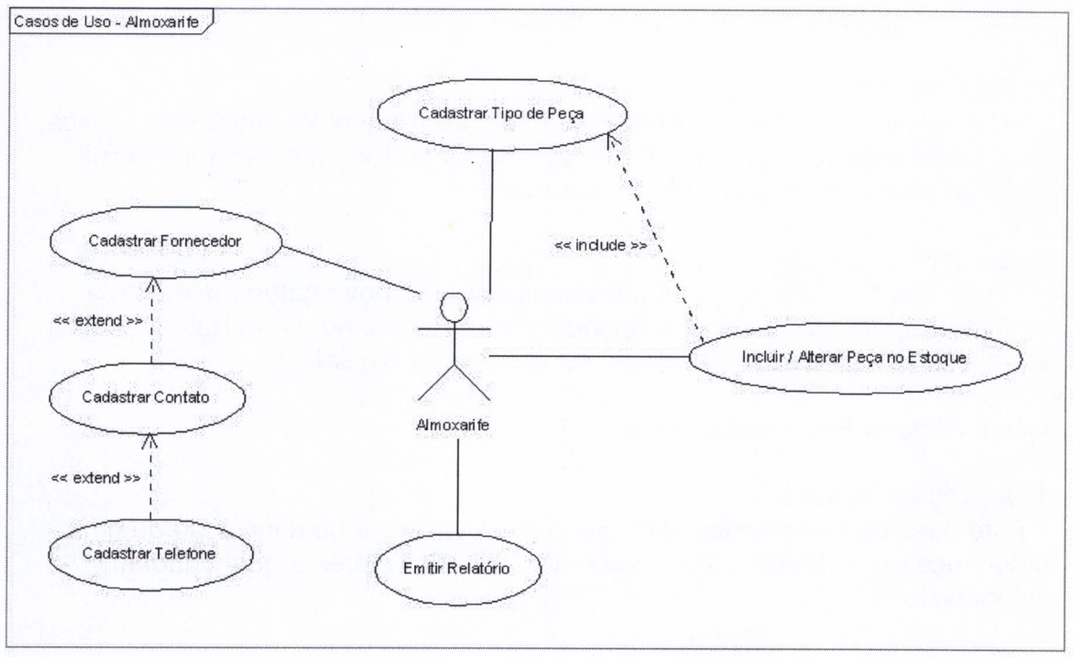
6. Pós-Condições:

São visualizadas e impressas as Ordens de Serviço.

7. Cenário Principal:

O Técnico Júlio, deseja visualizar e imprimir uma relação das Ordens de Serviço com status “Aguardando Peça”, para tal, acessa o menu Ordens de Serviço, seleciona “Aguardando Peça”, clica em Visualizar e clica no botão imprimir.

7.1.4 Caso de uso – Almoxarife:



Cadastrar Tipo de Peça:

1.Descrição Sumária:
Este caso de uso permite ao almoxarife cadastrar as categorias de peças que sejam passíveis de solicitação. Também permite a alteração de categorias já existentes.

2. Atores:
Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Tipo de Peças, no menu Estoque, usa a opção Incluir, insere a Descrição do Tipo e confirma.
2. Ator seleciona a opção Tipo de Peças, no menu Estoque, seleciona o registro do tipo de peça, usa a opção alterar, insere a nova Descrição do Tipo e confirma.

3.2 Fluxo secundário:
1. Caso o ator não selecione um registro e use a opção Incluir / Alterar, o sistema avisa que deve ser selecionado um registro.
2. Caso o ator não digite uma descrição do tipo ao Incluir / Alterar, o sistema informa que deve ser preenchido o campo Descrição do Tipo.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
Um categoria de tipo de peça é inclusa no sistema.

7. Cenário Principal:
O Almoxarife Alexandre, necessita cadastrar uma nova categoria de peça, para tal, acessa a opção Tipo de Peças, no menu Estoque, clica em incluir, digita a descrição do tipo e clica em salvar.

8. Cenário Secundário:
O Almoxarife Alexandre, necessita alterar uma nova categoria de peça cadastrada, para tal, acessa a opção Tipo de Peças, no menu Estoque, clica em alterar, digita a nova descrição do tipo e clica em salvar.

Incluir / Alterar Peça no Estoque:

1.Descrição Sumária:
Este caso de uso permite com que o almoxarife efetue a inclusão de uma nova peça em estoque, assim como alterar seus valores e quantidades disponíveis.

2. Atores:
Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Peça, no menu Estoque, usa a opção incluir, entra com a descrição da peça, quantidade, valor unitário, escolhe o tipo da peça e confirma.
2. Ator seleciona a opção Peça, no menu Estoque, escolhe um registro de peça, usa a opção alterar, entra com os novos valores e confirma.

3.2 Fluxo secundário:
1. Ator não seleciona o registro e usa a opção alterar. O sistema informa que um registro deve ser escolhido.
2. Ator não preenche o campo descrição da peça e/ou não seleciona o tipo da peça. O sistema informa que os campos devem ser preenchidos.

5. Pré-Condições:
O tipo da peça deve estar previamente cadastrado.
6. Pós-Condições:
A peça é inserida ao sistema.

7. Cenário Principal:
O almoxarife Alexandre necessita incluir uma nova peça no cadastro. Para tal, seleciona a opção de menu “Peça”, clica em “Incluir”, entra com as descrições da peça e confirma com o botão salvar.

8. Cenário Secundário:
O almoxarife Alexandre necessita alterar uma peça no cadastro. Para tal, seleciona a opção de menu "Peça", seleciona o registro da peça, clica em "Alterar", entra com as novas descrições da peça e confirma com o botão salvar.

Cadastrar Cliente:

1.Descrição Sumária:
Este caso de uso permite o cadastro de fornecedores no sistema. Também permite a alteração de cadastro, inclusão de contatos e observações.

2. Atores:
Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção fornecedores, no menu Estoque, usa a opção incluir, preenche os campos de cadastro e confirma.
2. Ator seleciona a opção fornecedores, no menu Estoque, seleciona um registro de Fornecedor, usa a opção alterar, altera os campos necessários e confirma.
3. Ator, dentro de um registro de Fornecedor, acessa a aba Observações, insere uma observação e confirma.

3.2 Fluxo secundário:
1. Caso o ator não preencha corretamente o campo CPF ou CNPJ, o sistema informa que o CPF / CNPJ está incorreto.
2. Caso o ator não selecione um registro de fornecedor ao usar a opção alterar, o sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
Um novo fornecedor é adicionado ao sistema.

7. Cenário Principal:
Alexandre , acessa a opção "Fornecedores", clica em "Incluir" adiciona os dados do fornecedor "PC Plus", escolhe a opção "Pessoa Jurídica", entra com o CNPJ / IE e demais dados e confirma, através do botão salvar.

8. Cenário Secundário:
Alexandre , acessa a opção "Fornecedores", clica em "Incluir" adiciona os dados do fornecedor "PC Plus", escolhe a opção "Pessoa Jurídica", entra com o CNPJ e ao acessar o campo IE, o sistema informa "CNPJ Inválido", então Alexandre verifica o número e o altera, entra com a IE e demais dados e confirma, através do botão salvar.

Cadastrar Contato:

1.Descrição Sumária:
Este caso de uso permite a inclusão e / ou alteração dos contatos de fornecedores previamente cadastrados no sistema.

2. Atores:
Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator, dentro de um registro de Fornecedor, acessa a aba Contatos, usa a opção incluir, insere os dados do contato e confirma.
2. Ator, dentro de um registro de Fornecedor, acessa a aba Contatos, seleciona um registro de contato e usa a opção alterar dados, altera os campos necessários e confirma.

3.2 Fluxo secundário:
1. Caso o ator não preencha o campo nome e confirme a inclusão do contato, o sistema informa que o campo deve ser preenchido.

5. Pré-Condições:
O fornecedor deve estar pré-cadastrado.

6. Pós-Condições:
Um novo contato é adicionado ao sistema.

7. Cenário Principal:
Alexandre acessa o registro do cliente “PC Plus”, seleciona a aba “Contatos”, clica com o botão direito e seleciona a opção “Incluir”, adiciona os dados do contato e confirma com a opção “Salvar”.

8. Cenário Secundário:
Alexandre acessa o registro do cliente “PC Plus”, seleciona a aba “Contatos”, clica com o botão direito e seleciona a opção “Alterar”, altera o campo “e-mail” e confirma com a opção “Salvar”.

Cadastrar Telefone:

1.Descrição Sumária:
Este caso de uso permite que sejam cadastrados e / ou alterados os telefones, de contatos de um fornecedor pré-cadastrado no sistema.

2. Atores:
Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator, dentro de um registro de Fornecedor, acessa a aba Contatos, seleciona um registro de contato, usa a opção alterar, digita o número e o tipo do telefone e confirma.

3.2 Fluxo secundário:

1. Caso o ator não selecione o tipo do telefone, o sistema informa que deve ser escolhido o tipo do telefone.

5. Pré-Condições:

O fornecedor e o contato devem estar previamente cadastrados.

6. Pós-Condições:

O telefone do contato é adicionado ao sistema.

7. Cenário Principal:

Alexandre acessa o menu fornecedores, seleciona o registro " PC Plus", clica em "Alterar", seleciona a aba "Contatos, escolhe o contato "César", clica com o botão direito e seleciona "Alterar", clica em "Alterar Telefones", digita o número do telefone, escolhe a opção "Celular" e confirma através do botão salvar.

Emitir Relatório:

1.Descrição Sumária:

Este caso de uso permite a emissão de relatórios de Peças em Estoque, por status, ordem ou valor.

2. Atores:

Almoxarife.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção Estoque, no menu Relatórios Gerais, seleciona opção de consulta, usa a opção visualizar e confirma a impressão.

3.2 Fluxo Secundário:

1. Ator não seleciona as condições de busca e seleciona visualizar. O sistema informa que uma condição deve ser selecionada.

5. Pré-Condições:

A Ordem de Serviço deve estar previamente cadastrada.

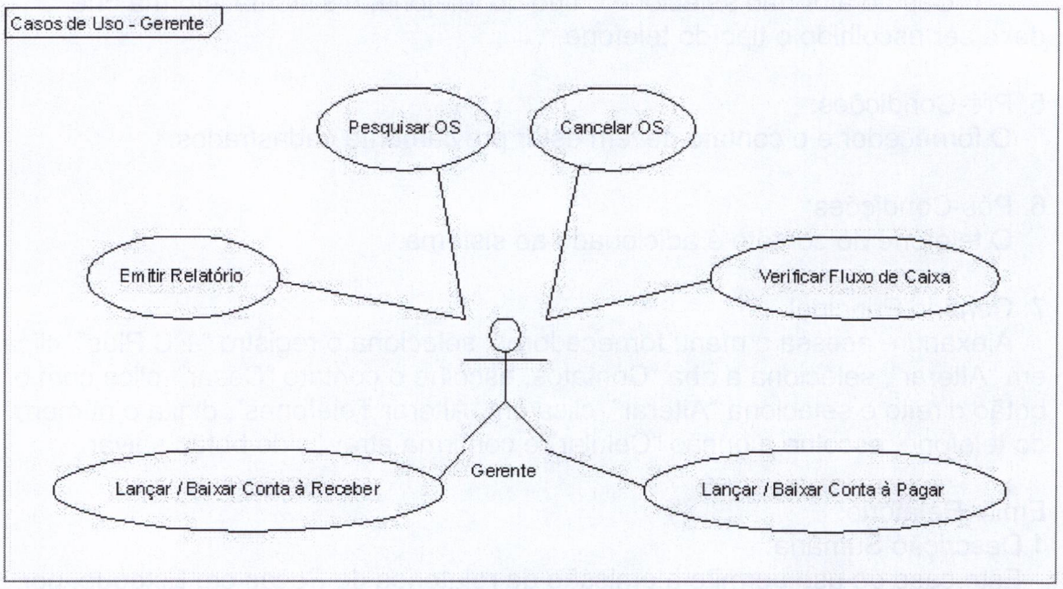
6. Pós-Condições:

É mostrado e impresso o relatório de Peças.

7. Cenário Principal:

O Almoxarife Alexander, deseja visualizar e imprimir uma relação das Peças com falta em estoque, para tal, acessa a opção Estoque, no menu Relatórios, seleciona "Produtos Sem Estoque", clica em Visualizar e clica no botão imprimir.

7.1.5 Caso de uso – Gerente:



Pesquisar OS:

1.Descrição Sumária:
Este caso de uso permite que sejam pesquisadas Ordens de Serviço pelo seu Status, que podem ser: Todas, canceladas, orçamentos, aprovadas, em andamento, aguardando peça, liberadas, fechadas, orçar, orçado, recusada e devolver.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona a opção Pesquisar OS, no menu OS – Ordens de Serviço, seleciona o status de visualização, seleciona o registro da OS e usa a opção Visualizar.

3.2 Fluxo secundário:
1. Ator não seleciona um registro de OS e usa a opção visualizar. O sistema informa que um registro dever ser selecionado.

5. Pré-Condições:

Uma solicitação de orçamento ou uma Ordem de Serviço deve estar previamente cadastrada.

6. Pós-Condições:
A ordem de serviço / solicitação de orçamento é visualizada.

7. Cenário Principal:
O Gerente Cláudio, gostaria de verificar quantas Ordens de Serviços estão Aguardando Peça, para tal, acessa o menu OS – Ordens de Serviço, seleciona o status "Aguardando Peça" e verifica o número de OS, podendo, também, visualizá-las, através do botão "Visualizar".

Cancelar OS:

1.Descrição Sumária:
Este caso de uso permite ao gerente cancelar qualquer ordem de serviço, desde que já não cancelada.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator seleciona o item "Cancelar OS", no menu OS – Ordens de Serviço, escolhe o status de listagem, escolhe o registro da OS e usa a opção cancelar.

3.2 Fluxo secundário:
1. Ator não seleciona um registro de OS e utiliza a opção cancelar. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
A Ordem de Serviço deve estar cadastrada no sistema.

6. Pós-Condições:
A Ordem de Serviço é cancelada.

7. Cenário Principal:
O gerente Cláudio, recebeu a informação de que um cliente, apesar de ter aprovado o orçamento, não deseja mais executar o serviço. Para tal, acessa o menu Cancelar OS, localiza o registro da OS e clica em "Cancelar OS" e confirma através do botão "Cancelar".

Lançar / Baixar Contas à Receber:

1.Descrição Sumária:
Este caso de uso permite que sejam inclusos lançamentos de contas à receber, incluindo operações de alteração, exclusão e baixa.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção contas à receber, no menu caixa, usa a opção incluir, entra com os dados referentes à conta à pagar e confirma.

2. Ator acessa a opção contas à receber, no menu caixa, seleciona o registro da conta, usa a opção alterar, altera os dados referentes à conta à pagar e confirma.

3. Ator acessa a opção contas à receber, no menu caixa, seleciona o registro da conta, usa a opção excluir e confirma.

4. Ator acessa a opção contas à receber, no menu caixa, usa a opção baixar conta, informa os dados da baixa e confirma.

3.2 Fluxo secundário:

1. Ator usa a opção incluir / alterar e não preenche os campos data emissão, data vencimento, tipo de pagamento, tipo do movimento, cliente ou histórico. Sistema avisa que os campos devem ser preenchidos.

2. Ator usa a opção alterar / excluir / baixar conta e não seleciona um registro. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
A conta a receber é incluida / alterada / excluída / baixada do sistema.

7. Cenário Principal:
O gerente Cláudio deseja fazer um lançamento de uma conta à receber. Para tal, seleciona a opção “contas à pagar”, no menu “caixa”, clica em incluir, insere os dados referentes à conta e confirma o lançamento clicando em “salvar” e depois em “sim”.

8. Cenário Secundário:
O gerente Cláudio deseja fazer uma baixa de uma conta à receber. Para tal, seleciona a opção “contas à pagar”, no menu “caixa”, seleciona o registro da conta, clica em “baixar conta”, clica em “sim”, insere os dados referentes a baixa e confirma clicando em “confirmar”.

Lançar / Baixar Contas à Pagar:

1.Descrição Sumária:
Este caso de uso permite que sejam inclusos lançamentos de contas à pagar, incluindo operações de alteração, exclusão e baixa.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção contas à pagar, no menu caixa, usa a opção incluir, entra com os dados referentes à conta à pagar e confirma.
2. Ator acessa a opção contas à pagar, no menu caixa, seleciona o registro da conta, usa a opção alterar, altera os dados referentes à conta à pagar e confirma.
3. Ator acessa a opção contas à pagar, no menu caixa, seleciona o registro da conta, usa a opção excluir e confirma.
4. Ator acessa a opção contas à pagar, no menu caixa, usa a opção baixar conta, informa os dados da baixa e confirma.

3.2 Fluxo secundário:

1. Ator usa a opção incluir / alterar e não preenche os campos data emissão, data vencimento, tipo de pagamento, tipo do movimento, fornecedor ou histórico. Sistema avisa que os campos devem ser preenchidos.
2. Ator usa a opção alterar / excluir / baixar conta e não seleciona um registro. O sistema informa que um registro deve ser selecionado.

5. Pré-Condições:
Não se aplica.

6. Pós-Condições:
A conta a pagar é incluida / alterada / excluída / baixada do sistema.

7. Cenário Principal:
O gerente Cláudio deseja fazer um lançamento de uma conta à pagar. Para tal, seleciona a opção “contas à pagar”, no menu “caixa”, clica em incluir, insere os dados referentes à conta e confirma o lançamento clicando em “salvar” e depois em “sim”.

8. Cenário Secundário:
O gerente Cláudio deseja fazer uma baixa de uma conta à pagar. Para tal, seleciona a opção “contas à pagar”, no menu “caixa”, seleciona o registro da conta, clica em “baixar conta”, clica em “sim”, insere os dados referentes a baixa e confirma clicando em “confirmar”.

Verificar Fluxo de Caixa:

1.Descrição Sumária:
Este caso de uso permite com que sejam verificadas informações quanto ao fluxo de caixa, tais como contas à pagar e contas à receber.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:

1. Ator acessa a opção prev. Contas pagar-receber, seleciona a data para pesquisa e usa a opção filtrar.

5. Pré-Condições:
As contas a pagar e receber devem estar previamente cadastradas no sistema.

6. Pós-Condições:
São mostradas as contas a pagar / receber.

7. Cenário Principal:
O gerente Cláudio seleciona a opção "Prev. Contas pagar-receber", no menu "Caixa", seleciona as datas de início e fim e clica em "Filtrar".

Emitir Relatório:

1.Descrição Sumária:
Este caso de uso permite a emissão de relatórios de Ordens de Serviço, fluxo de Caixa e controle de Estoque.

2. Atores:
Gerente.

3. Fluxos de Eventos:

3.1 Fluxo principal:
1. Ator acessa a opção Estoque, no menu Relatórios Gerais, seleciona opção de consulta, usa a opção visualizar e confirma a impressão.
2. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona o status da Ordem serviço, usa a opção visualizar e confirma a impressão.
3. Ator acessa a opção Ordem de Serviço, no menu Relatórios Gerais, seleciona a aba Específicos, seleciona a condição de busca, usa a opção visualizar e confirma a impressão.
4. Ator acessa a opção Caixa, no menu Relatórios Gerais, seleciona a opção fluxo de caixa, insere os dados de busca, usa a opção visualizar e confirma a impressão.
5. Ator acessa a opção Caixa, no menu Relatórios Gerais, seleciona a opção Despesas Gerais, insere os dados de busca, usa a opção visualizar e confirma a impressão.
6. Ator acessa a opção Caixa, no menu Relatórios Gerais, seleciona a opção receitas gerais, insere os dados de busca, usa a opção visualizar e confirma a impressão.

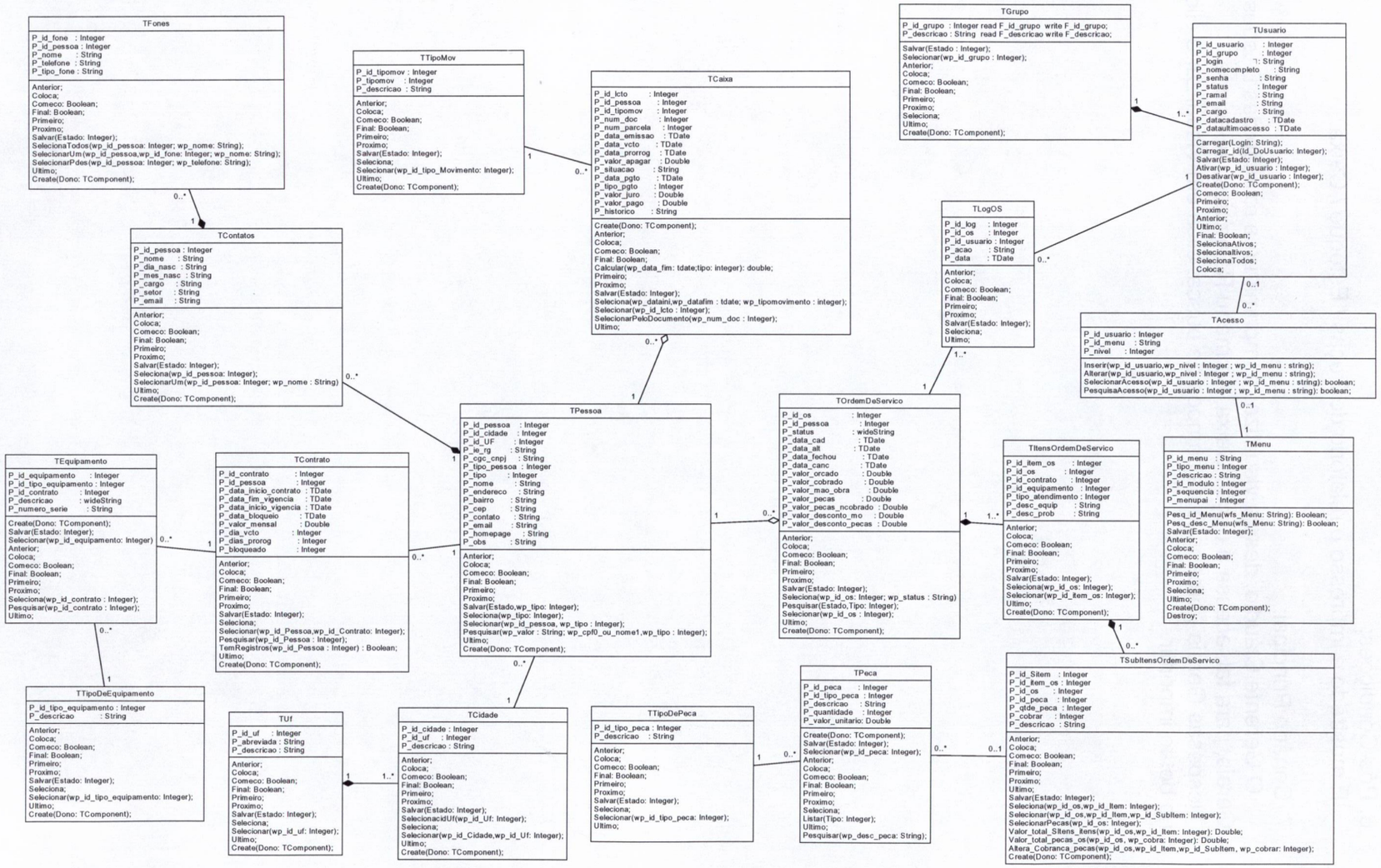
3.2 Fluxo Secundário:
1. Ator não seleciona as condições de busca e seleciona visualizar. O sistema informa que uma condição deve ser selecionada.

5. Pré-Condições:
A Ordem de Serviço deve estar previamente cadastrada.

6. Pós-Condições:
É mostrado e impresso o relatório de Peças / Estoque / Caixa.

7. Cenário Principal:
O Gerente Cláudio, deseja visualizar e imprimir uma relação das Despesas Gerais, para tal, acessa a opção Caixa, no menu Relatórios, seleciona "Despesas Gerais", preenche os campos de busca, clica em Visualizar e clica no botão imprimir.

8 DIAGRAMA DE CLASSES



9 DIAGRAMAS DE SEQUÊNCIA

Diagrama de Sequência - Cadastrar Cidades

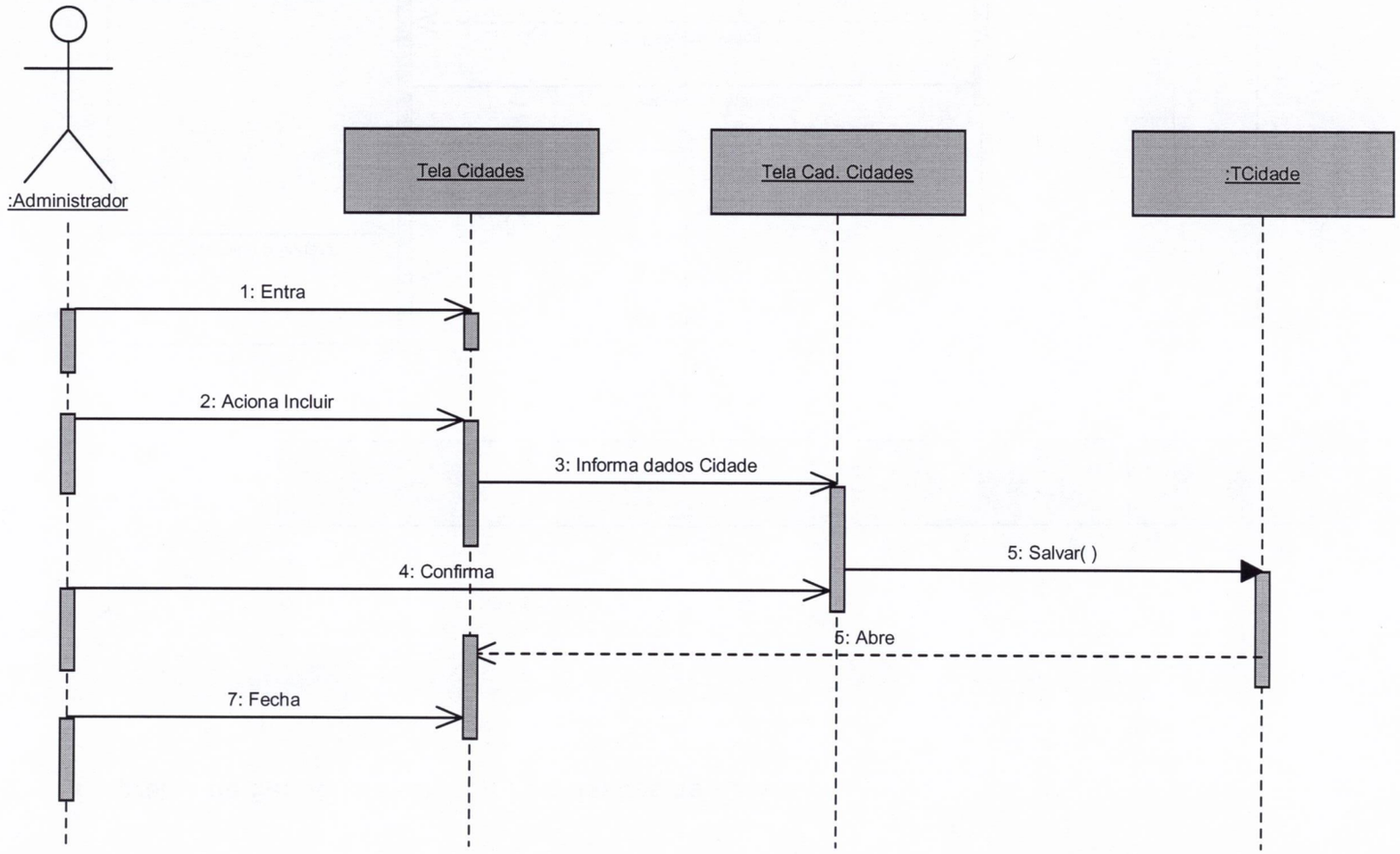


Diagrama de Sequência - Definir Permissões de Acesso

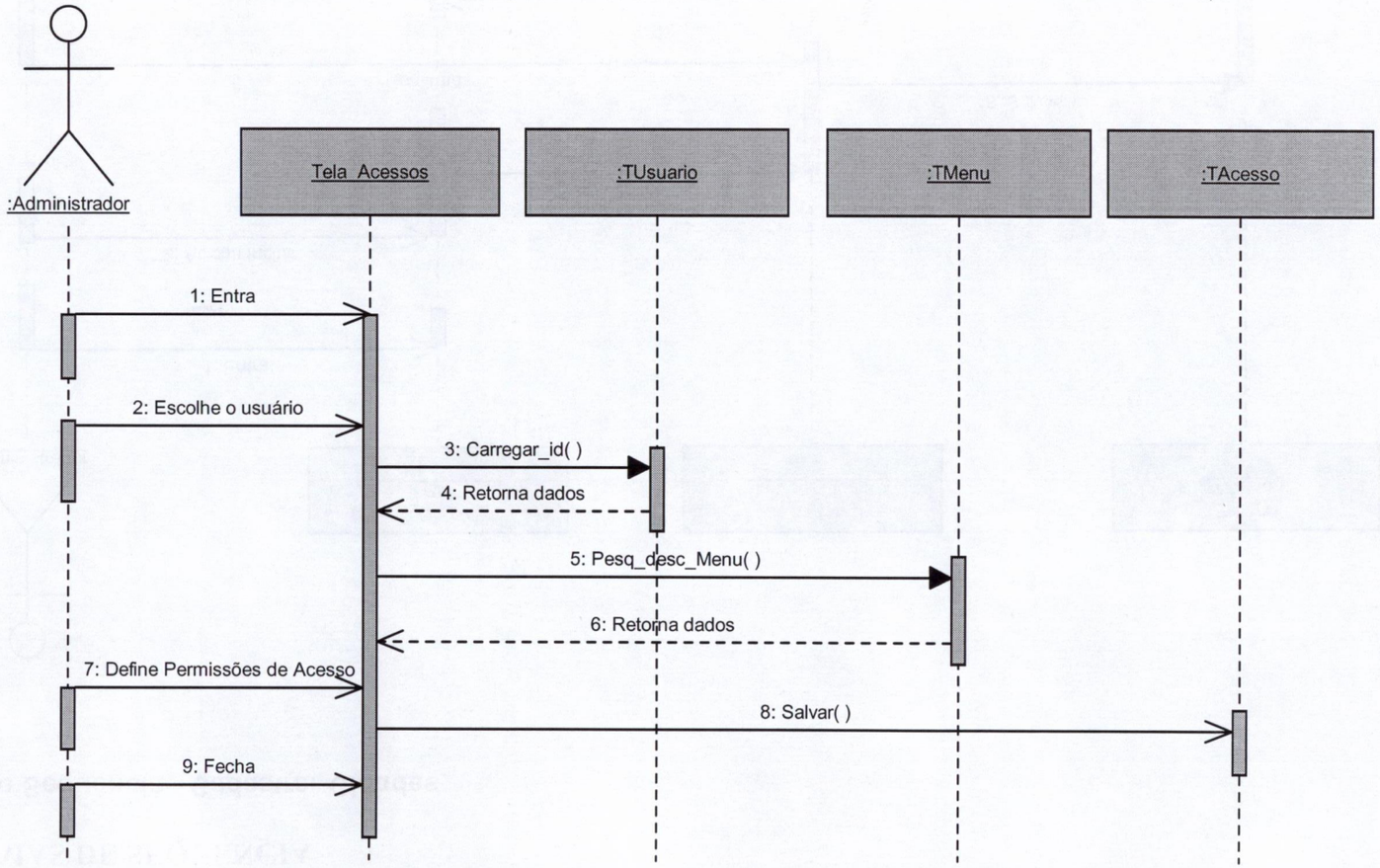


Diagrama de Sequência - Cadastrar Usuário

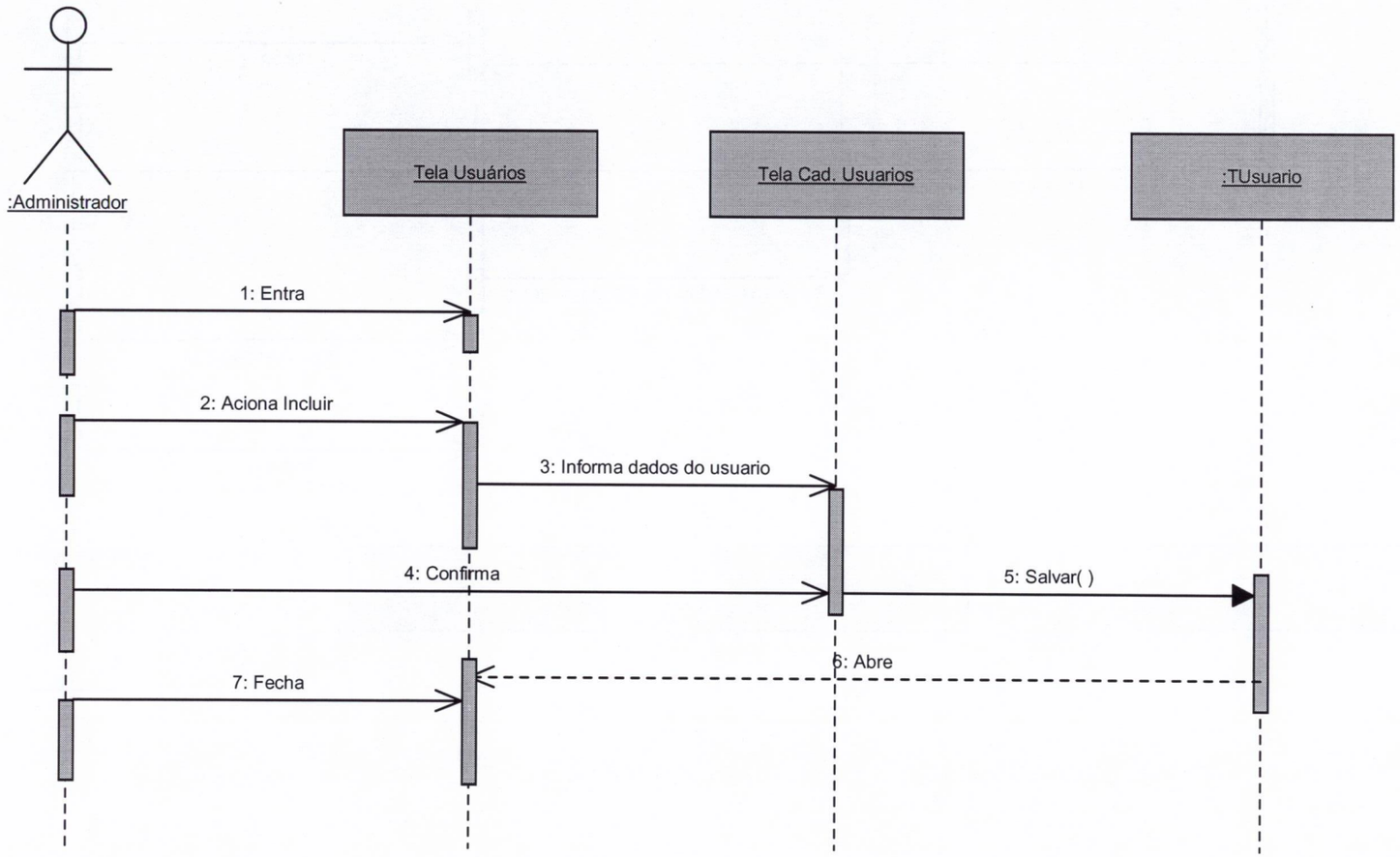


Diagrama de Sequência - Cadastrar Grupo de Usuários

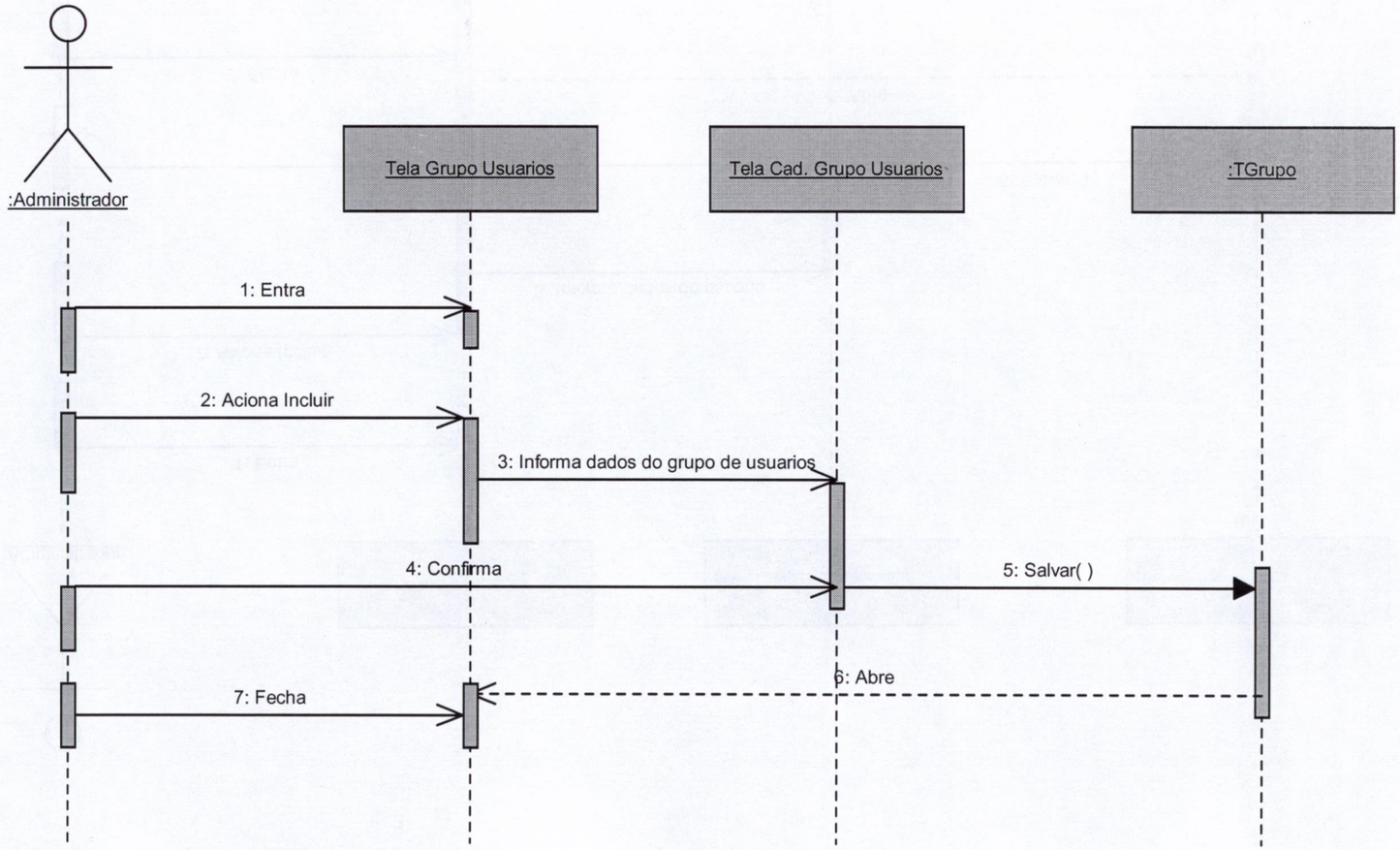


Diagrama de Sequência - Cadastrar Estados

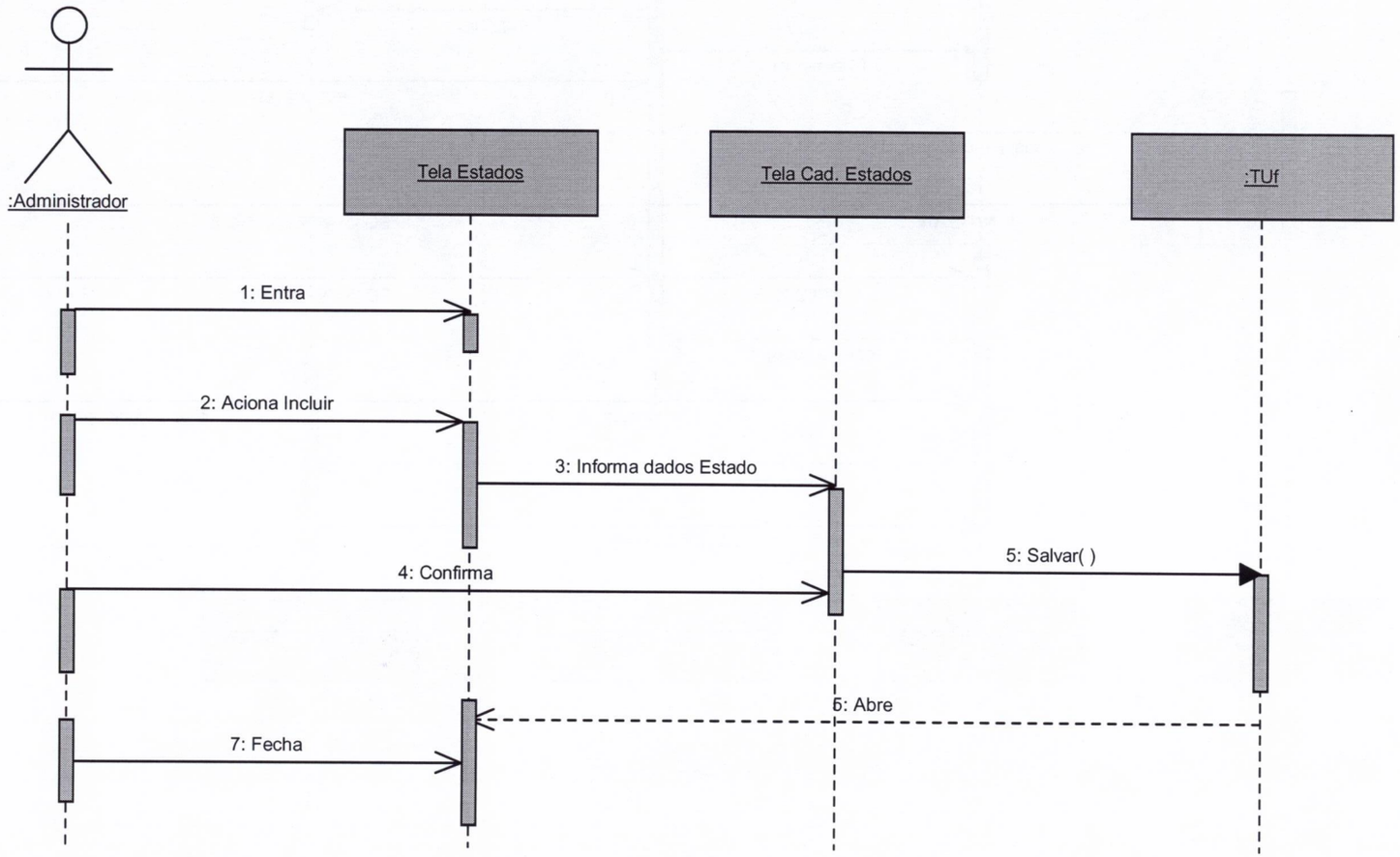


Diagrama de Sequência - Atualizar OS

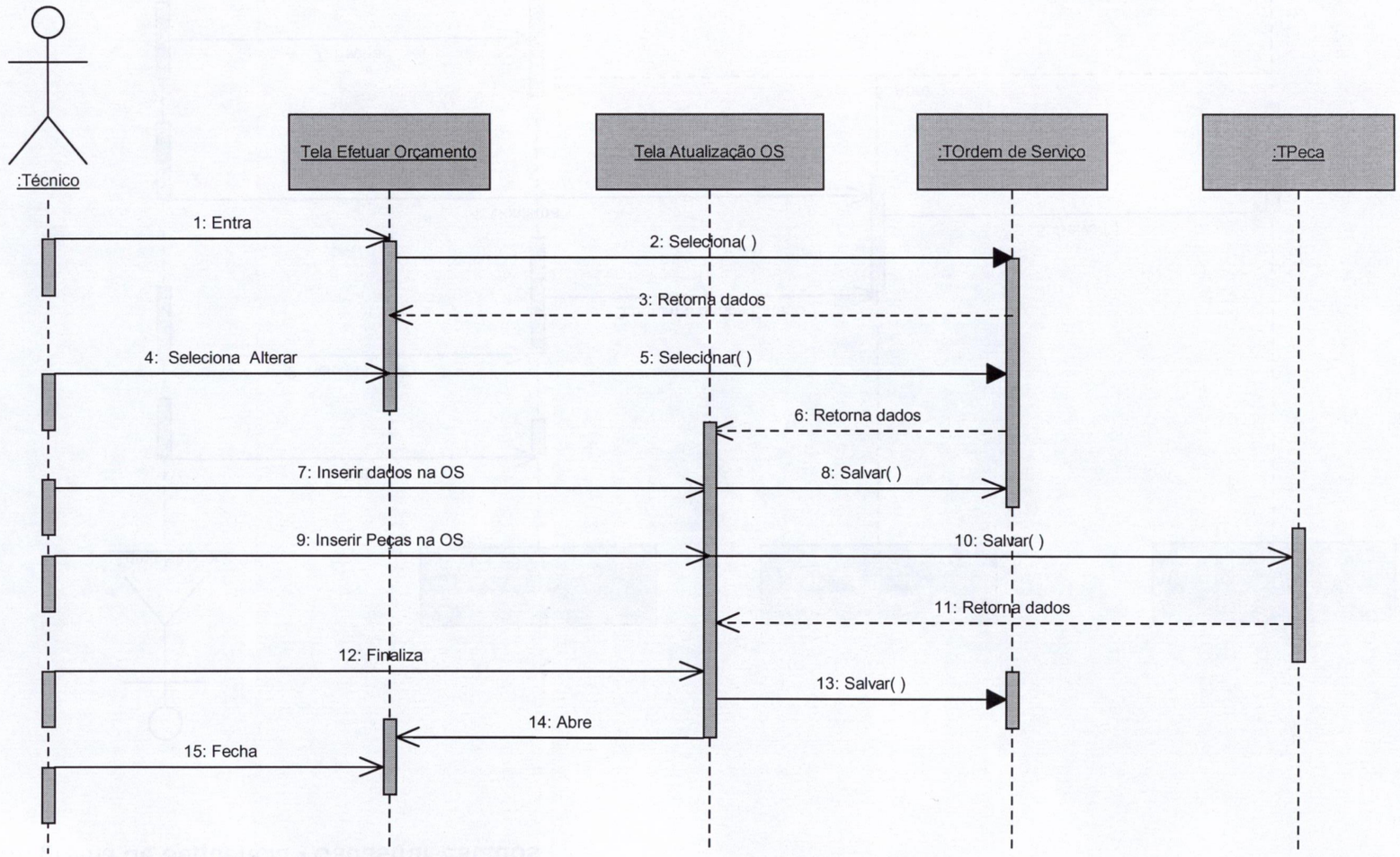


Diagrama de Sequência - Cadastrar tipo de peça

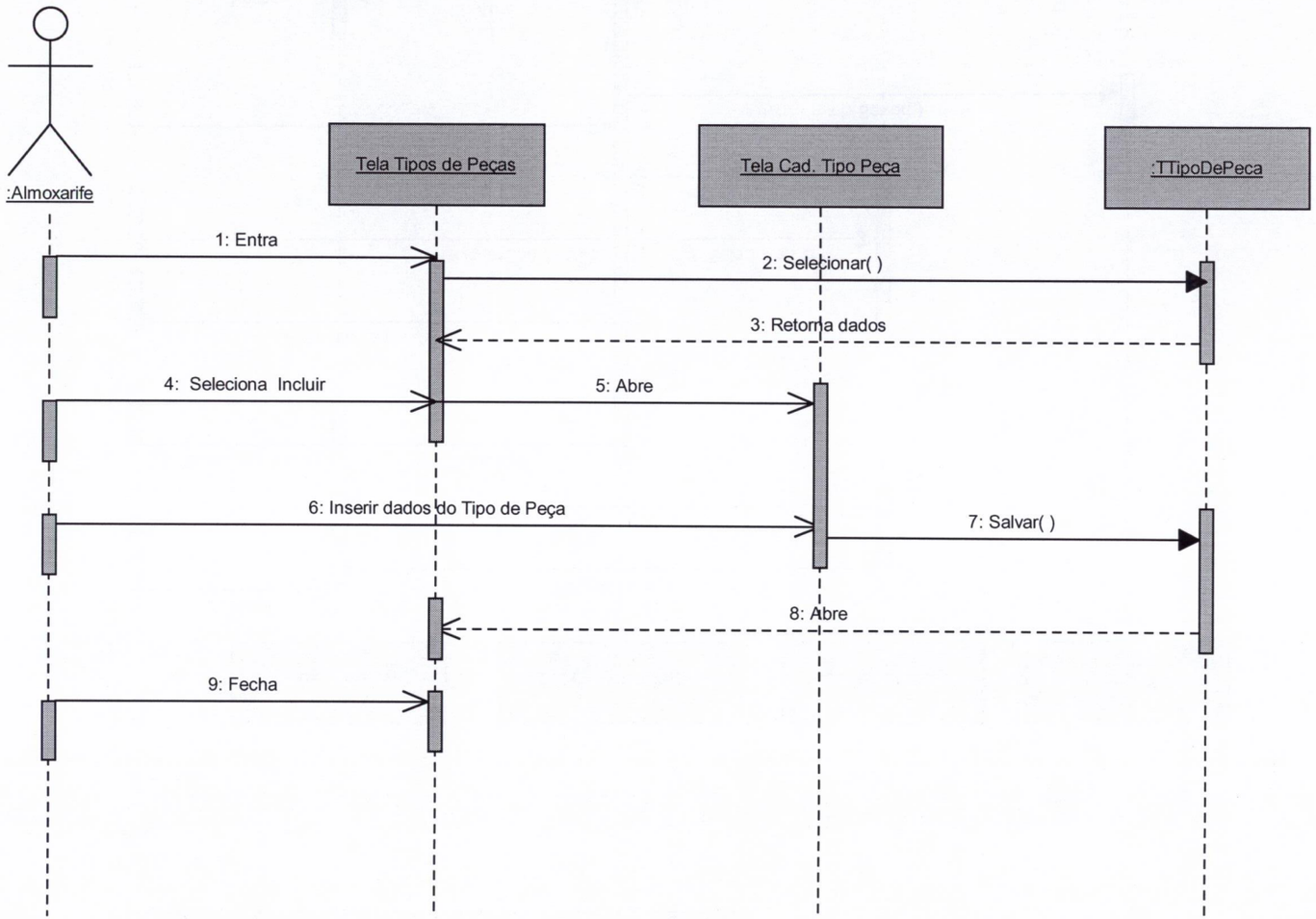


Diagrama de Sequência - Incluir Peça no Estoque

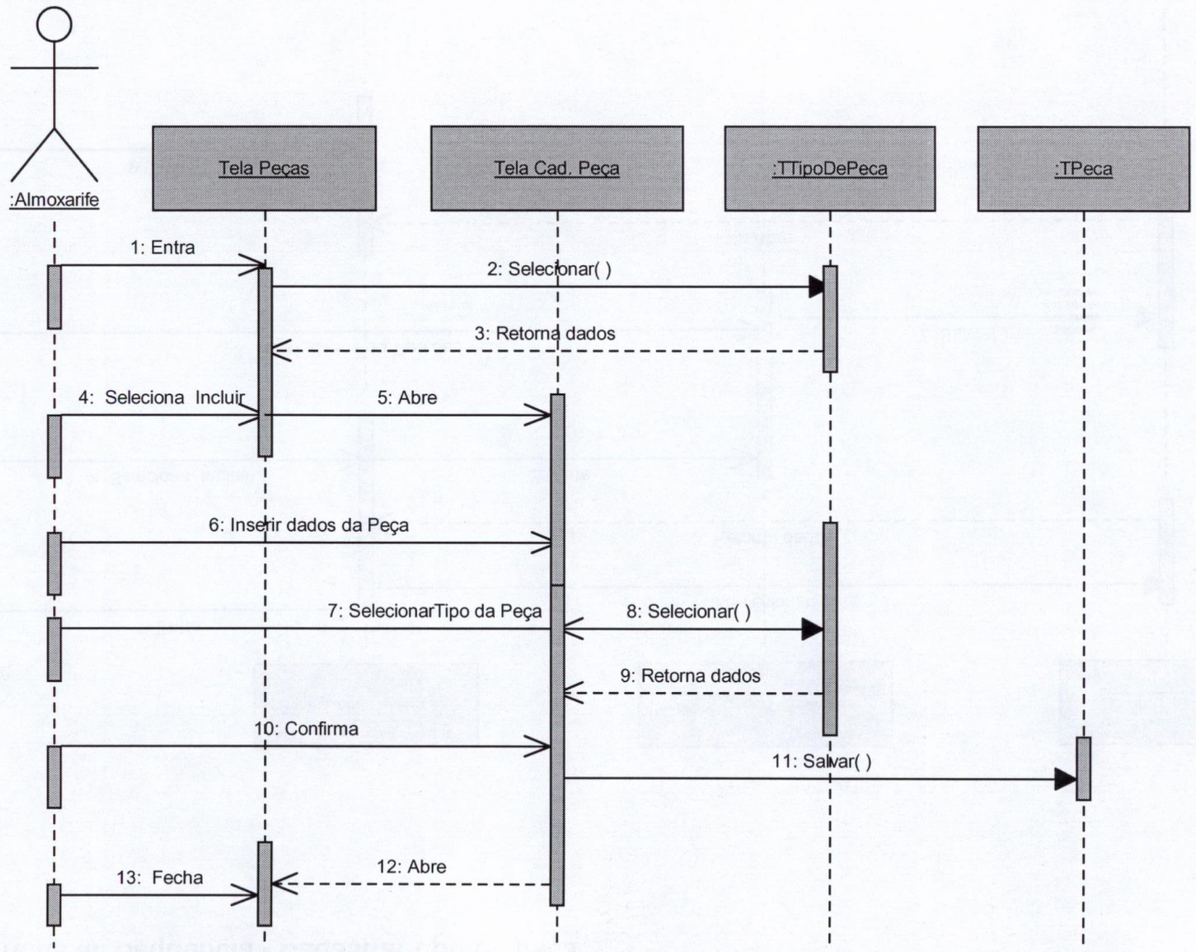


Diagrama de Sequência - Alterar Peça em Estoque

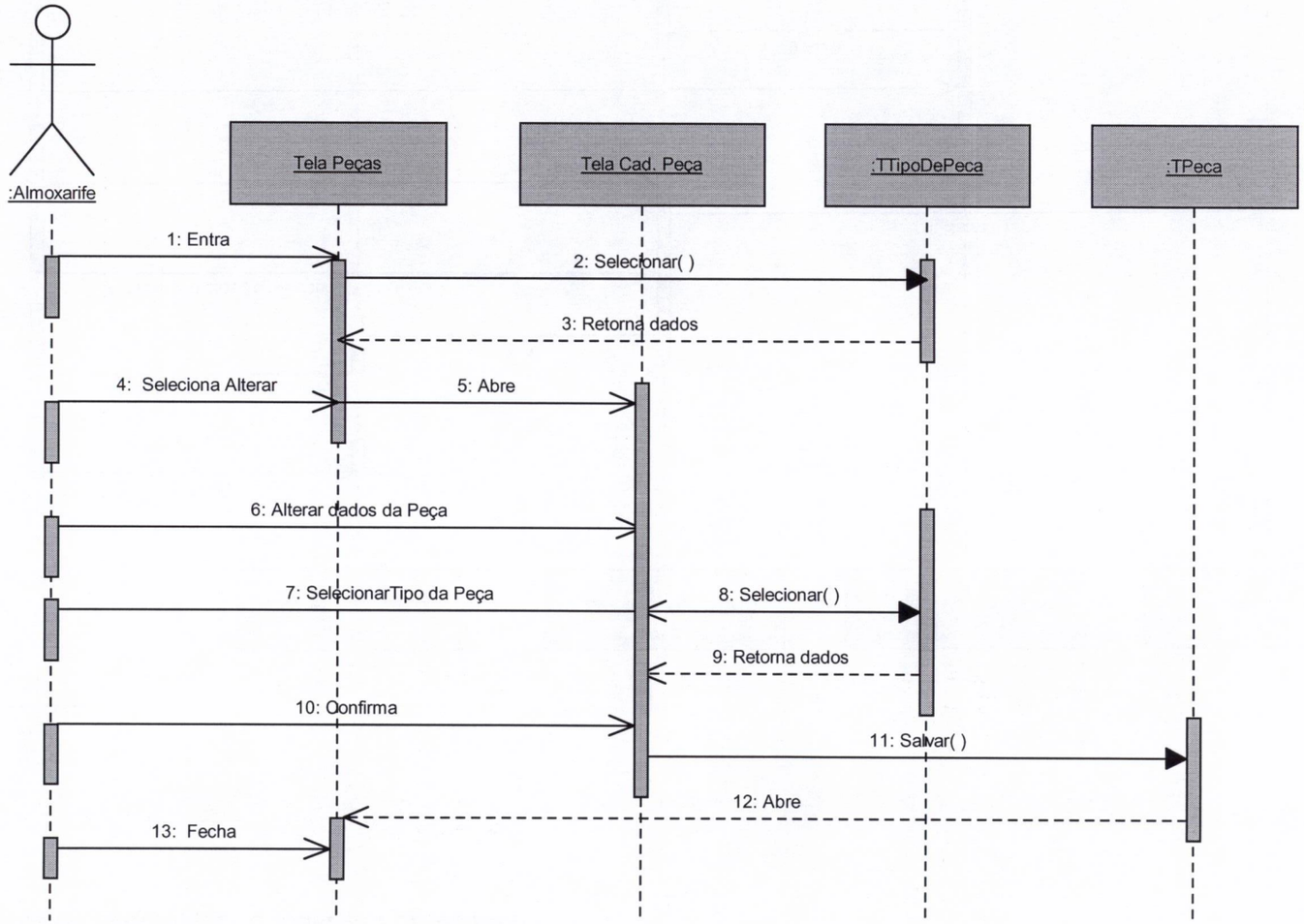


Diagrama de Sequência - Cadastrar Fornecedor

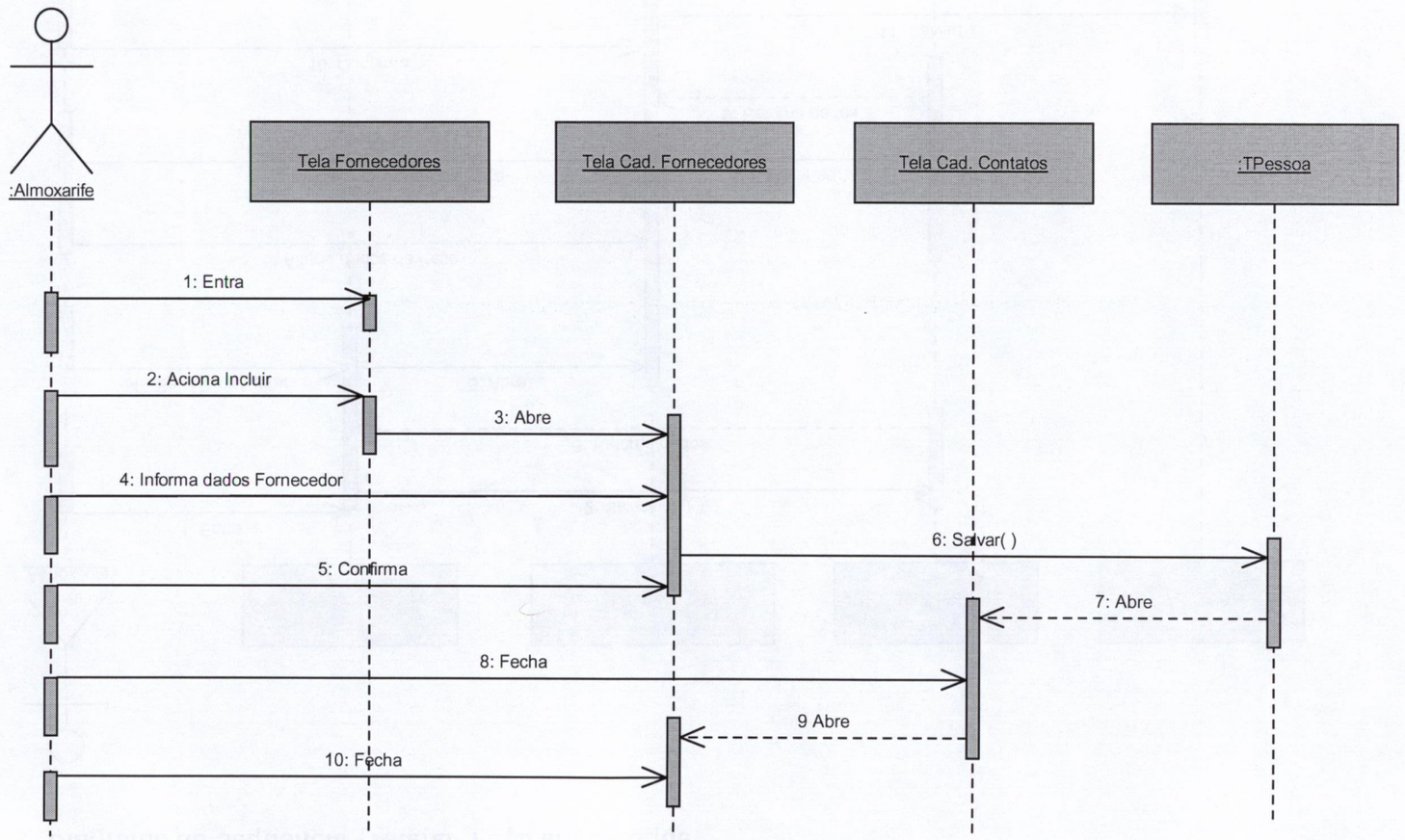


Diagrama de Sequência - Cadastrar Contato

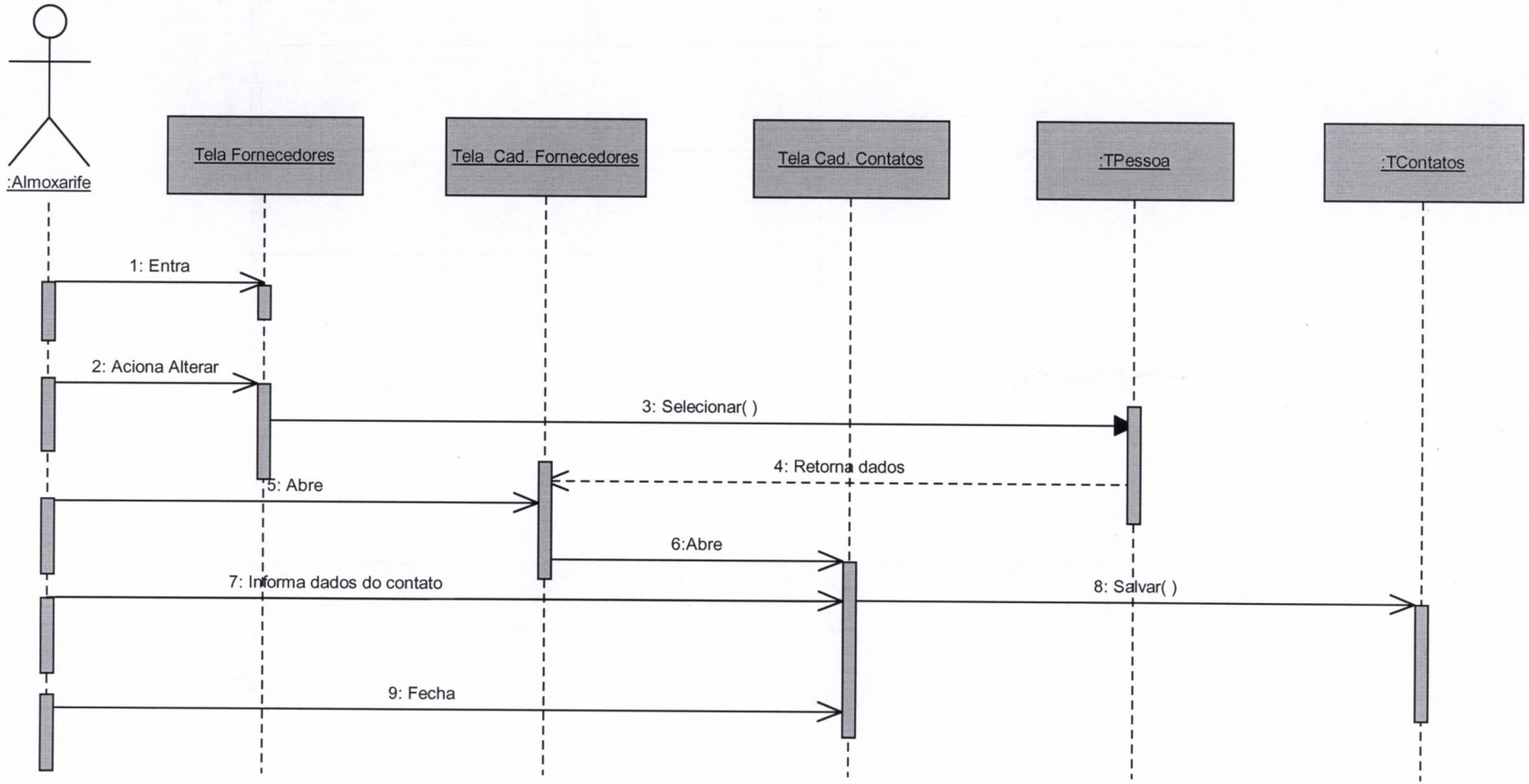


Diagrama de Sequência - Cadastrar Telefone

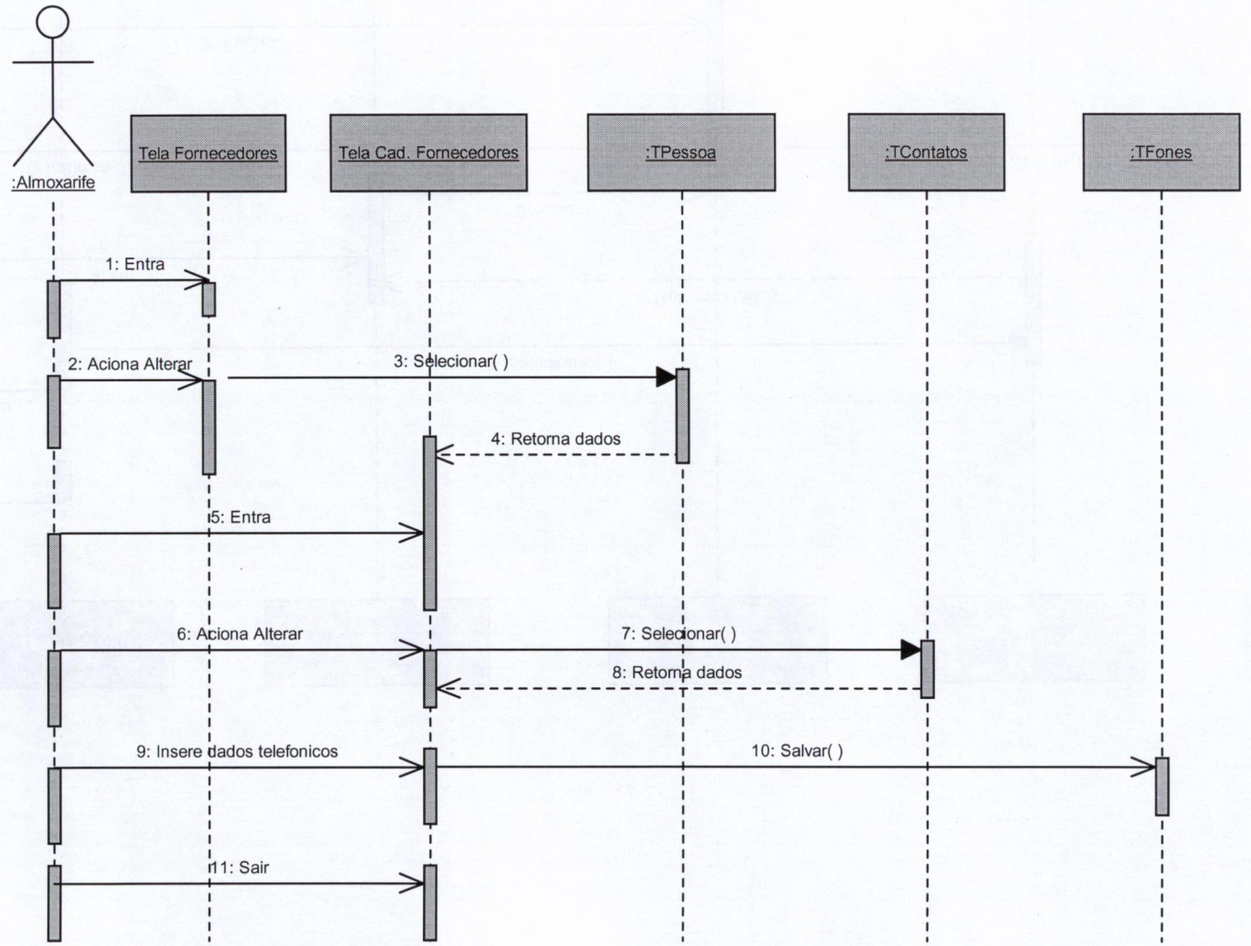


Diagrama de Sequência - Cancelar OS

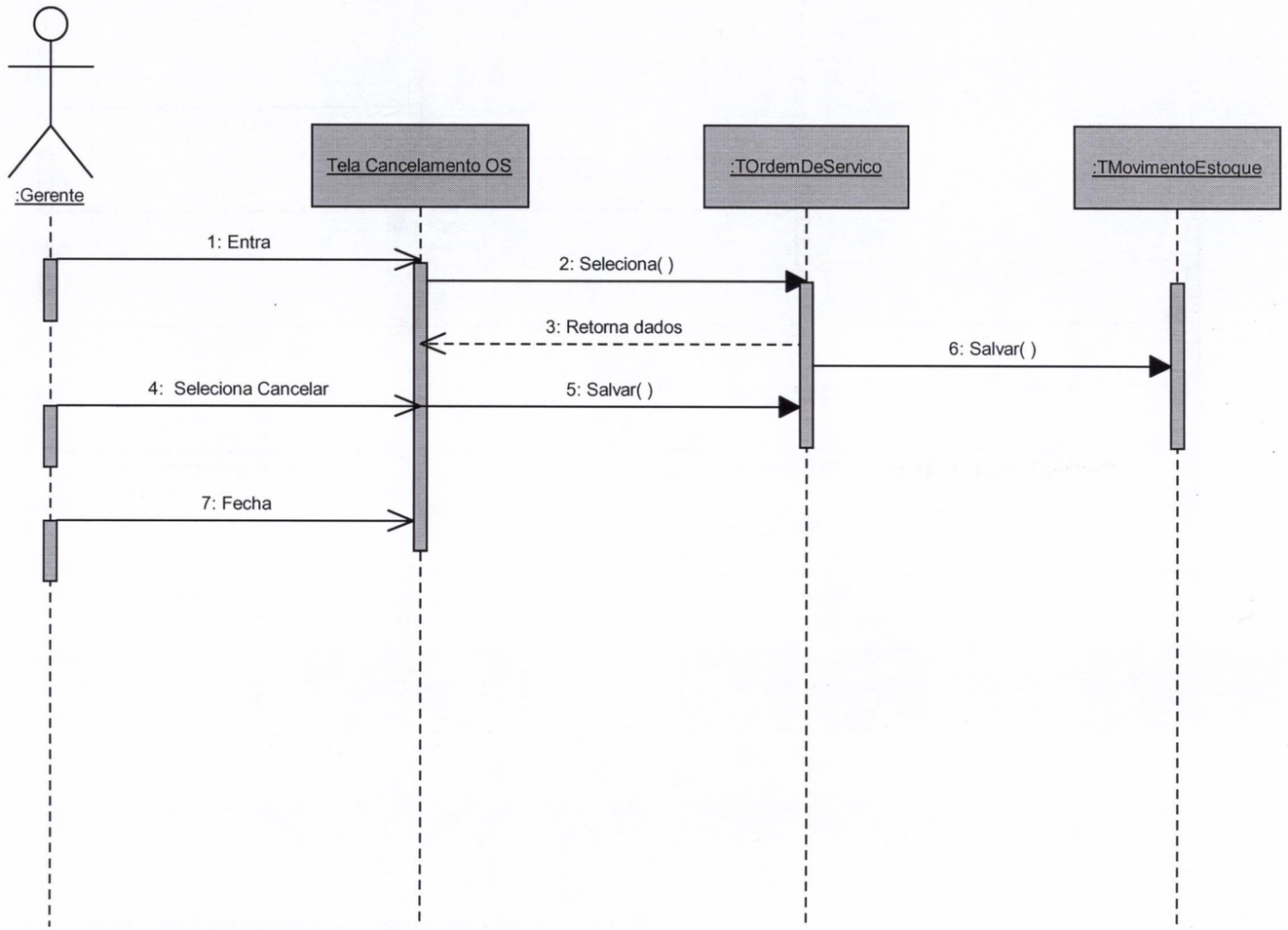


Diagrama de Sequência - Pesquisar OS

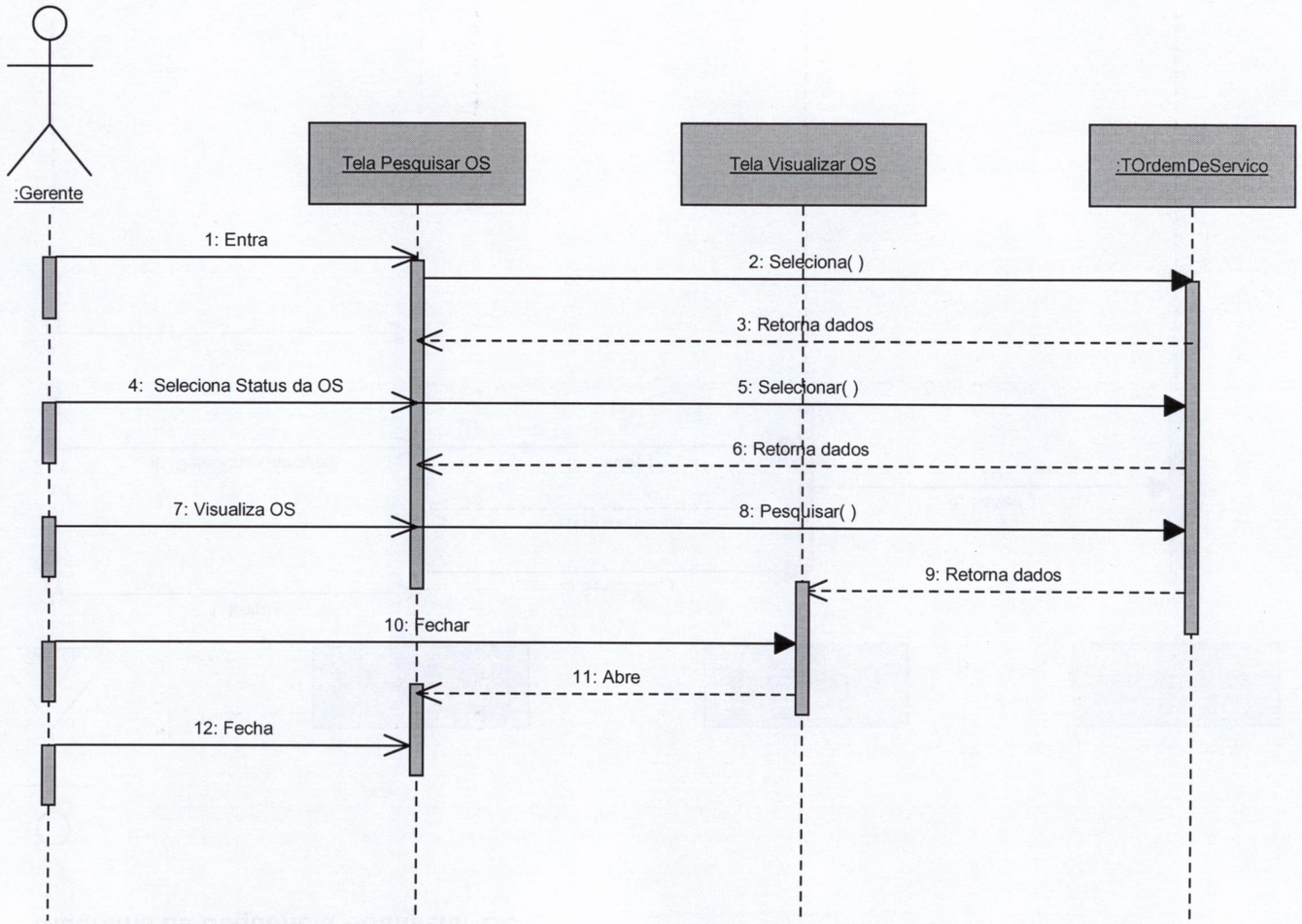


Diagrama de Sequência - Emitir Relatório

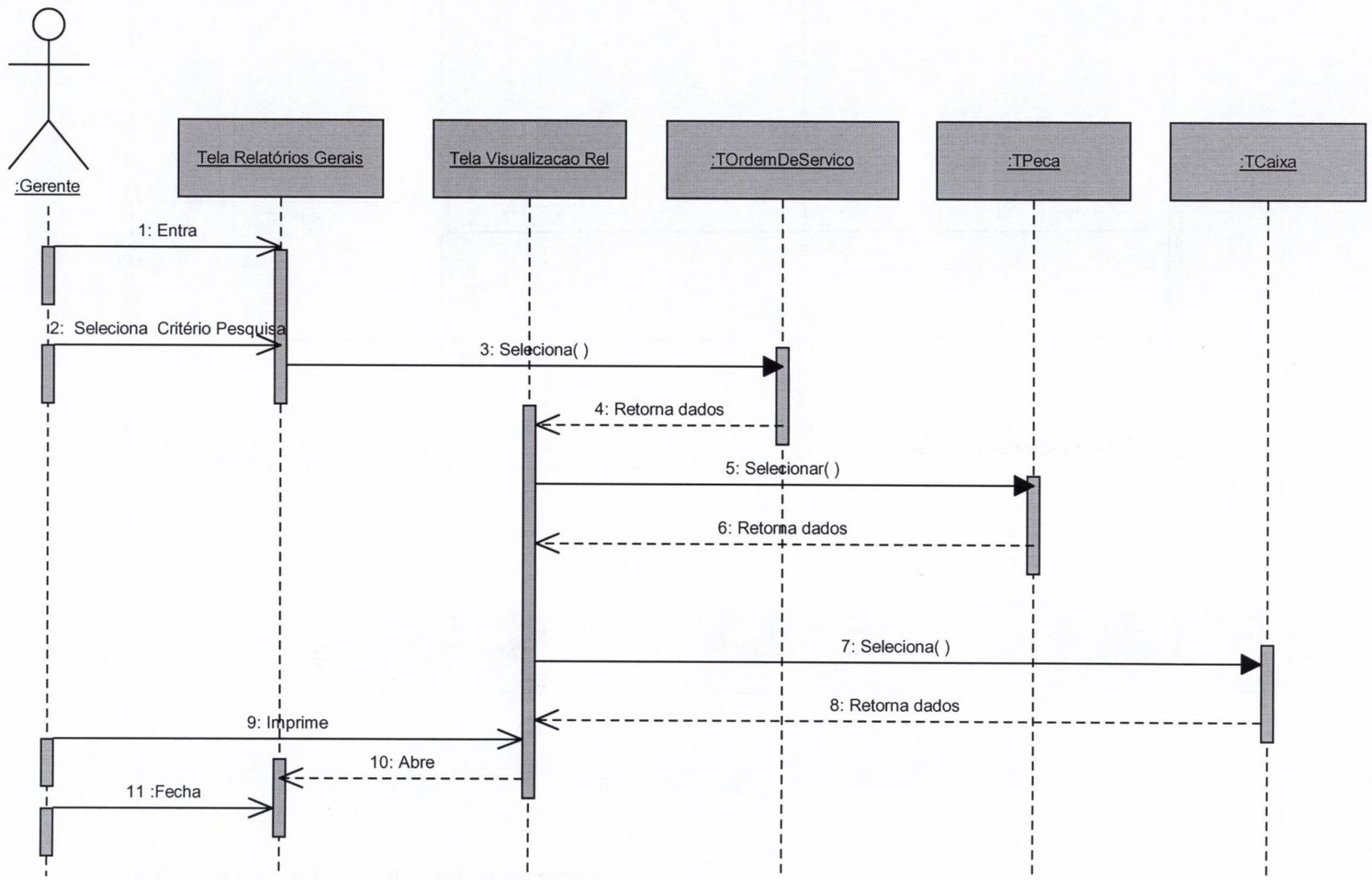


Diagrama de Sequência - Contas à Pagar

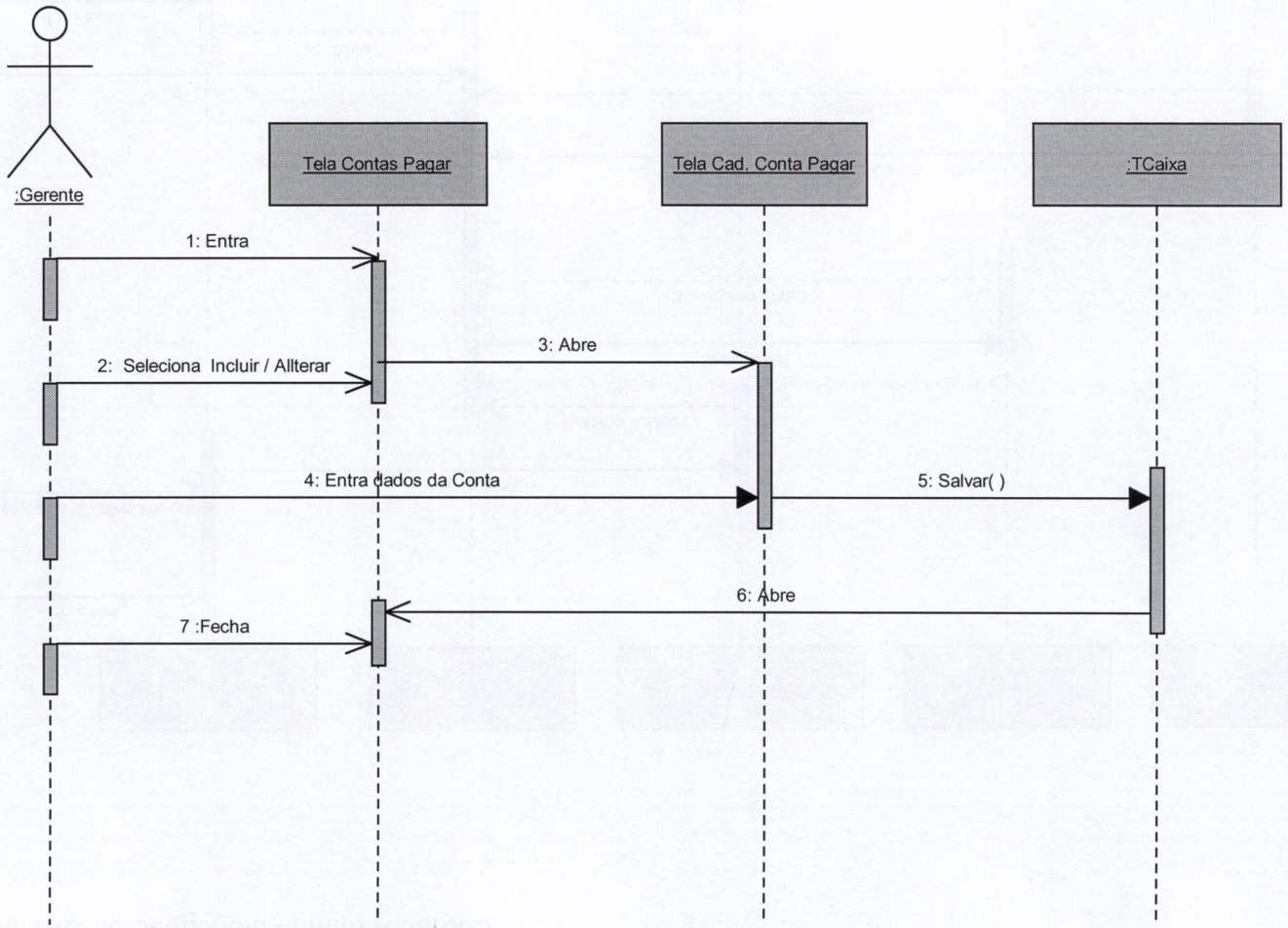


Diagrama de Sequência - Contas à Receber

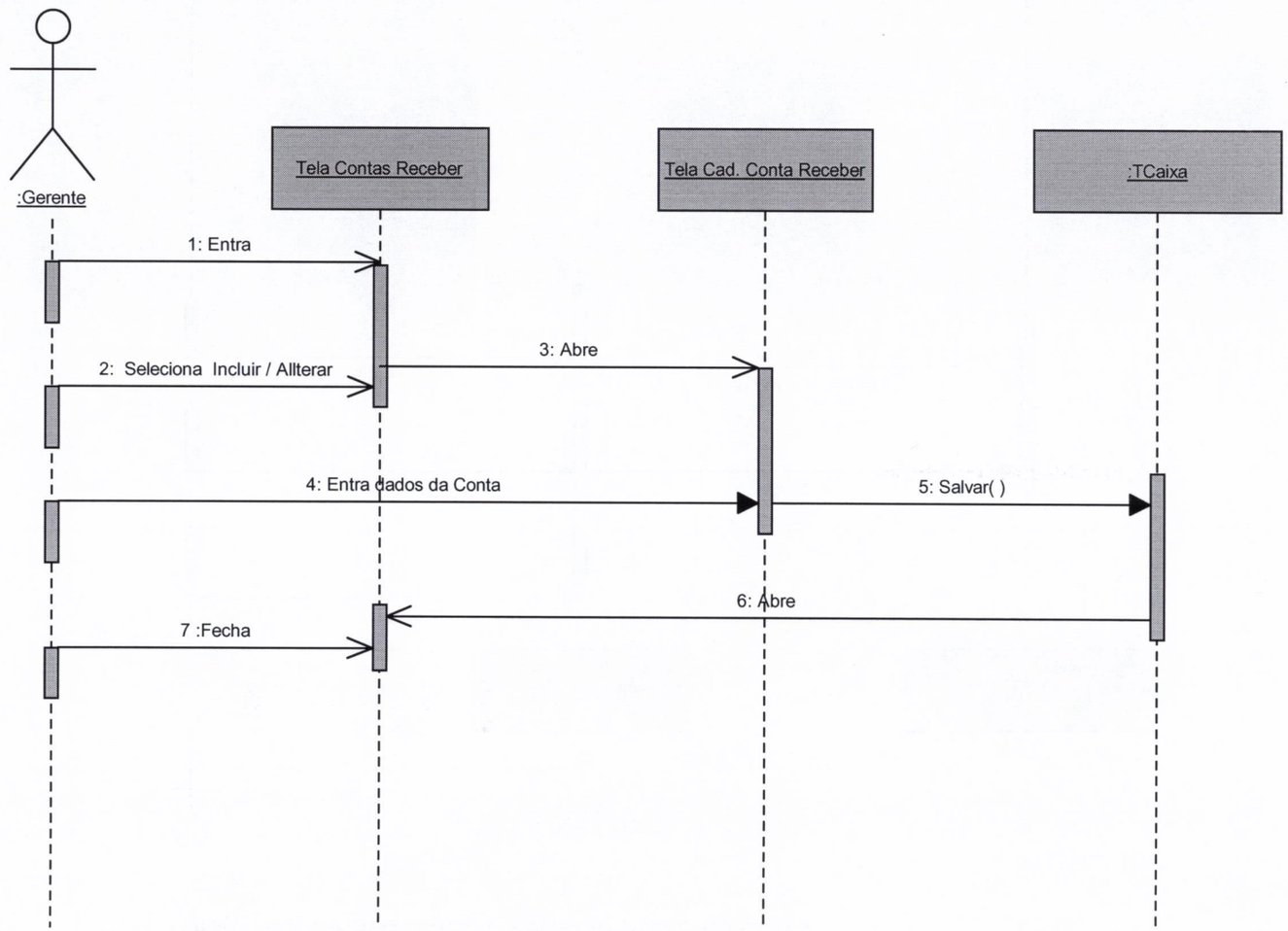


Diagrama de Sequência - Verificar Fluxo de Caixa

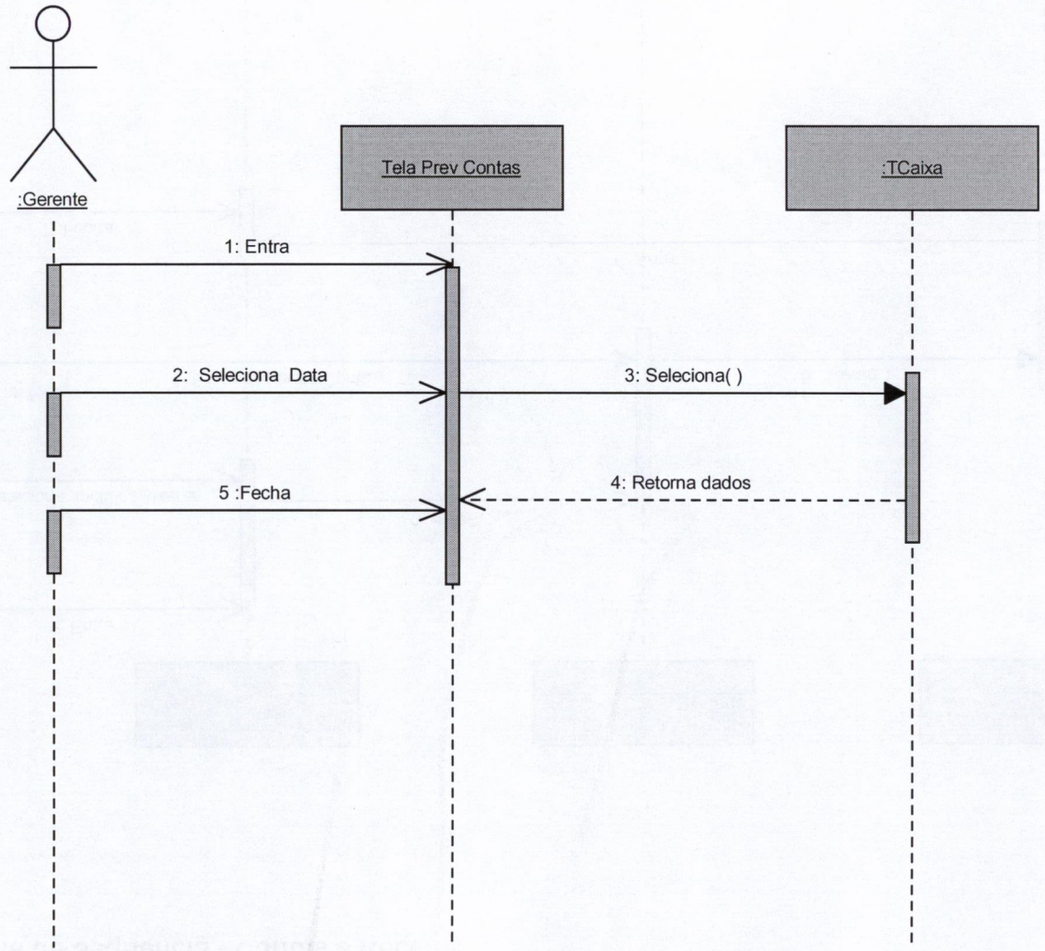


Diagrama de Sequência - Cadastrar Cliente

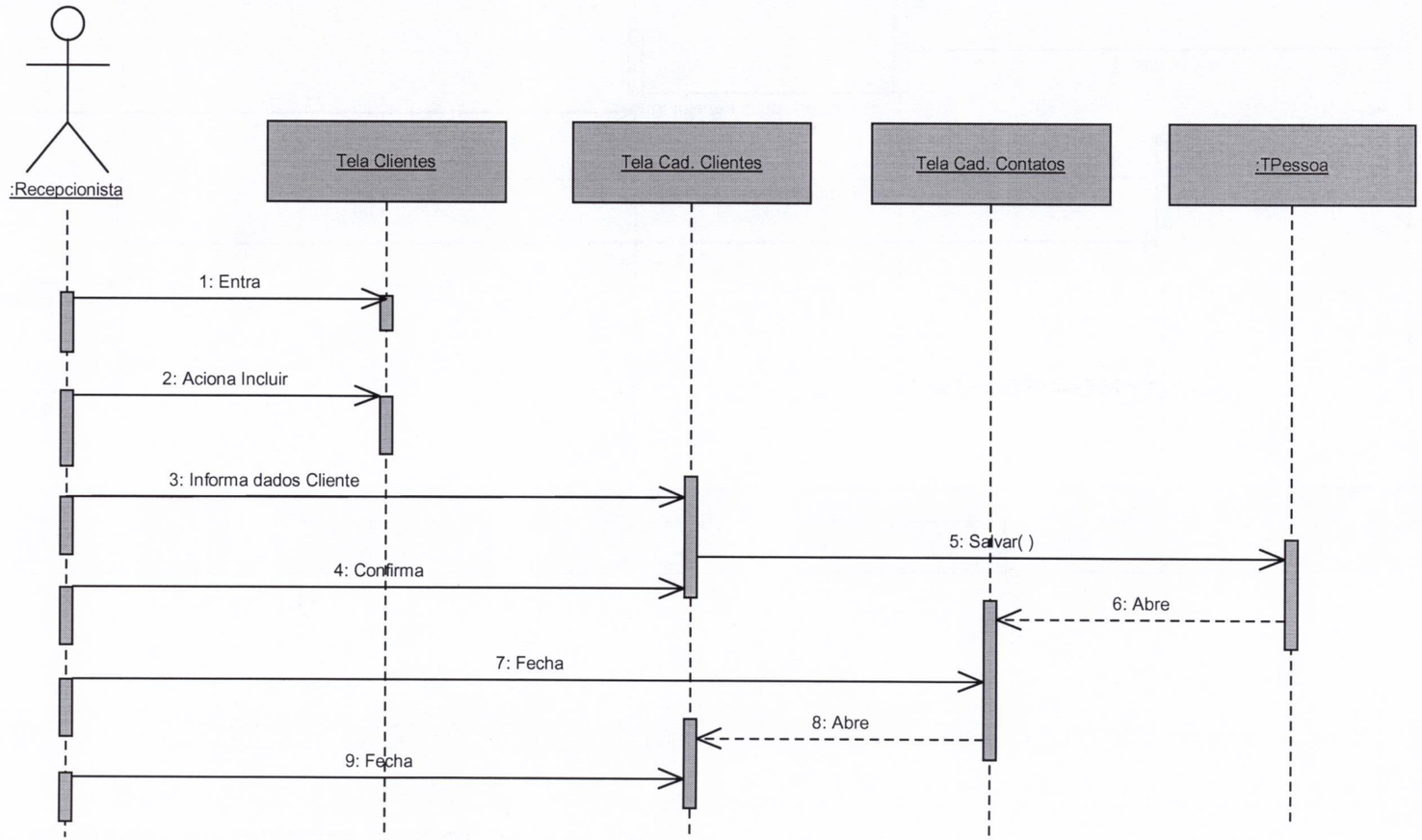


Diagrama de Sequência - Cadastrar Contato

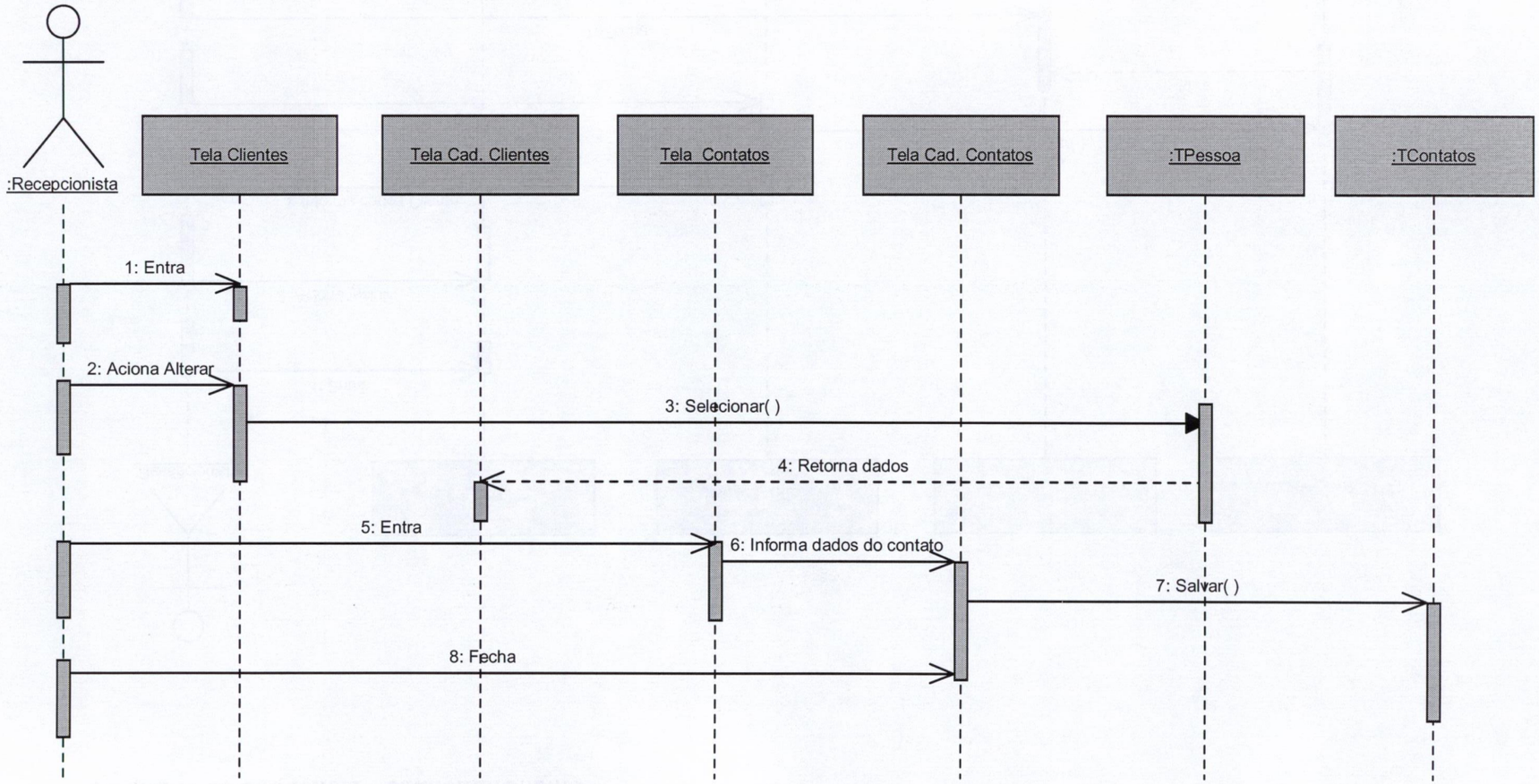


Diagrama de Sequência - Cadastrar Telefone

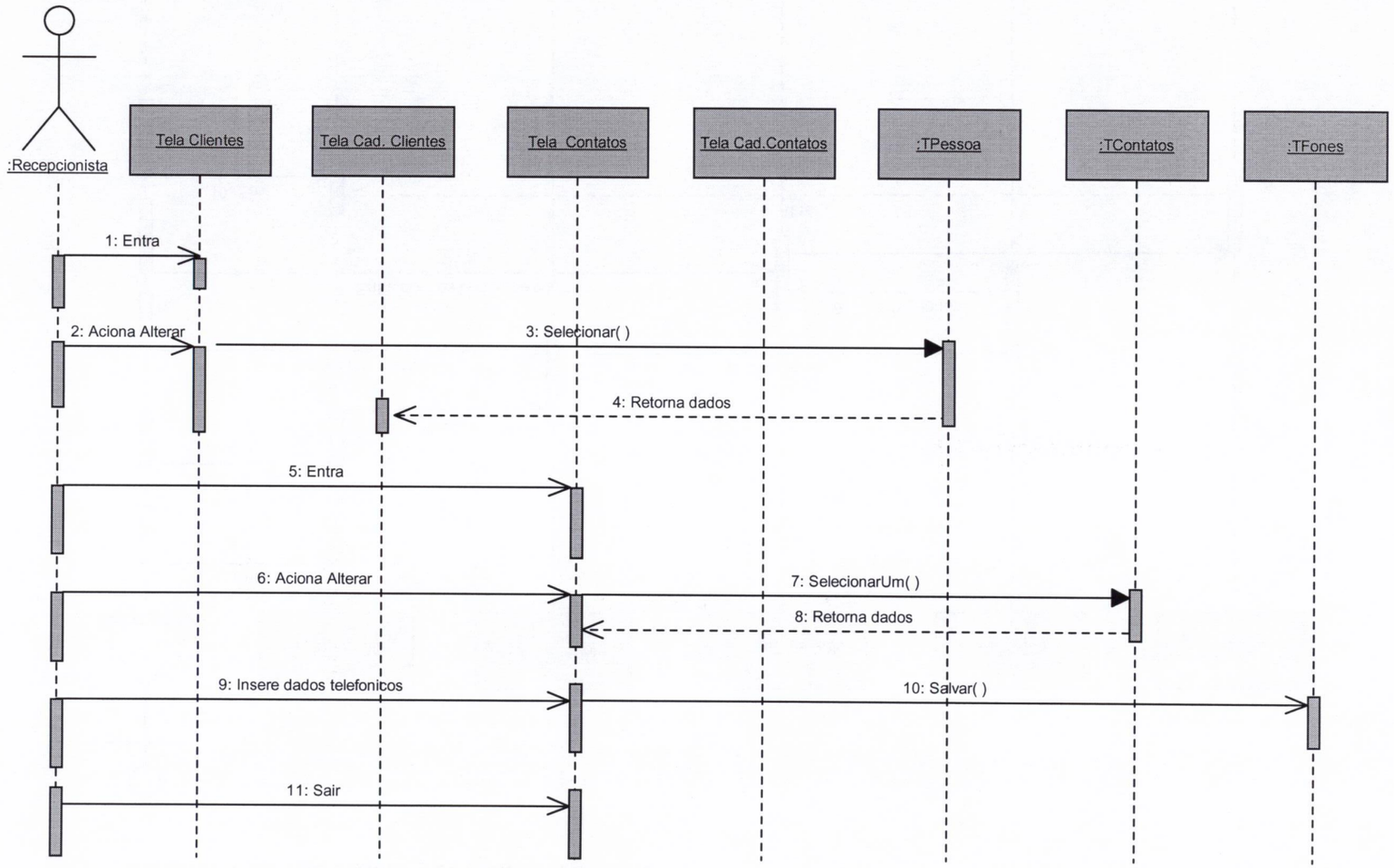


Diagrama de Sequência - Cadastrar Contrato

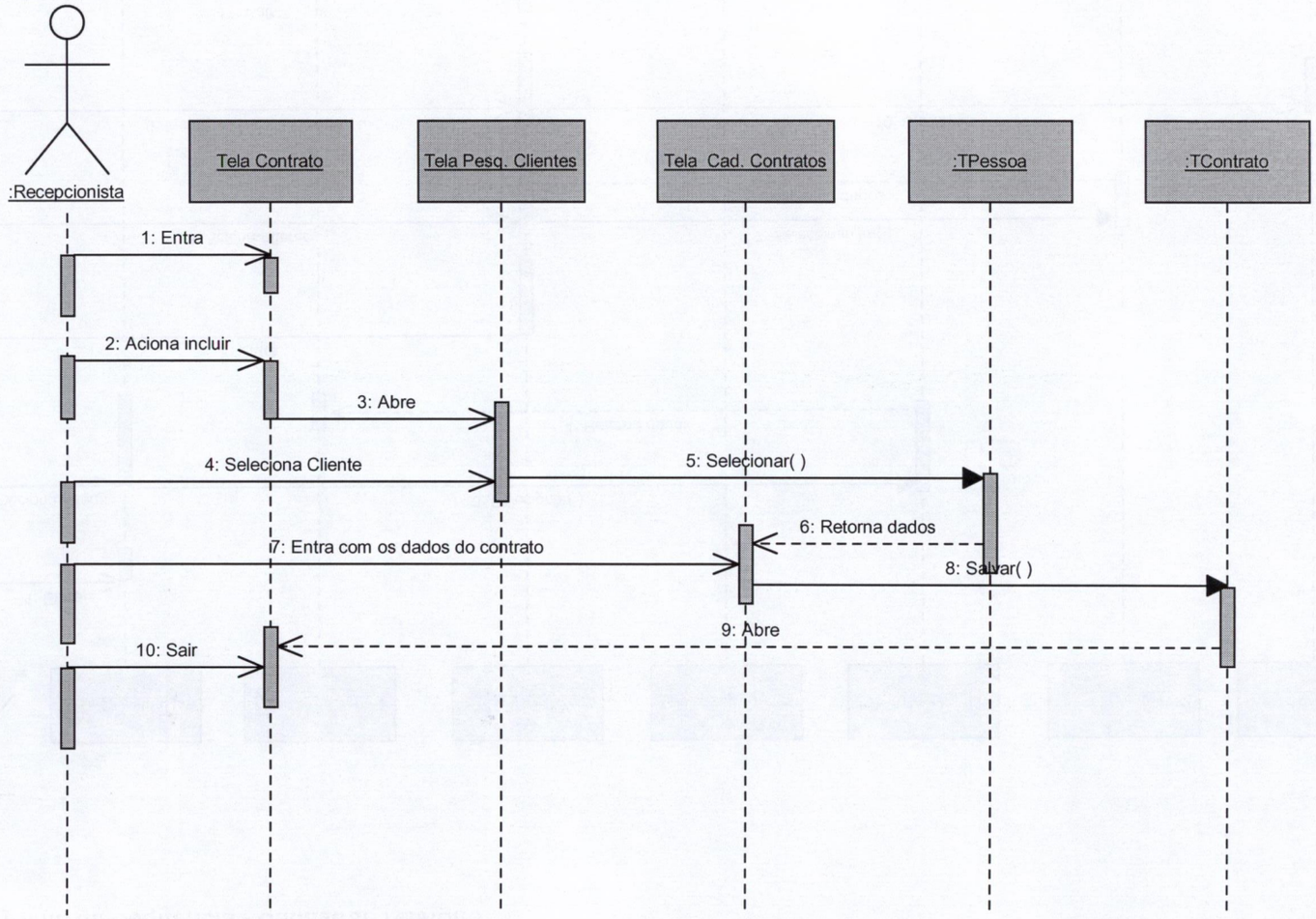


Diagrama de Sequência - Cadastrar Equipamentos por Contrato

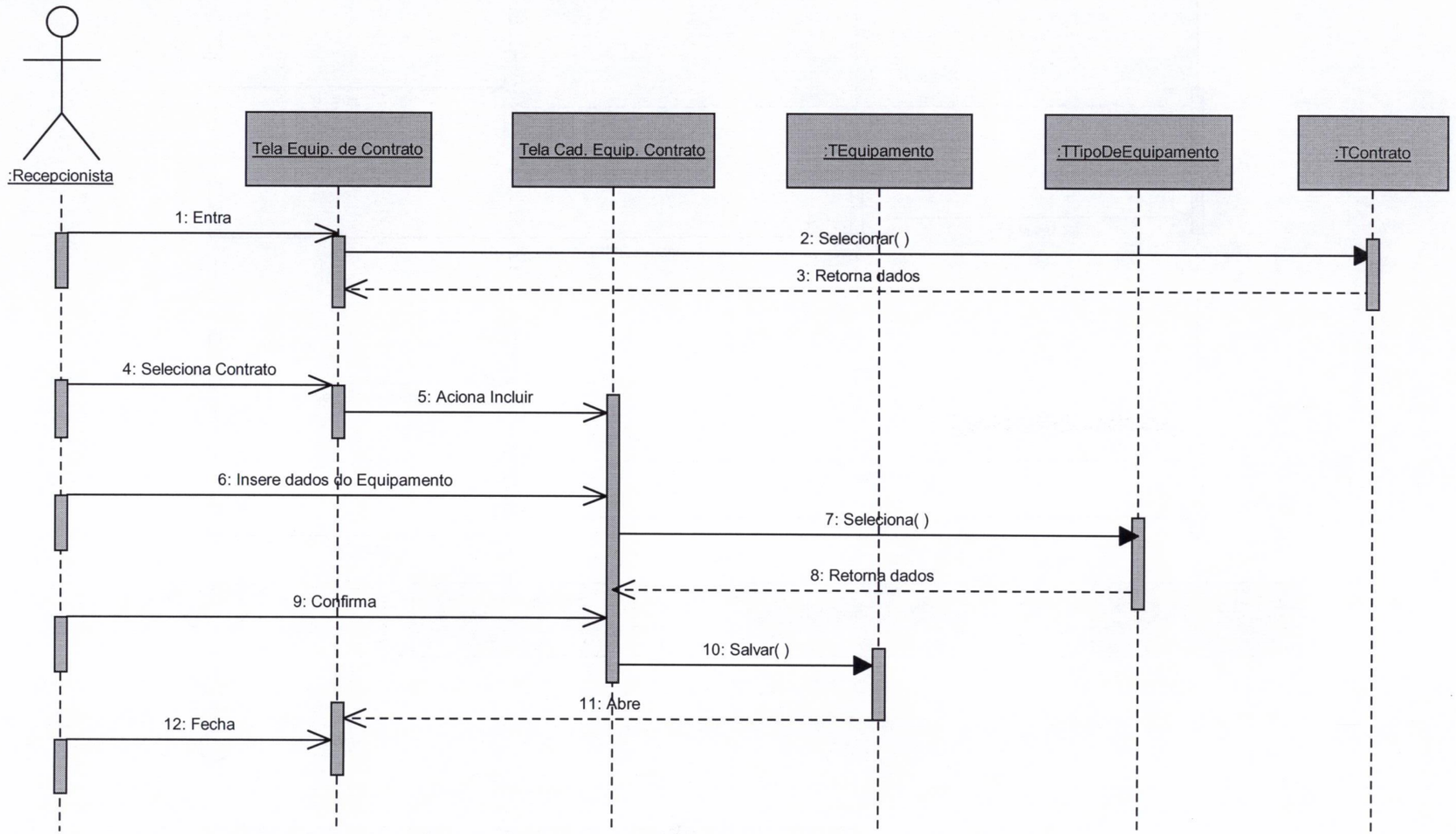


Diagrama de Sequência - Cadastrar Tipo do Equipamento

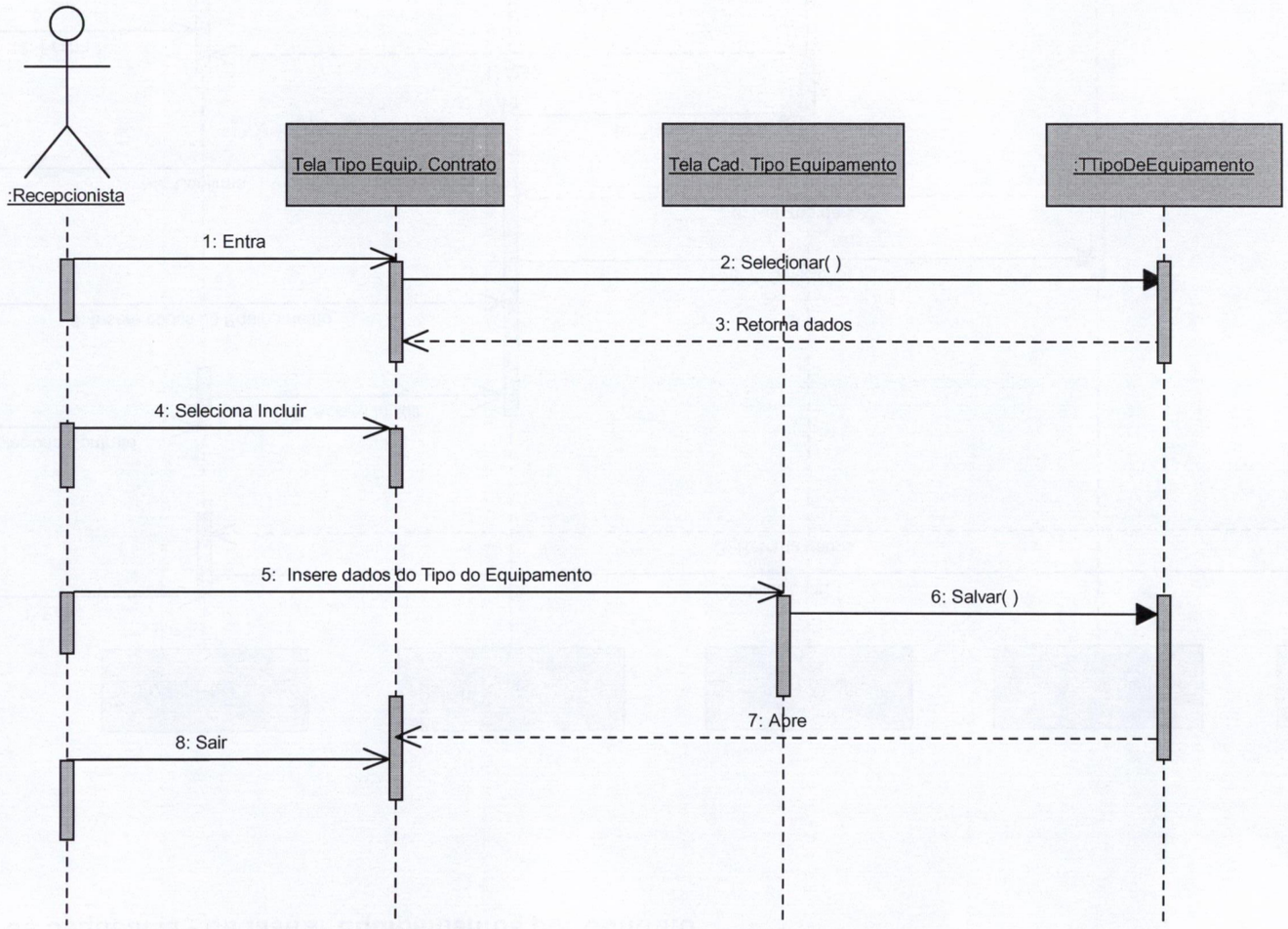


Diagrama de Sequência - Abrir Orçamento (OS)

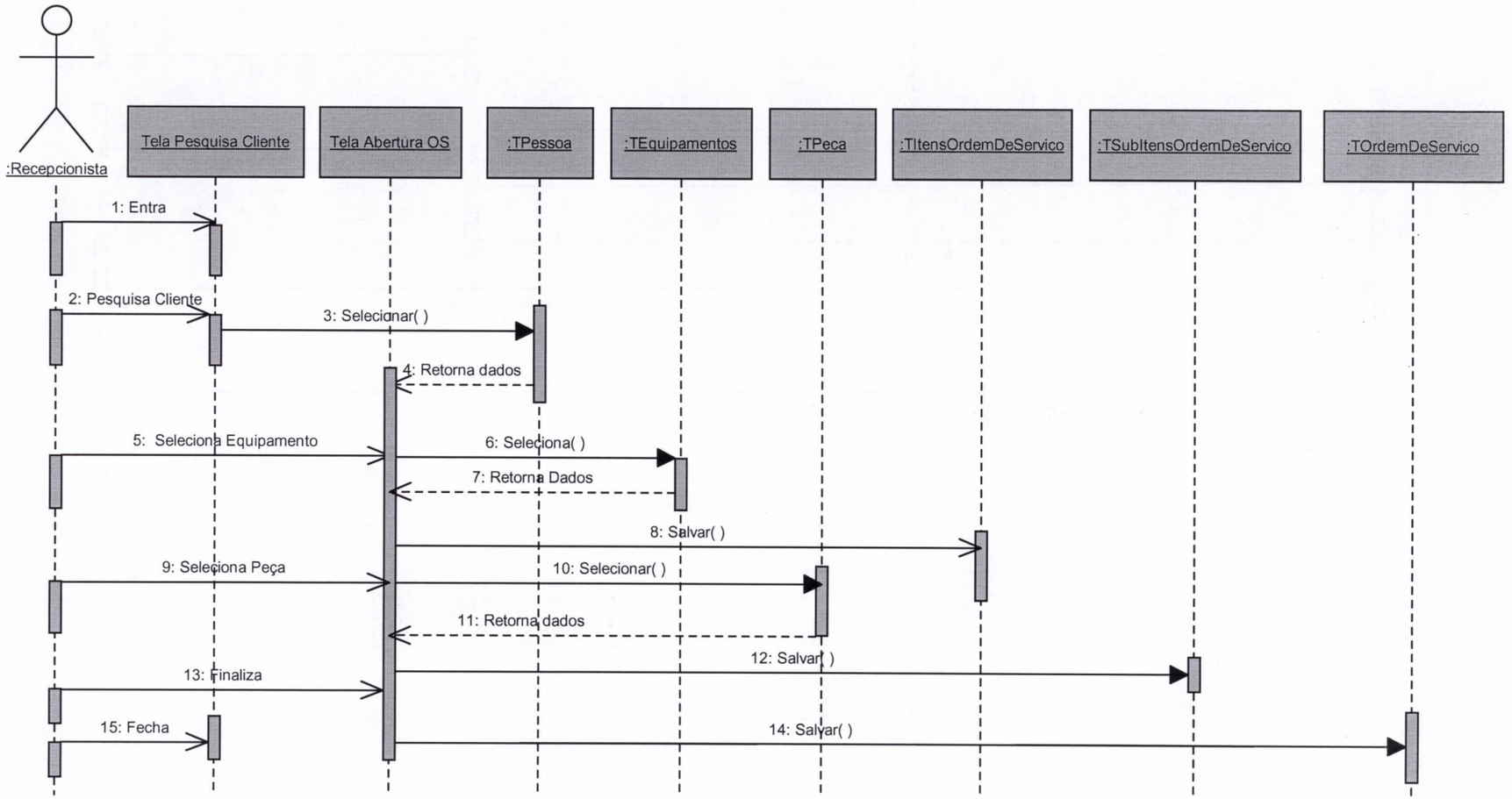


Diagrama de Sequência - Alterar / Liberar para Orçar

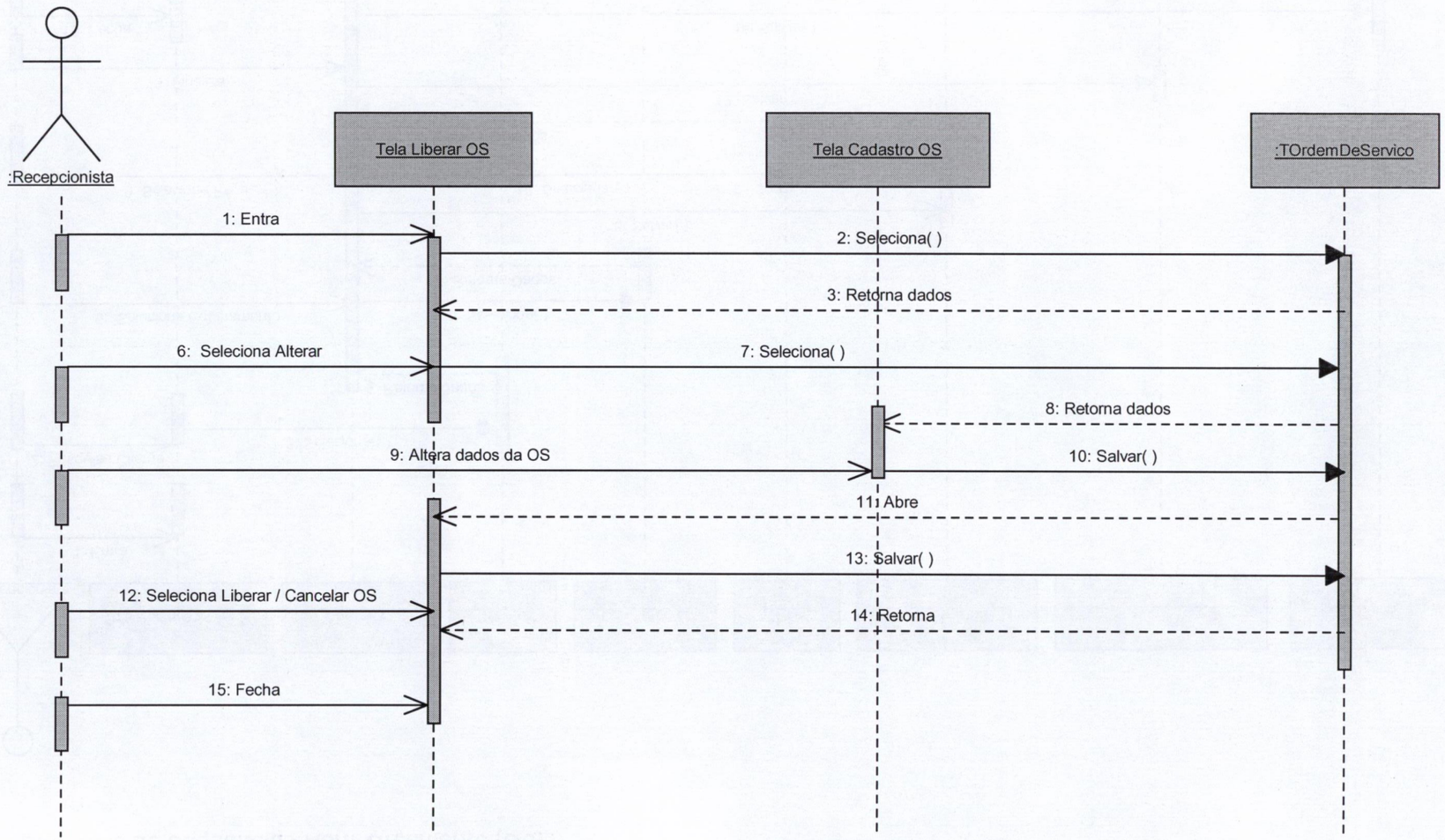


Diagrama de Sequência - Aprovar Orçamento

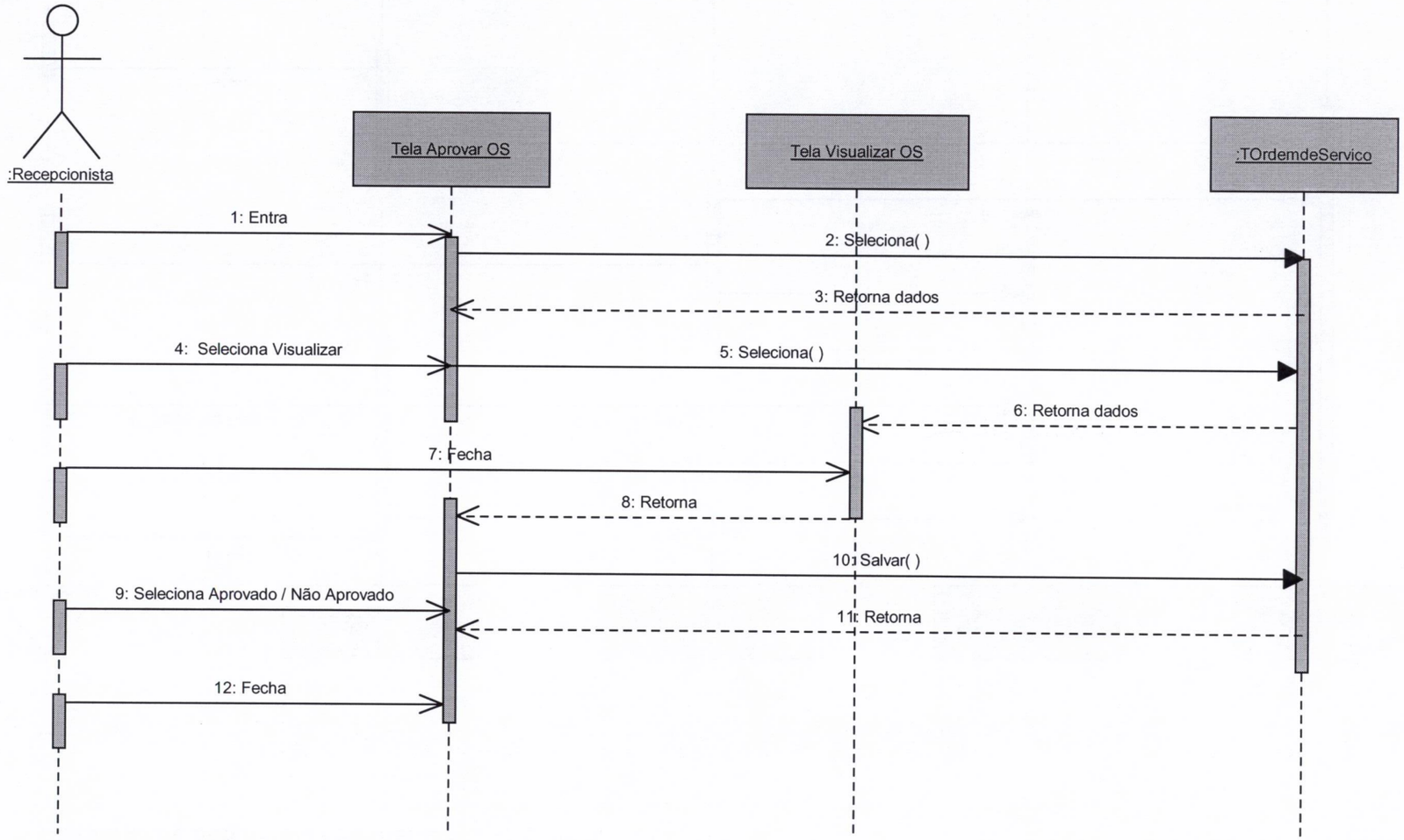


Diagrama de Sequência - Fechar OS

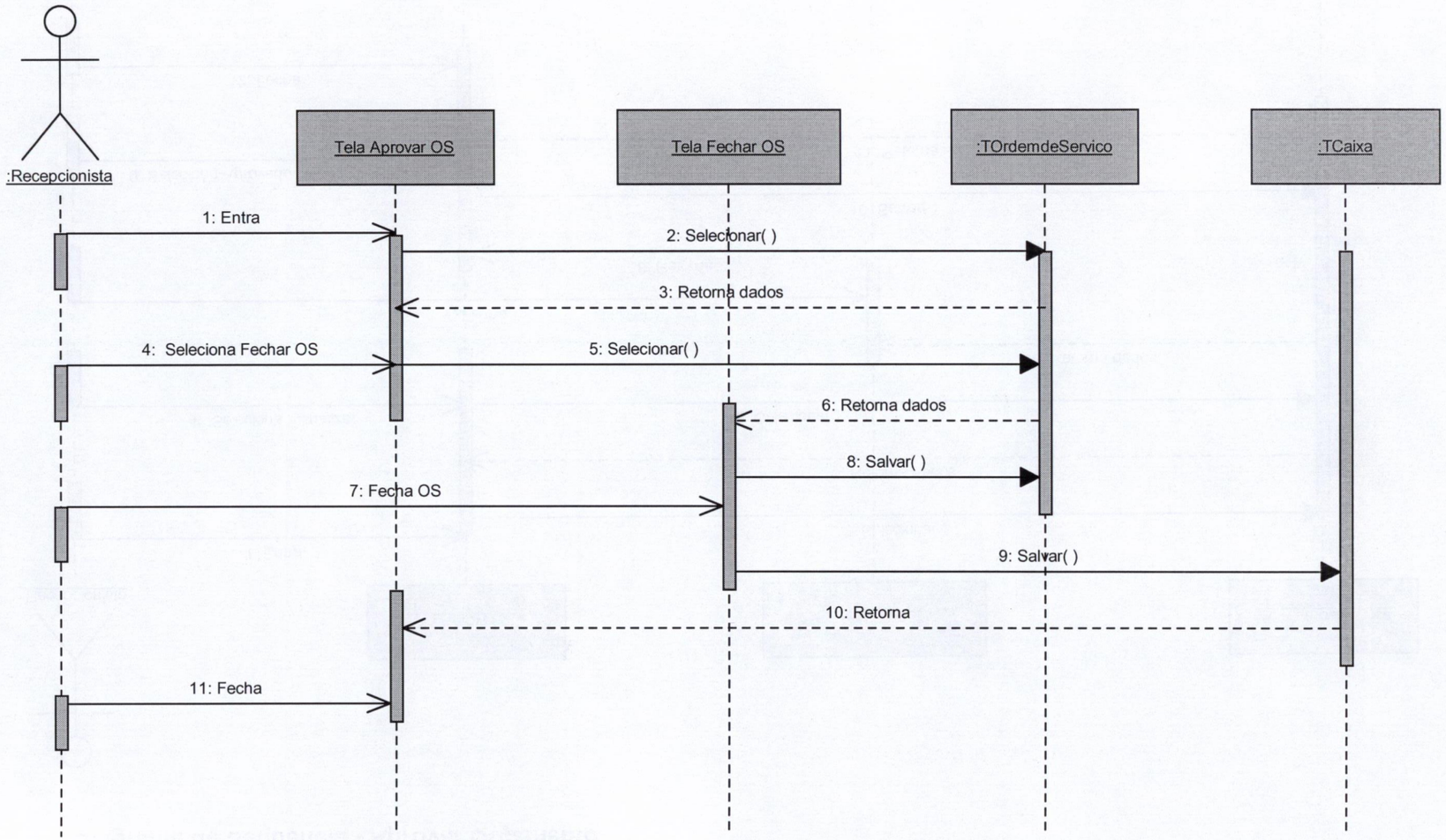


Diagrama de Sequência - Pesquisar OS

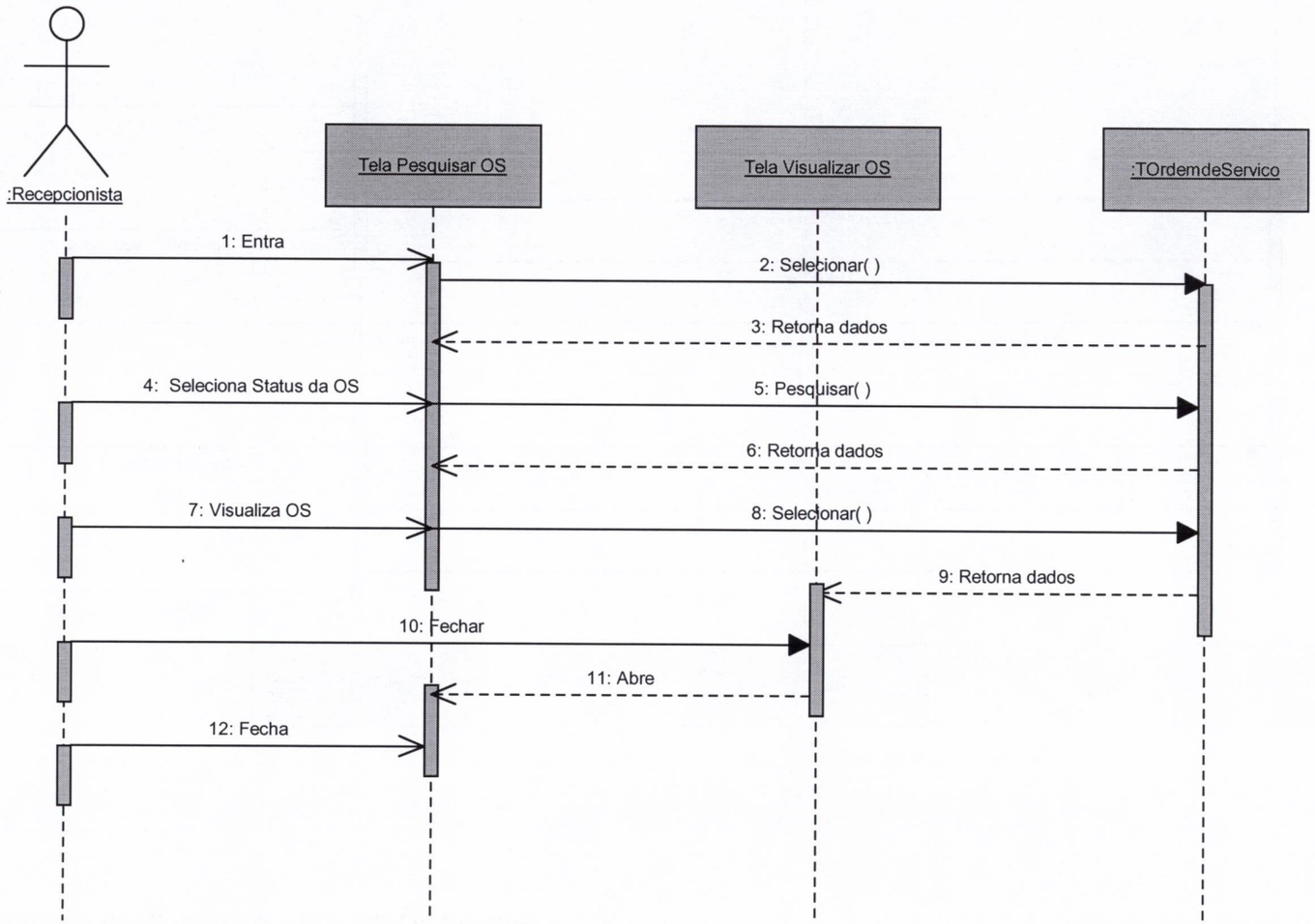


Diagrama de Sequência - Efetuar Orçamento

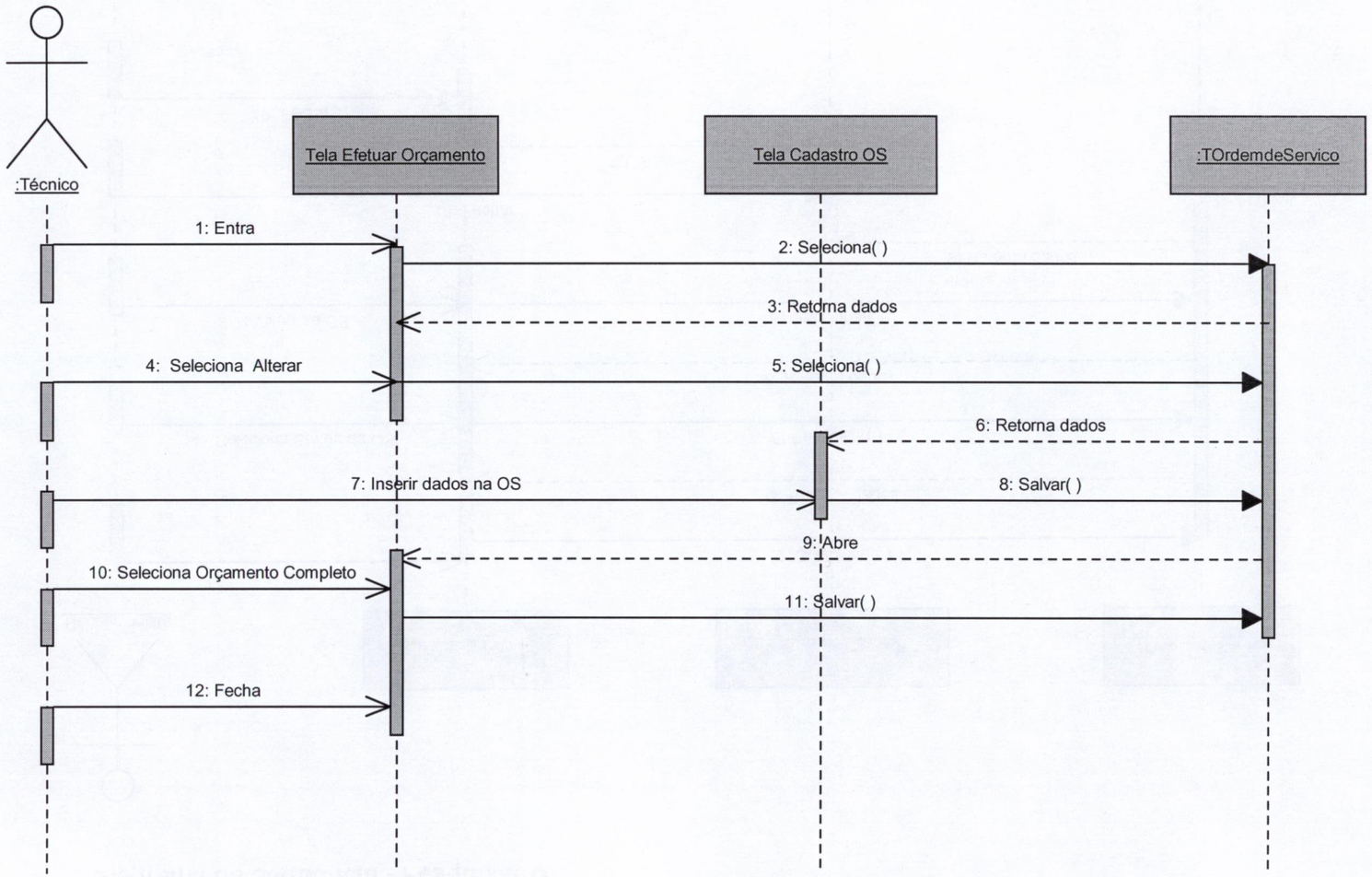
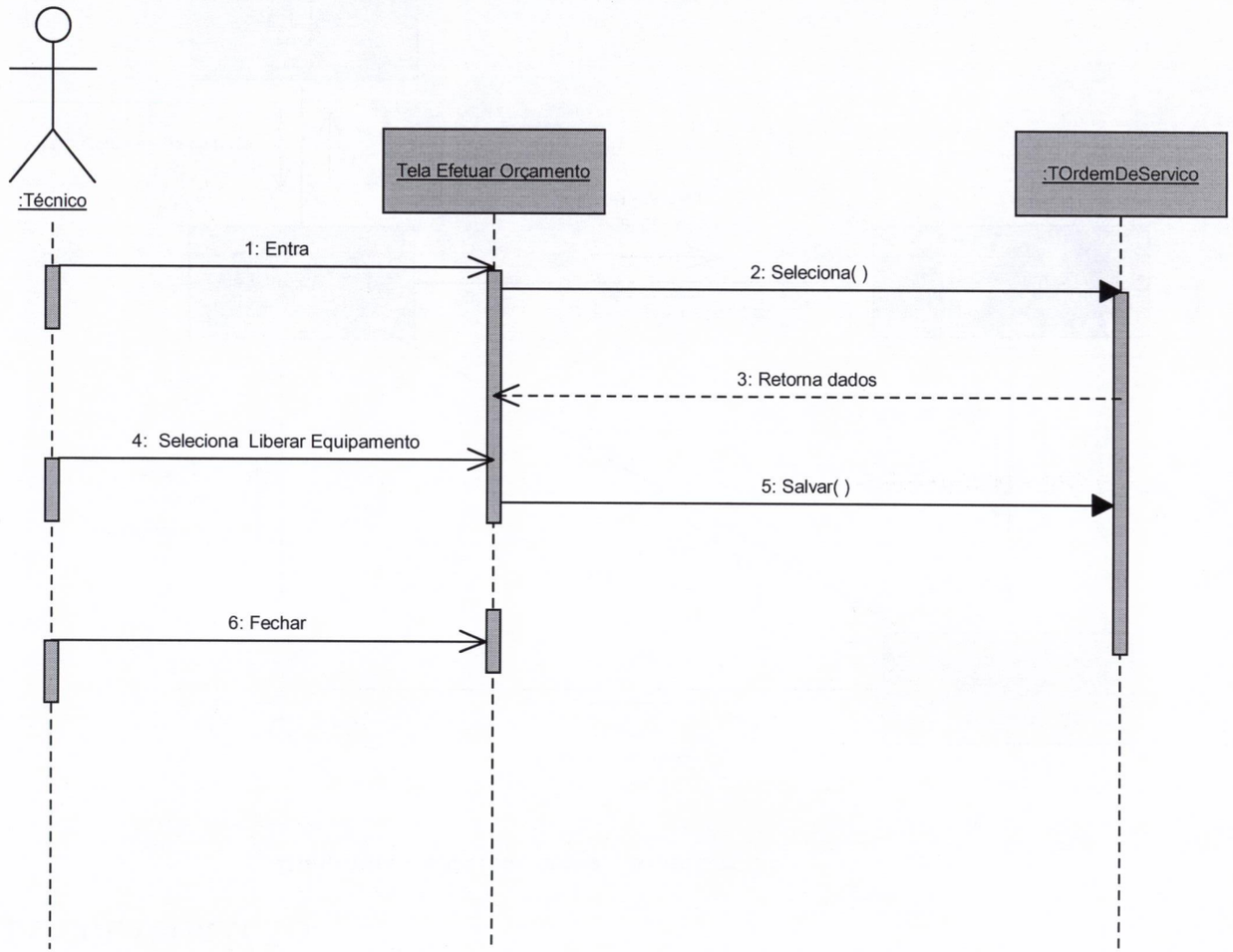


Diagrama de Sequência - Orçamentos Não Aprovados



10 DIAGRAMA DE COLABORAÇÃO

Diagrama de Colaboração - Atualizar OS

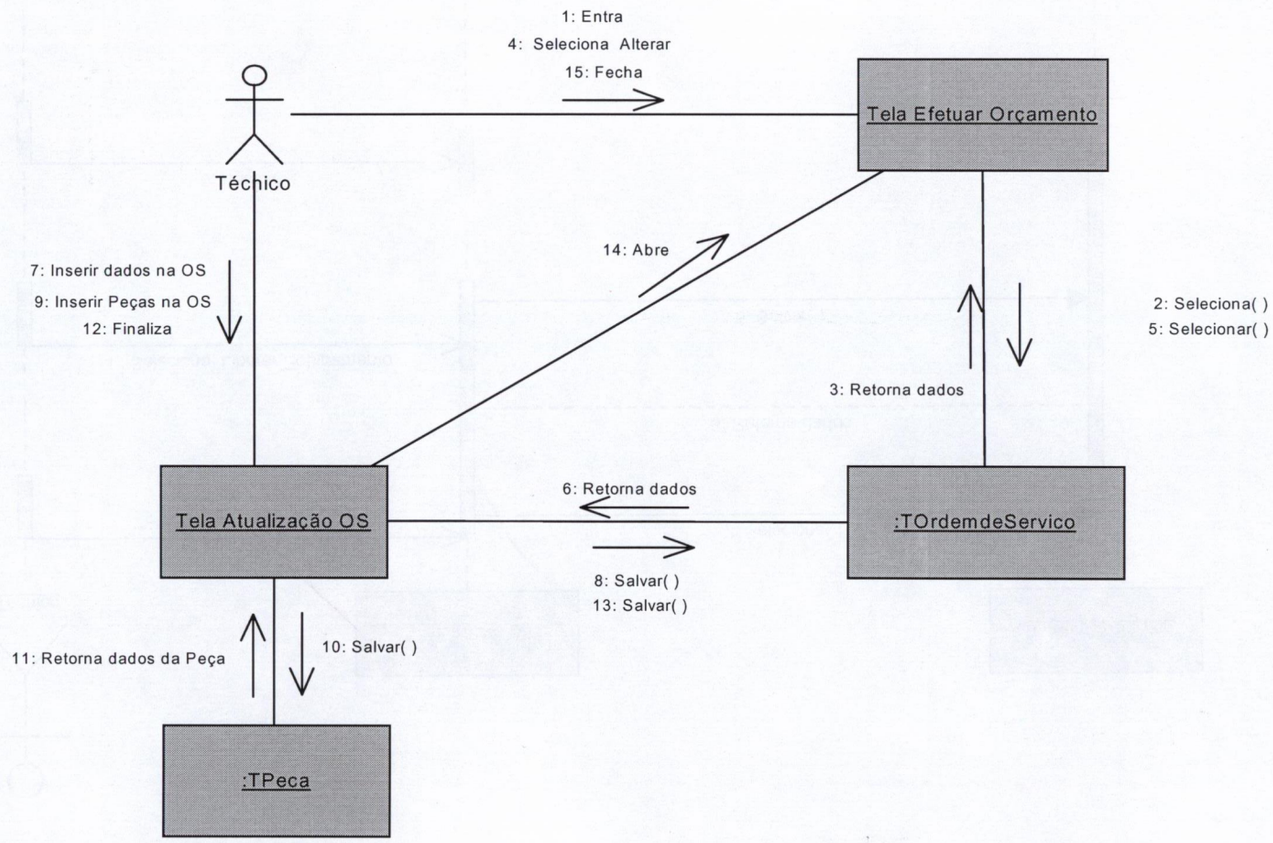


Diagrama de Colaboração - Definir Permissões de Acesso

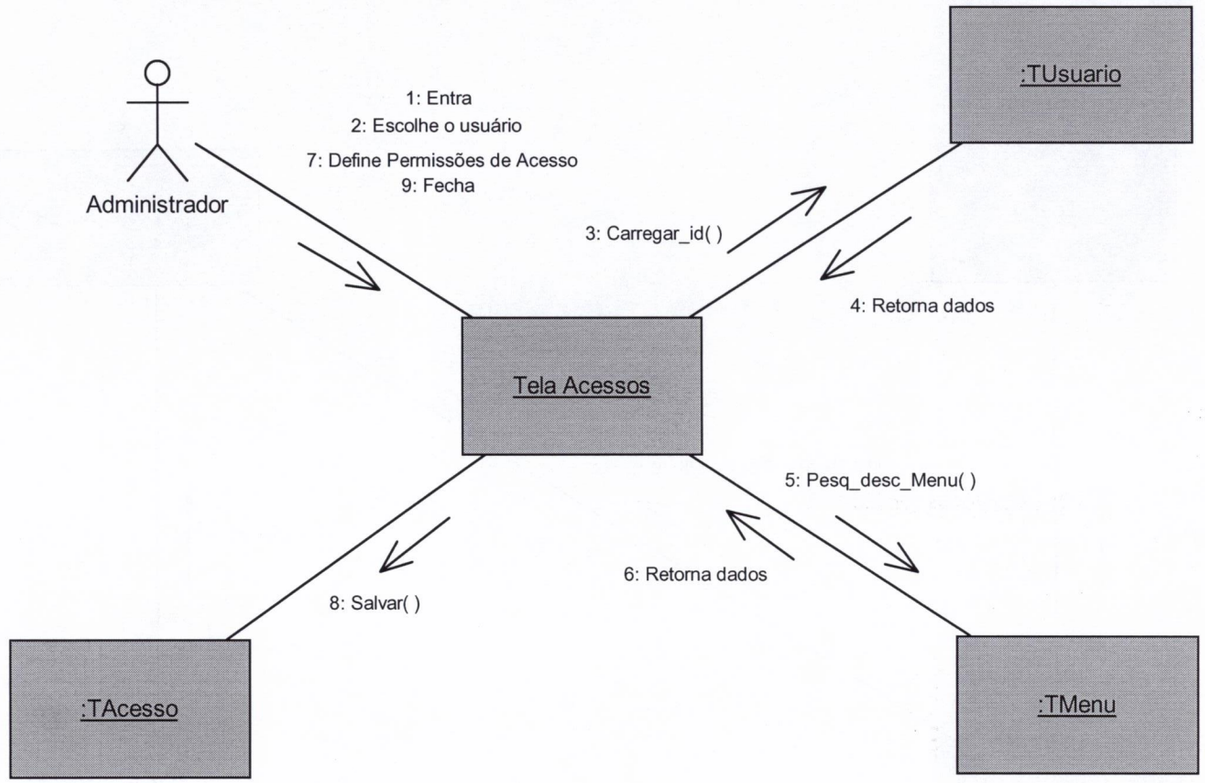


Diagrama de Colaboração - Cadastrar Usuário

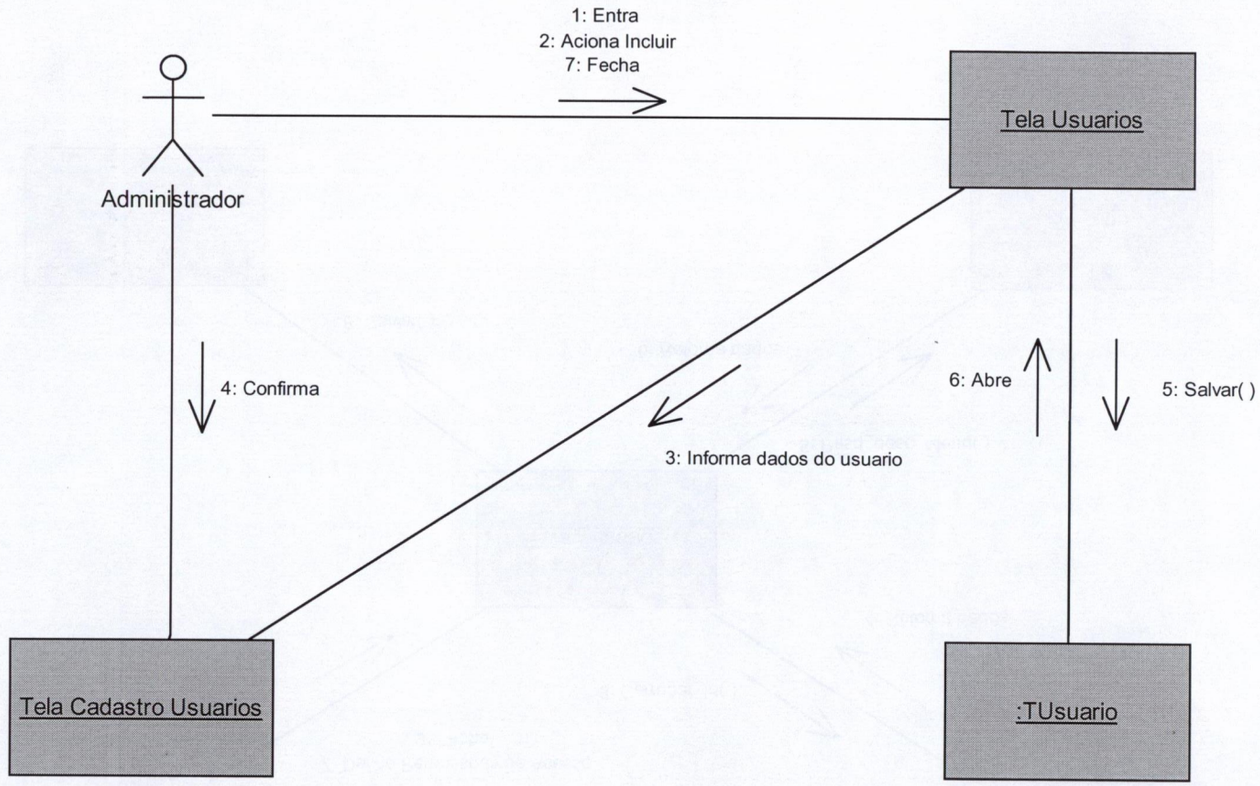


Diagrama de Colaboração - Cadastrar Grupo de Usuários

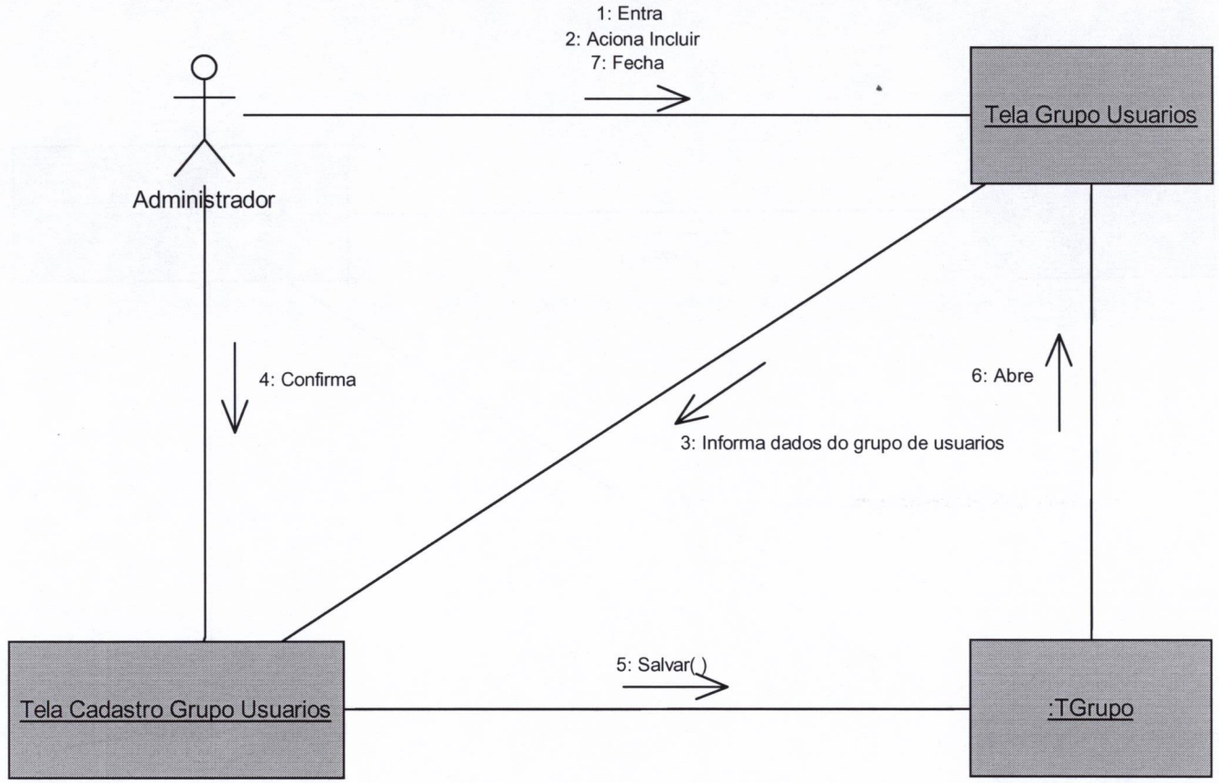


Diagrama de Colaboração - Cadastrar Estados

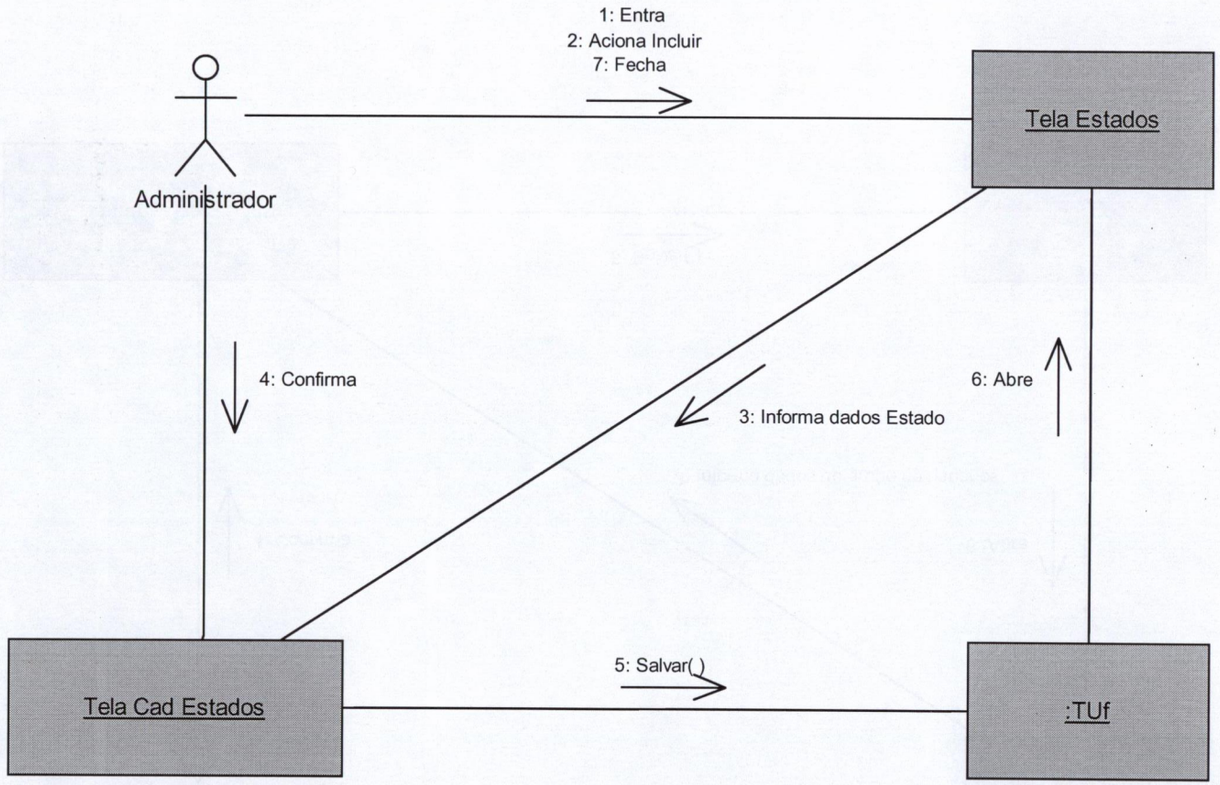


Diagrama de Colaboração - Cadastrar Cidades

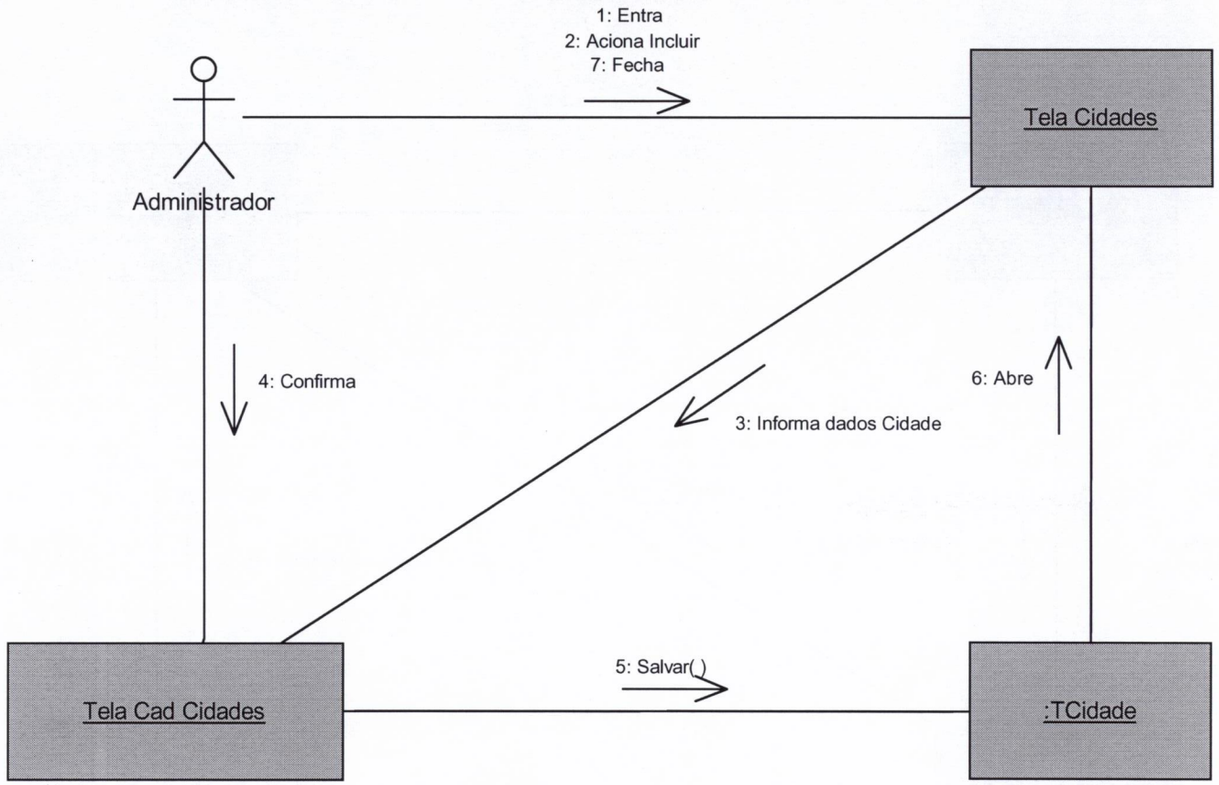


Diagrama de Colaboração - Cadastrar tipo de peça

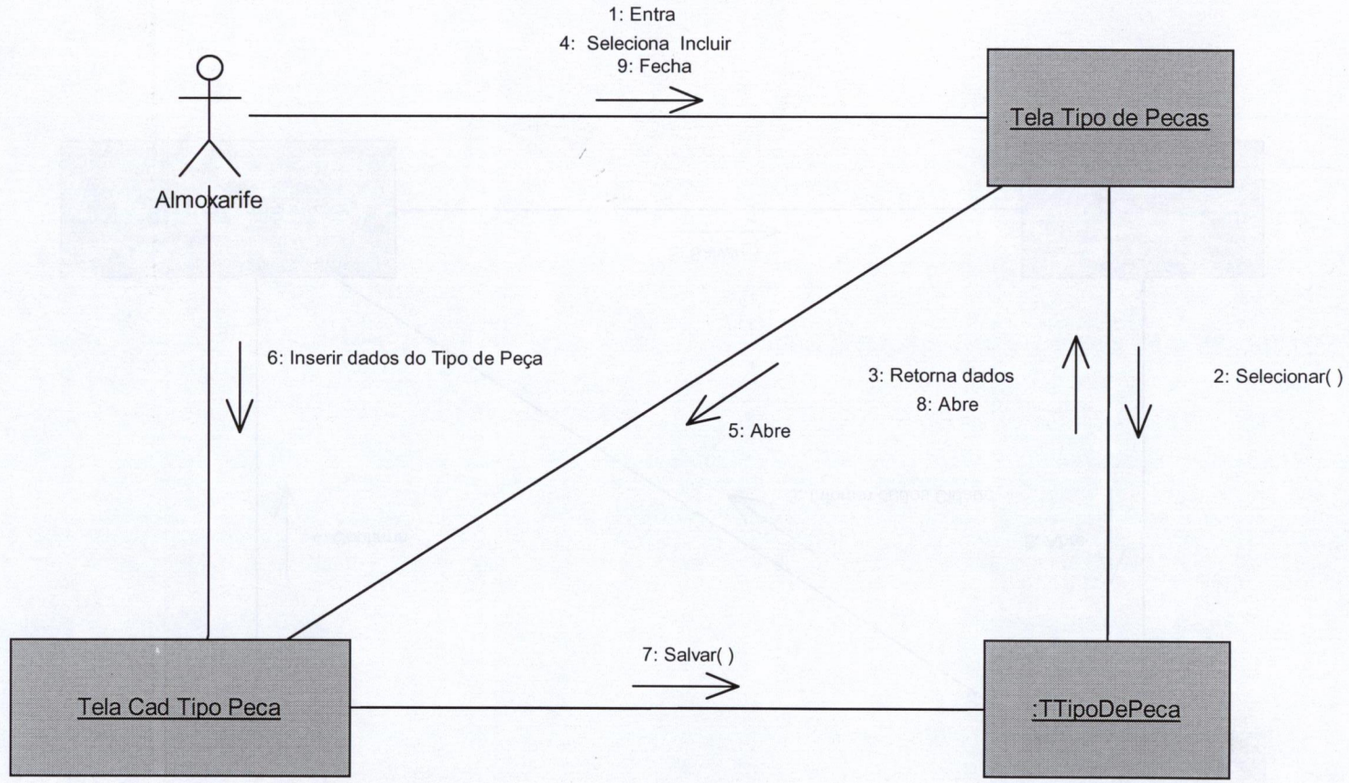


Diagrama de Colaboração - Incluir Peça no Estoque

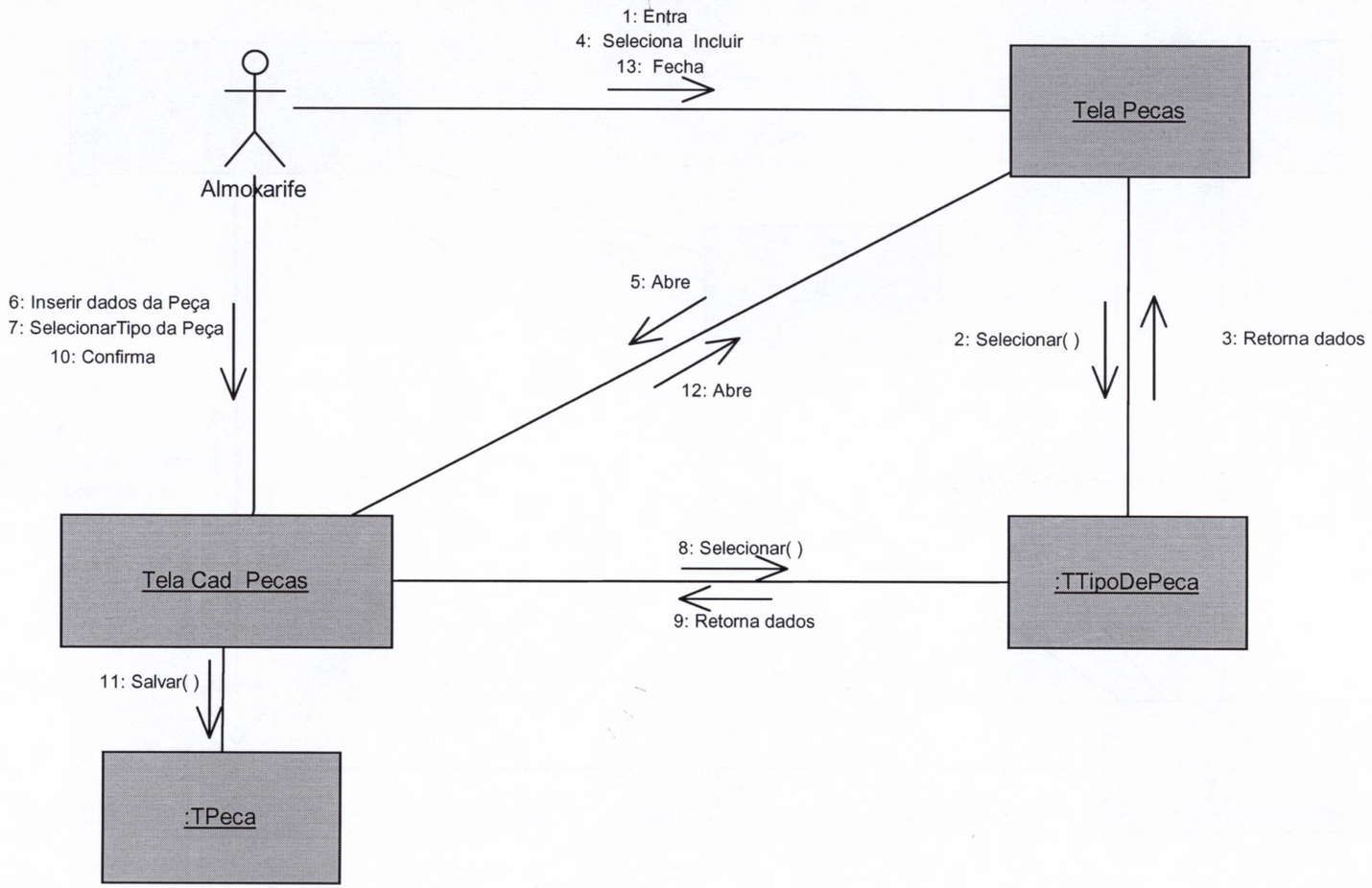


Diagrama de Colaboração - Alterar Peça no Estoque

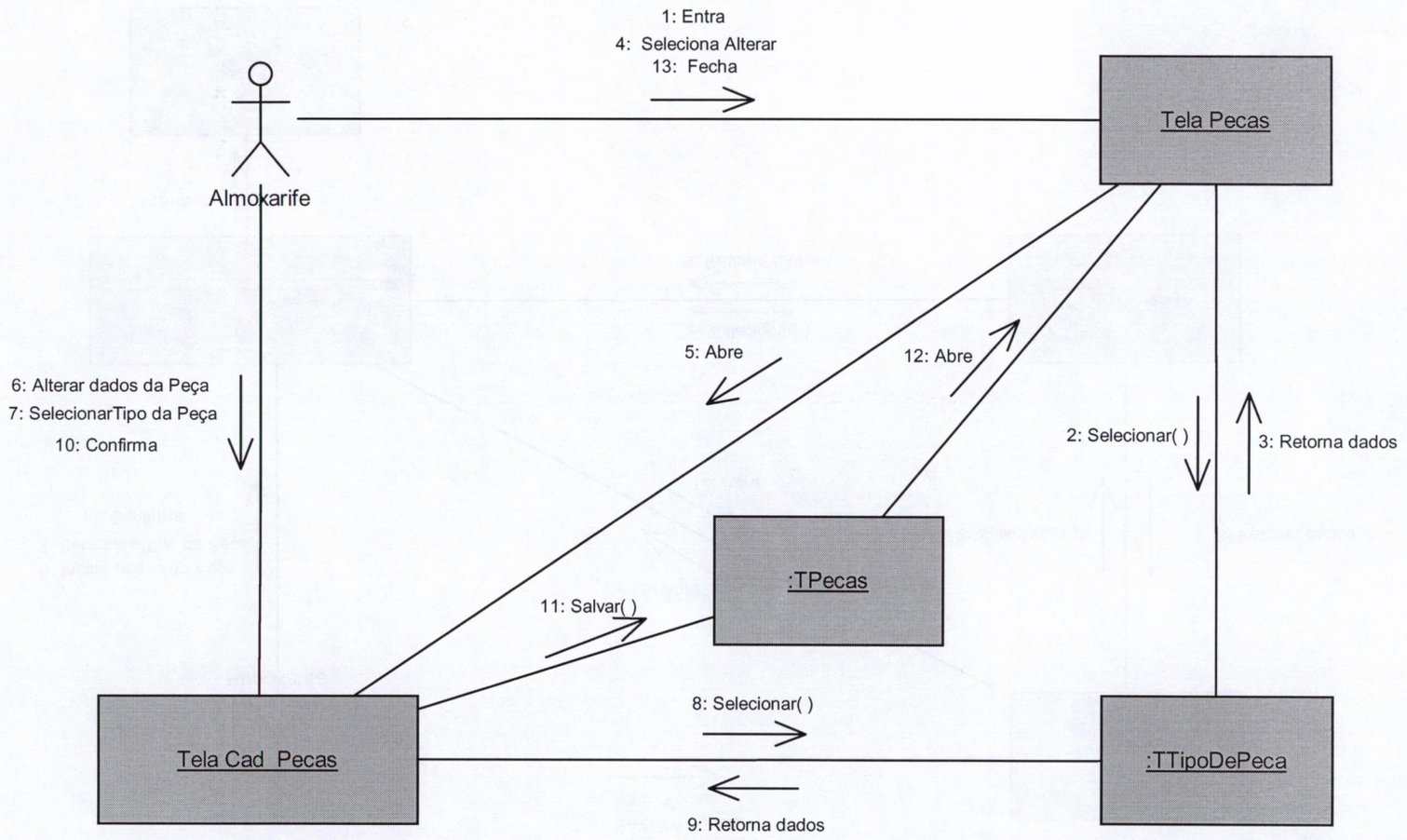


Diagrama de Colaboração - Cadastrar Fornecedor

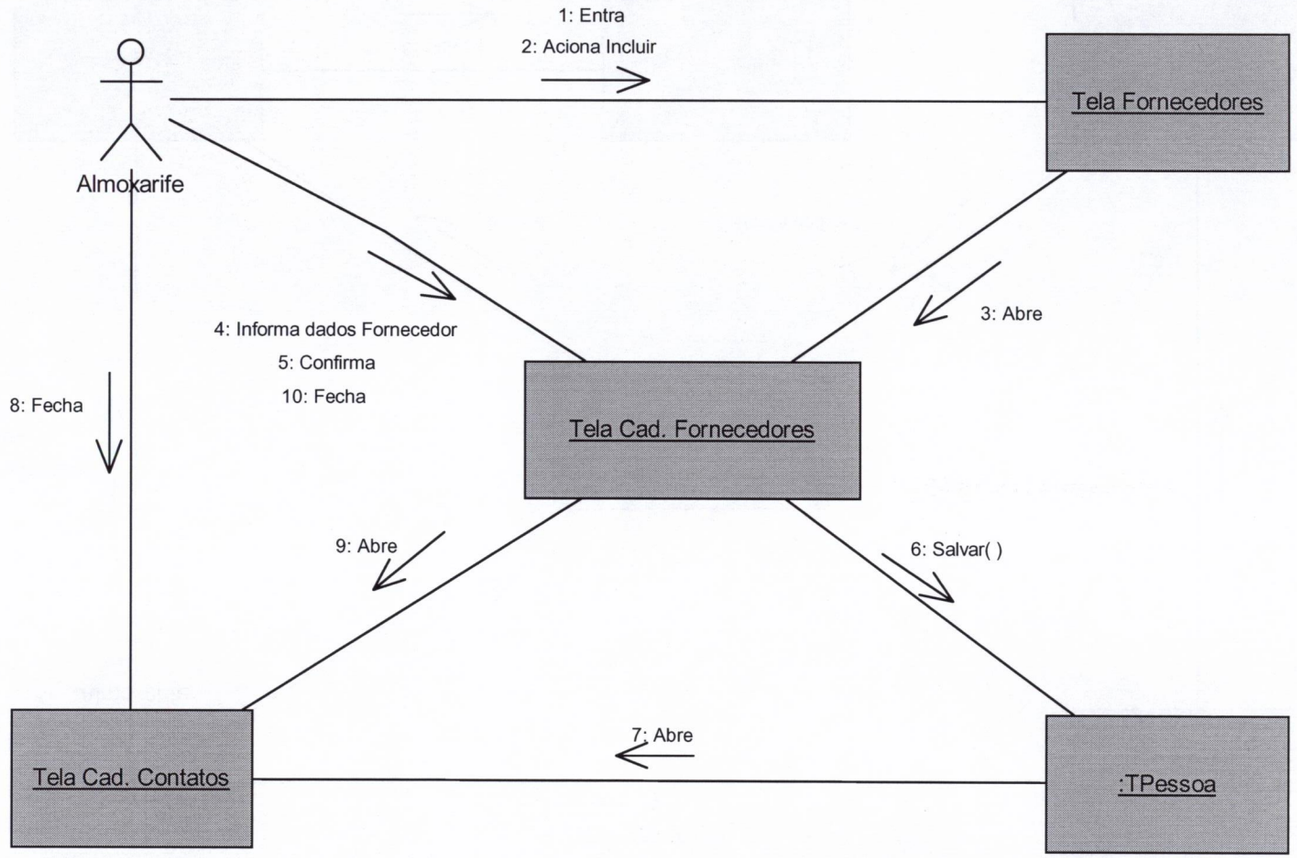


Diagrama de Colaboração - Cadastrar Contato

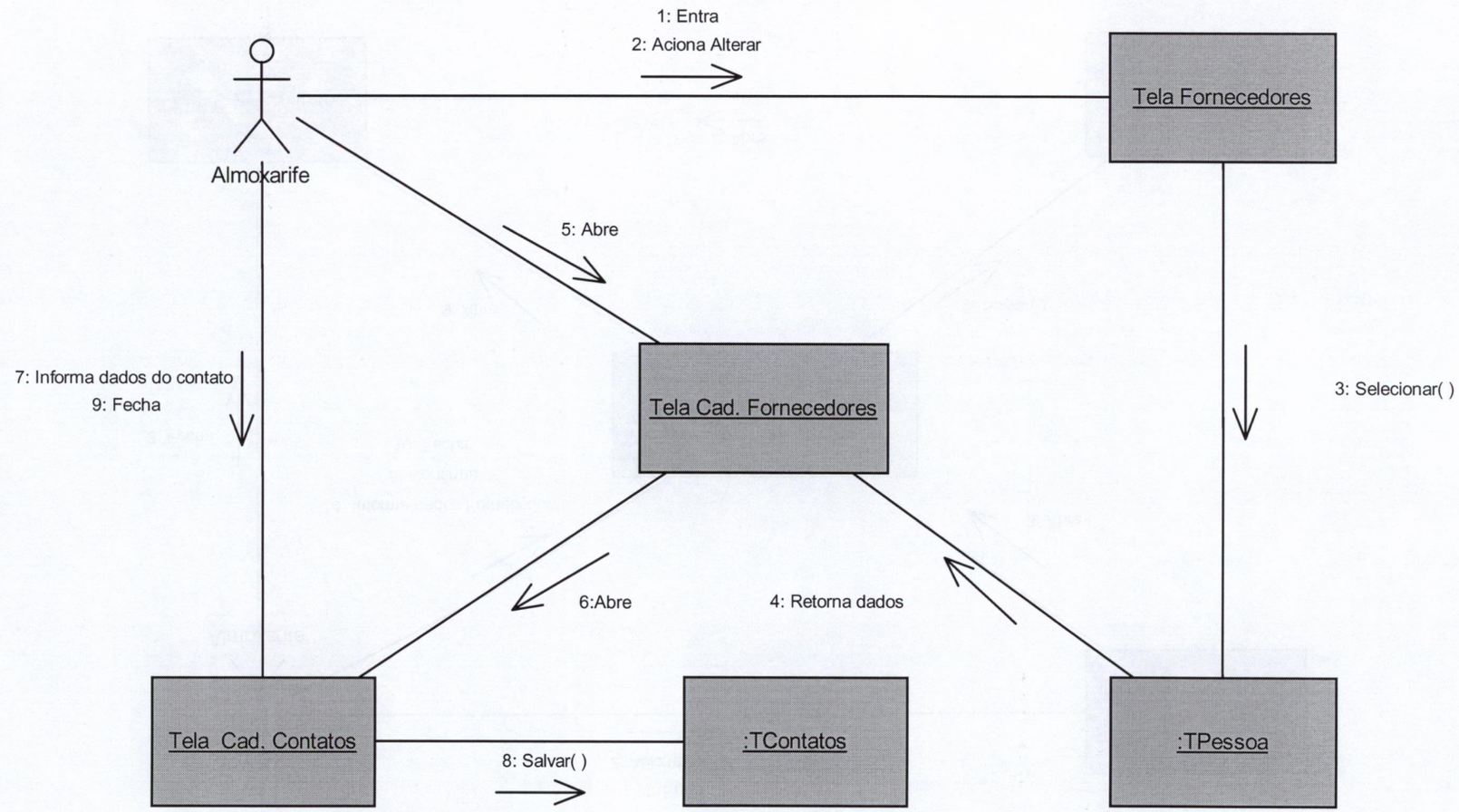


Diagrama de Colaboração - Cadastrar Telefone

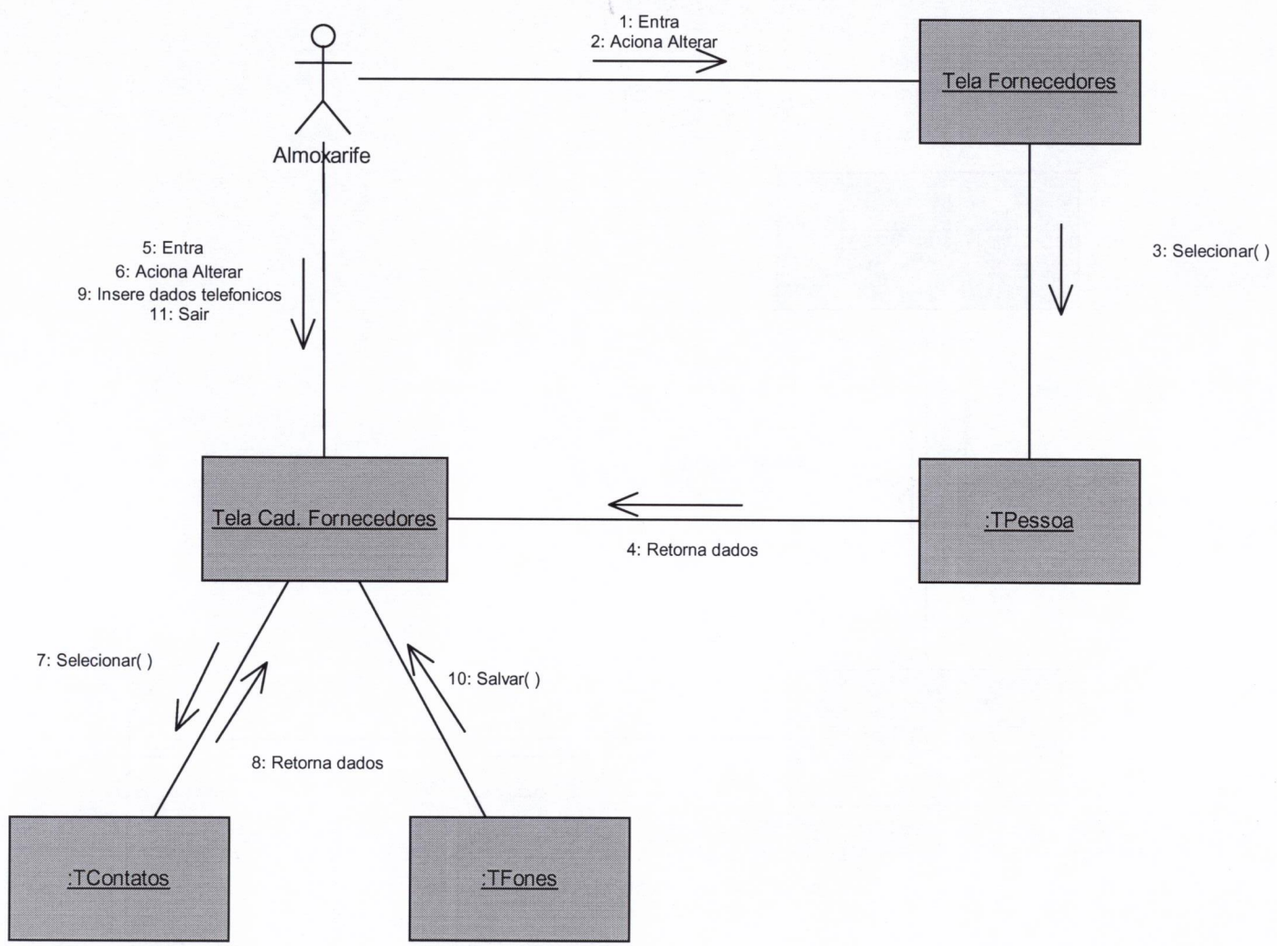


Diagrama de Colaboração - Cancelar OS

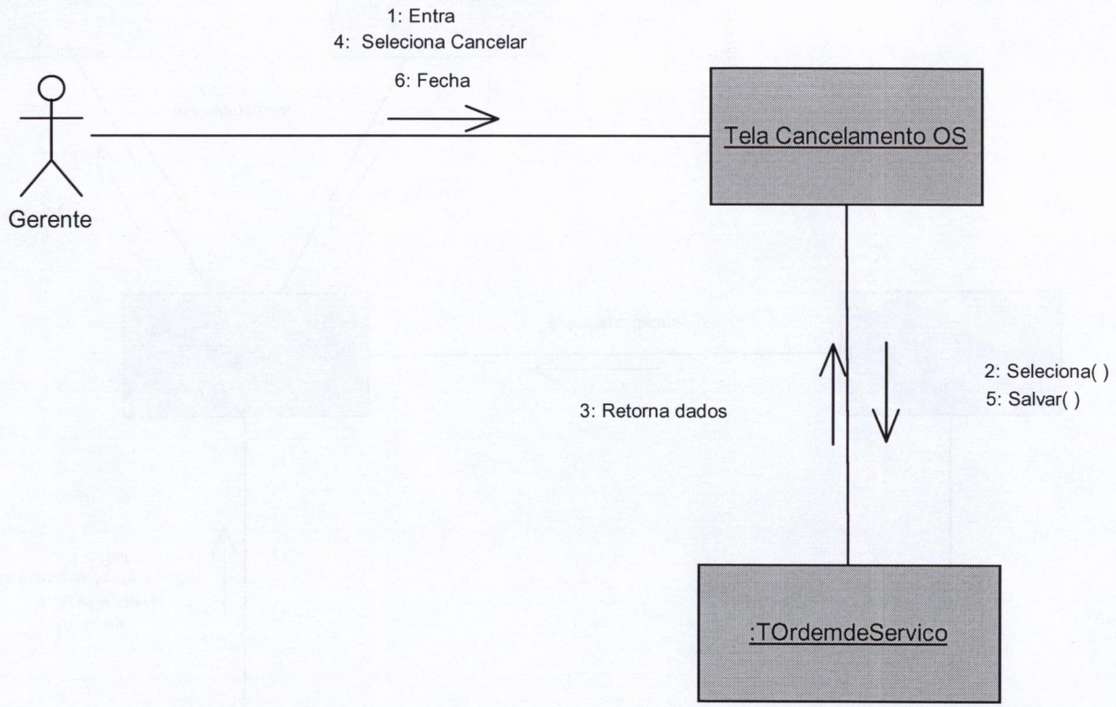


Diagrama de Colaboração - Pesquisar OS

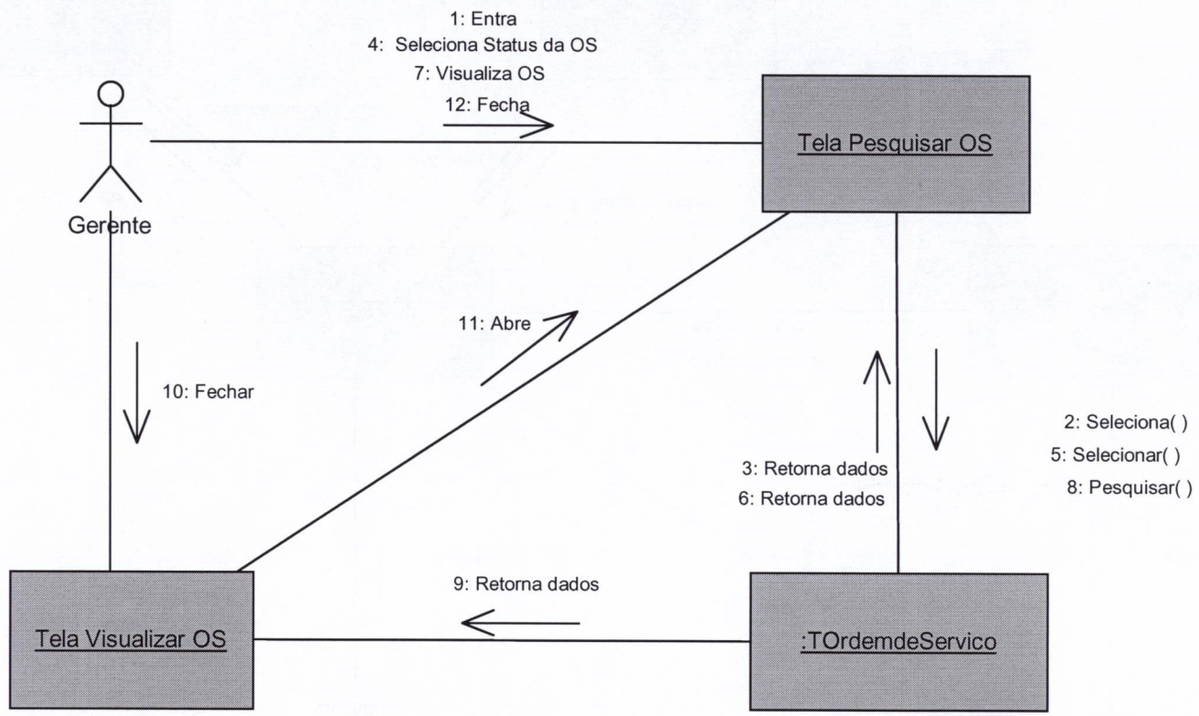


Diagrama de Colaboração - Emitir Relatório

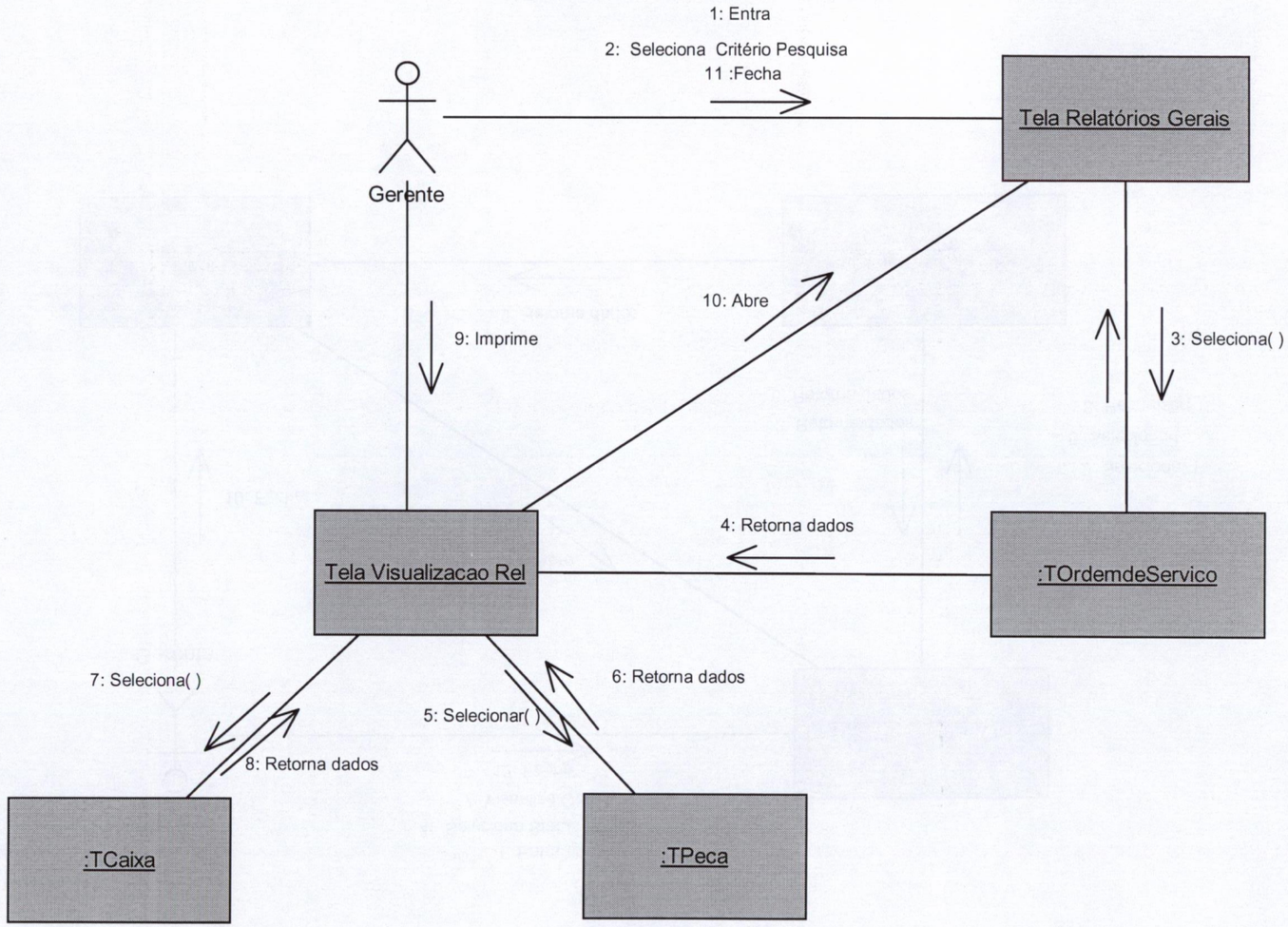


Diagrama de Colaboração - Contas à Pagar

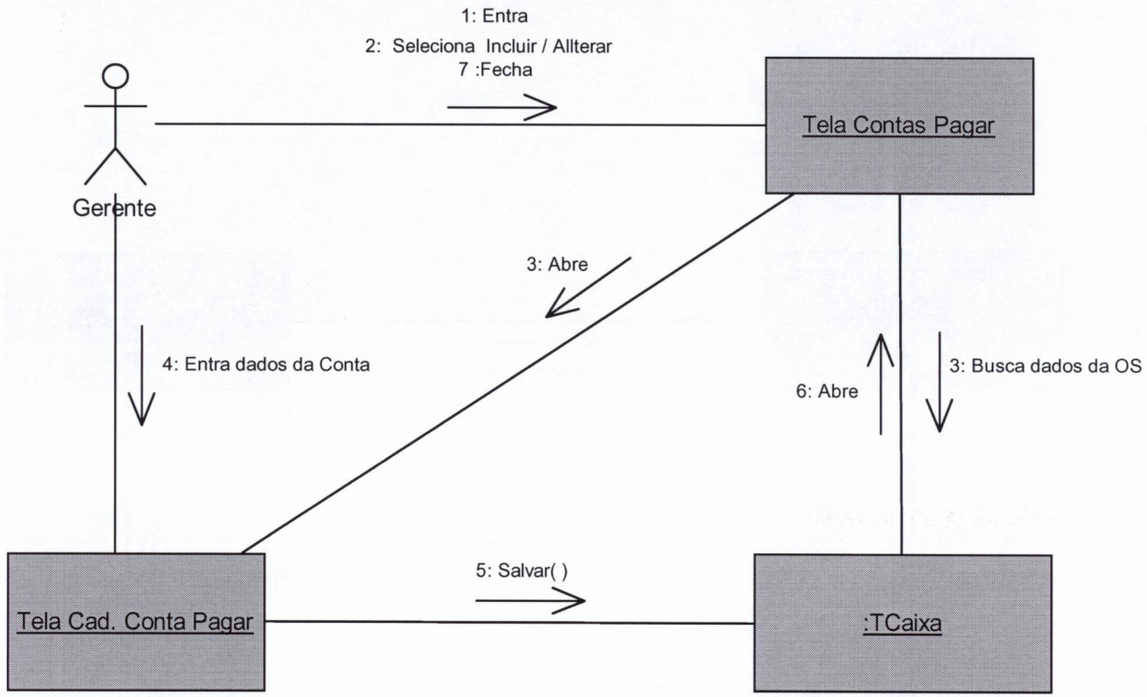


Diagrama de Colaboração - Contas à Receber

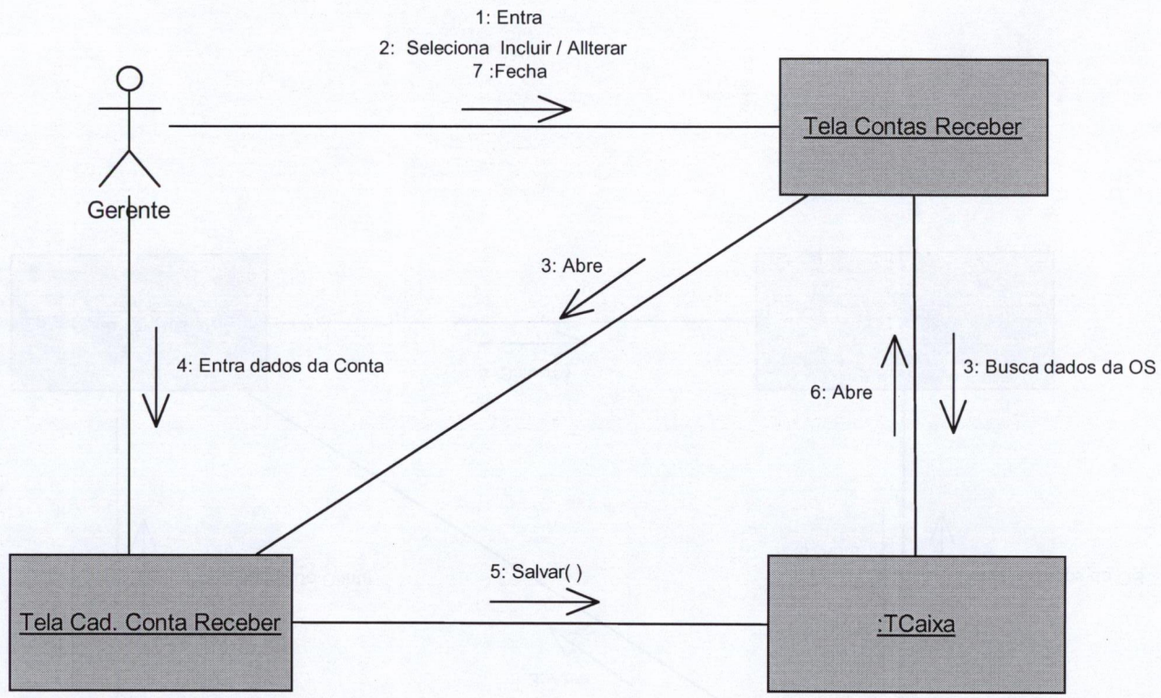


Diagrama de Colaboração - Verificar Fluxo de Caixa

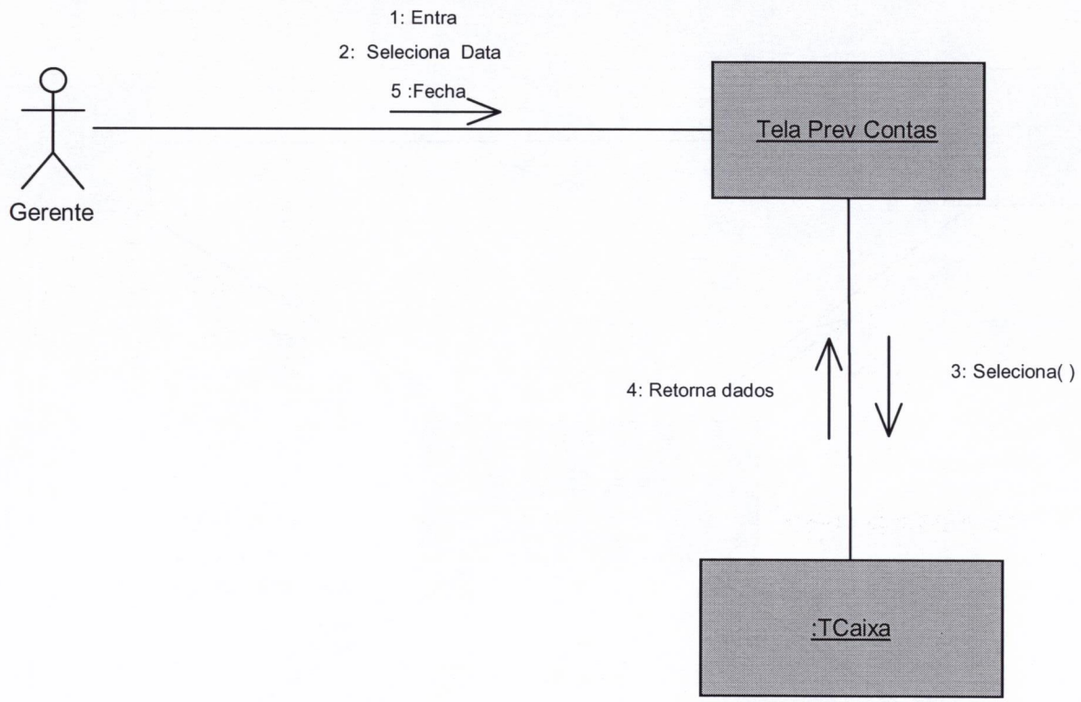


Diagrama de Colaboração - Cadastrar Cliente

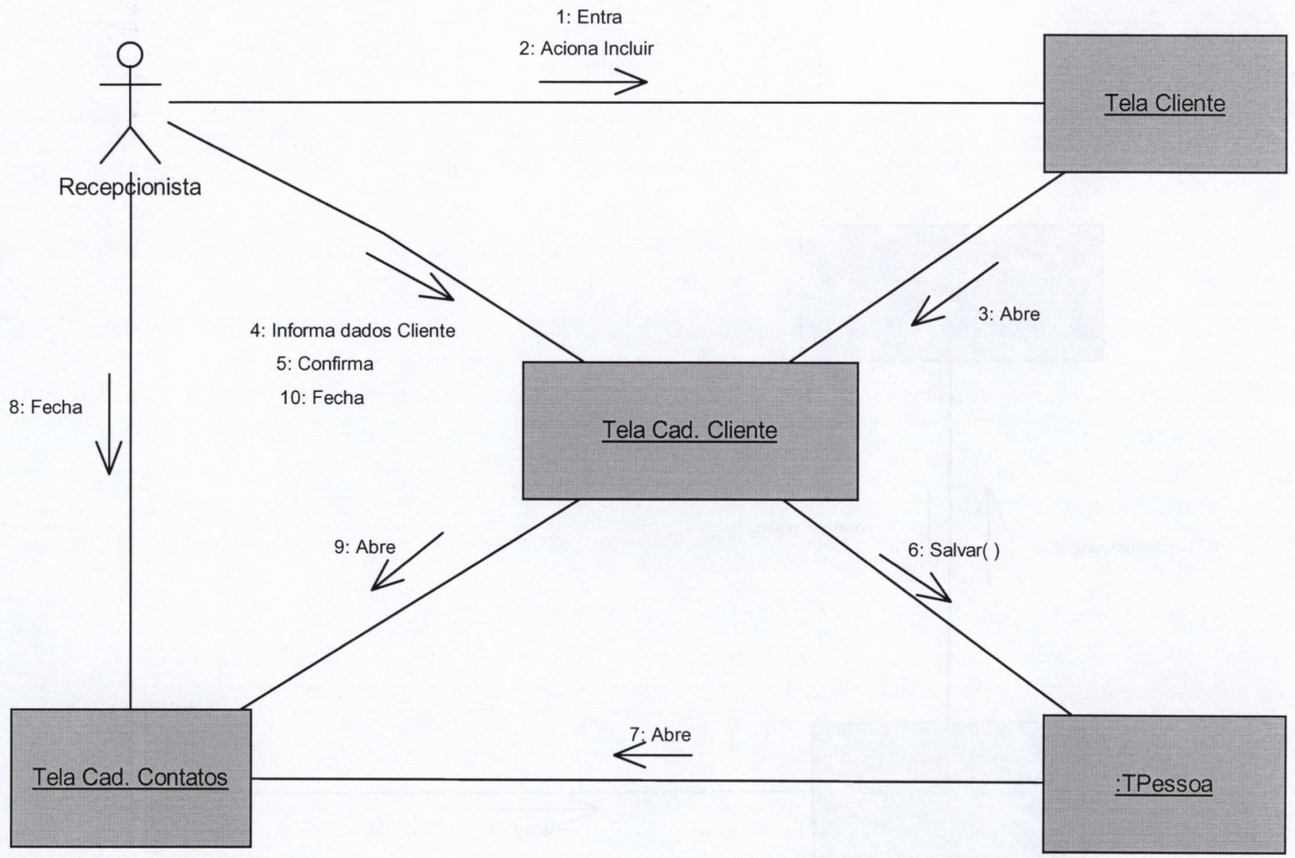


Diagrama de Colaboração - Cadastrar Contato

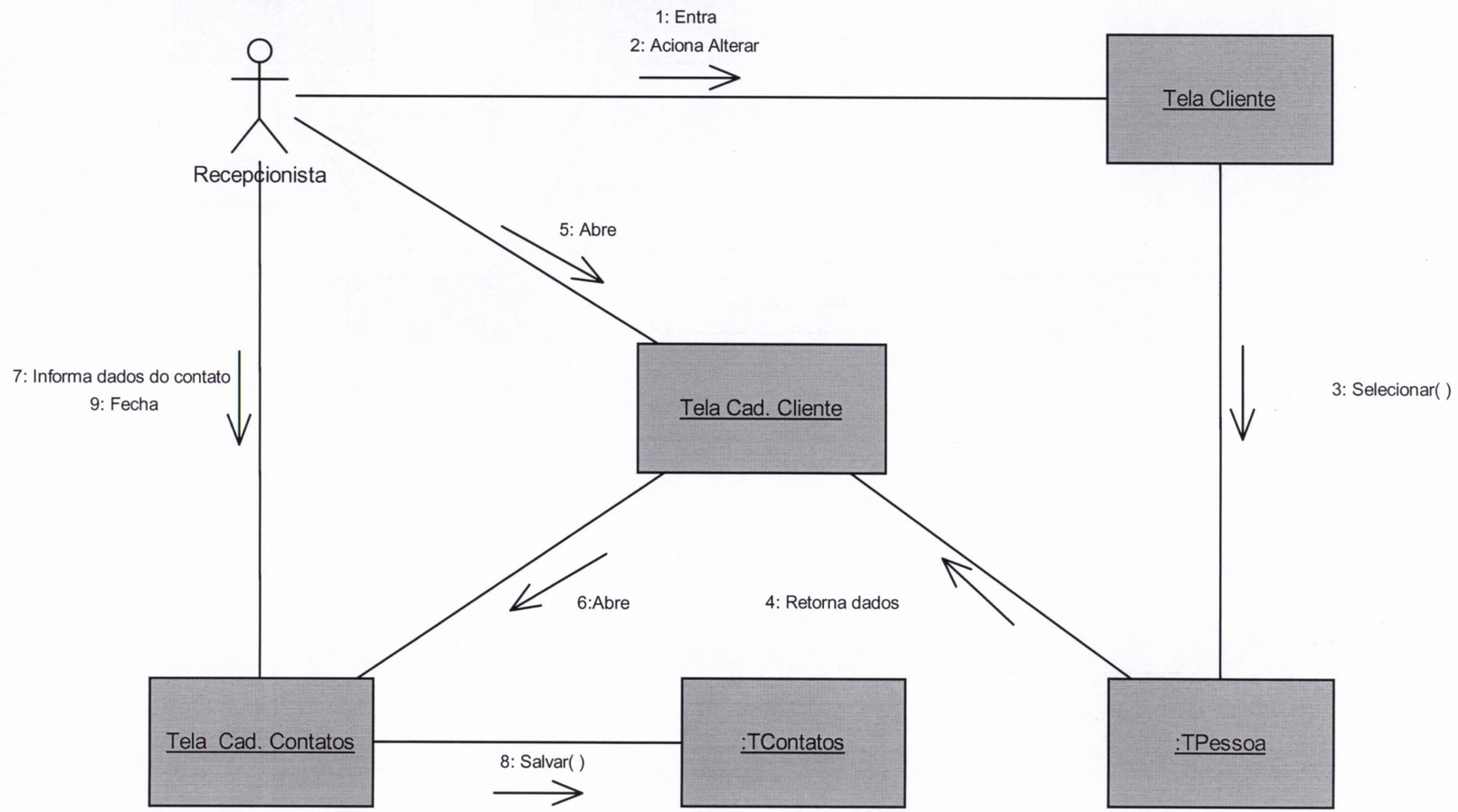


Diagrama de Colaboração - Cadastrar Telefone

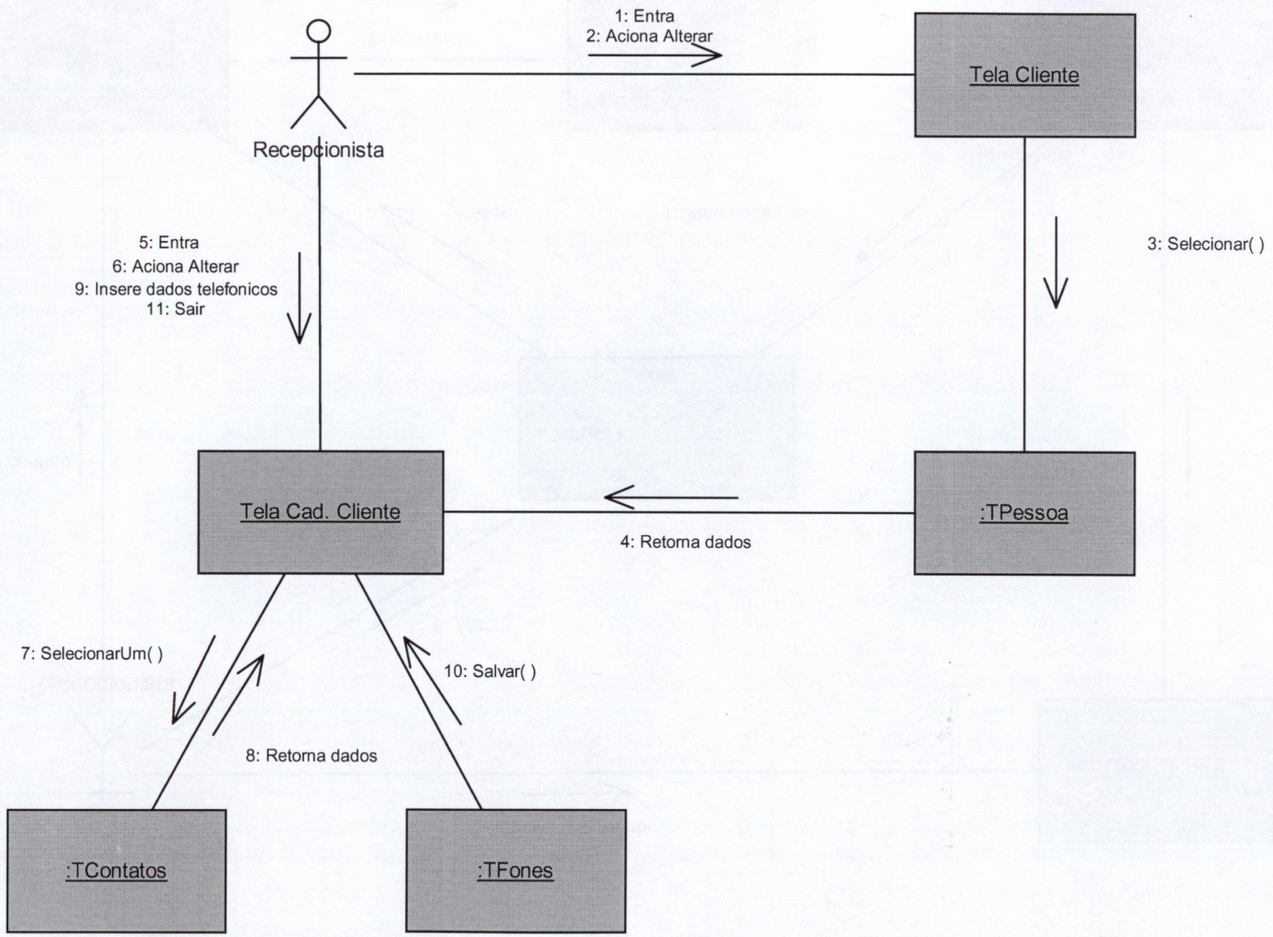


Diagrama de Colaboração - Cadastrar Contrato

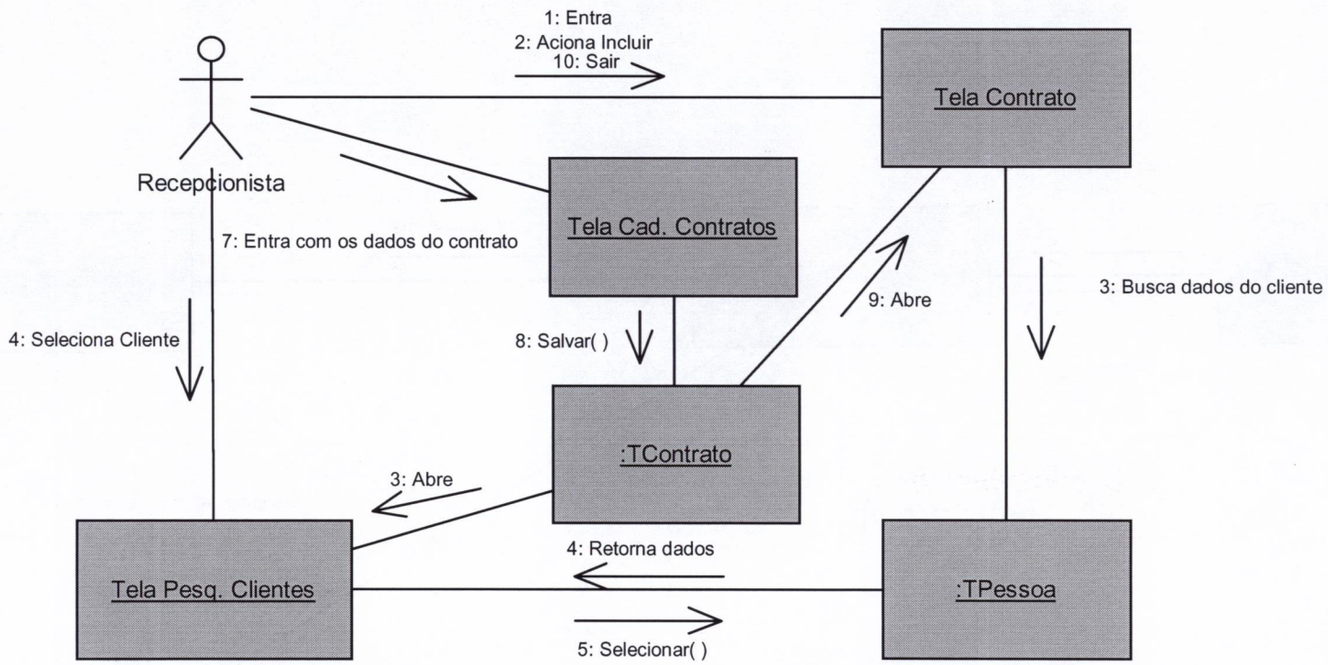


Diagrama de Colaboração - Cadastrar Equipamentos por Contrato

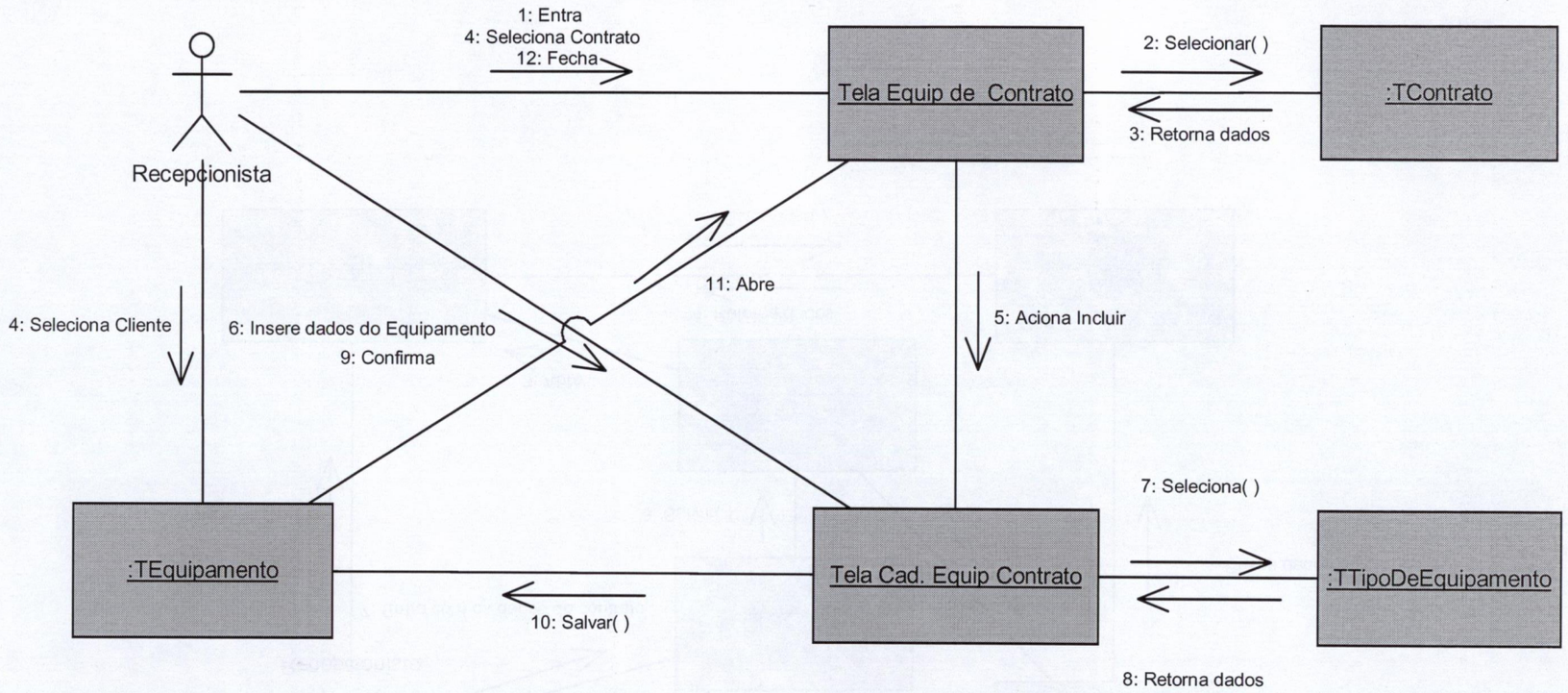


Diagrama de Colaboração - Cadastrar tipo de Equipamento

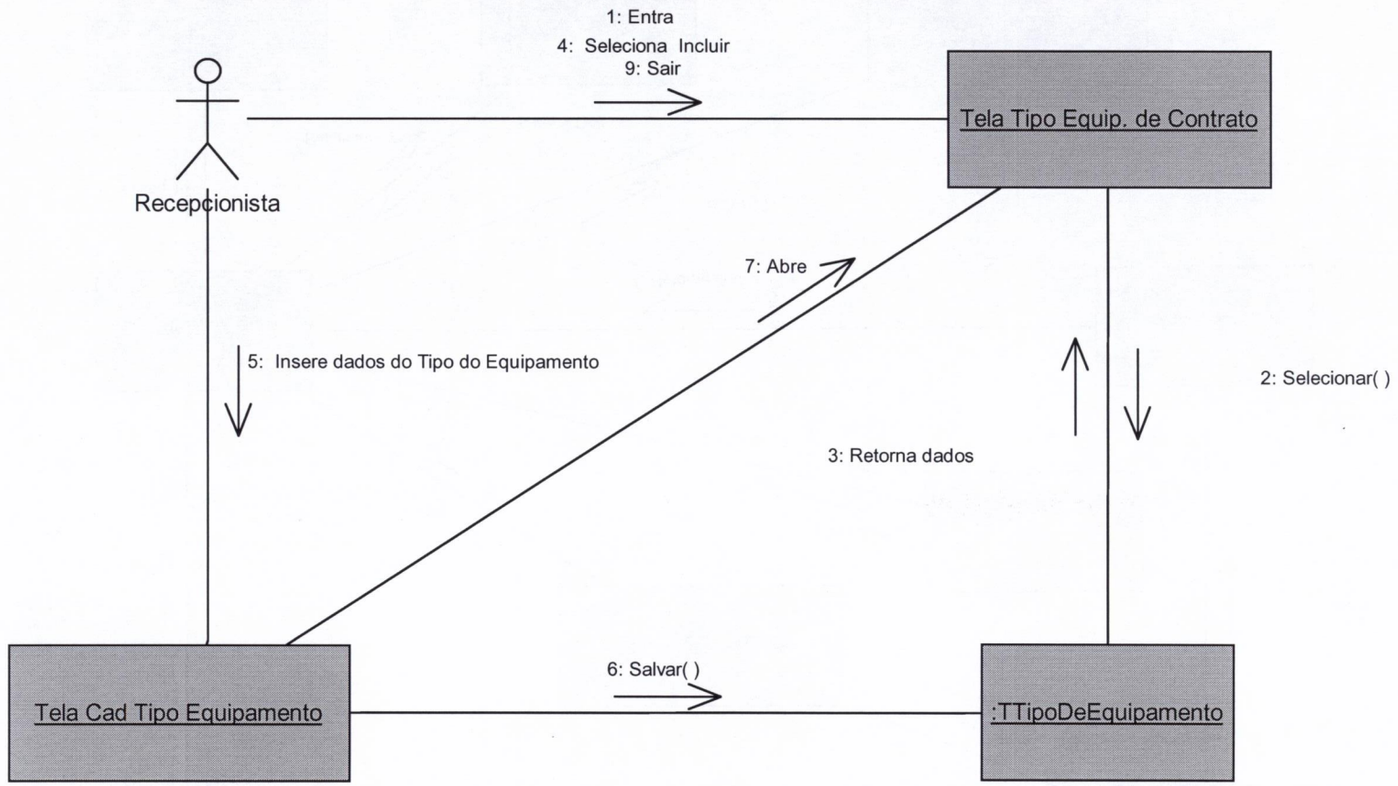


Diagrama de Colaboração - Abrir Orçamento

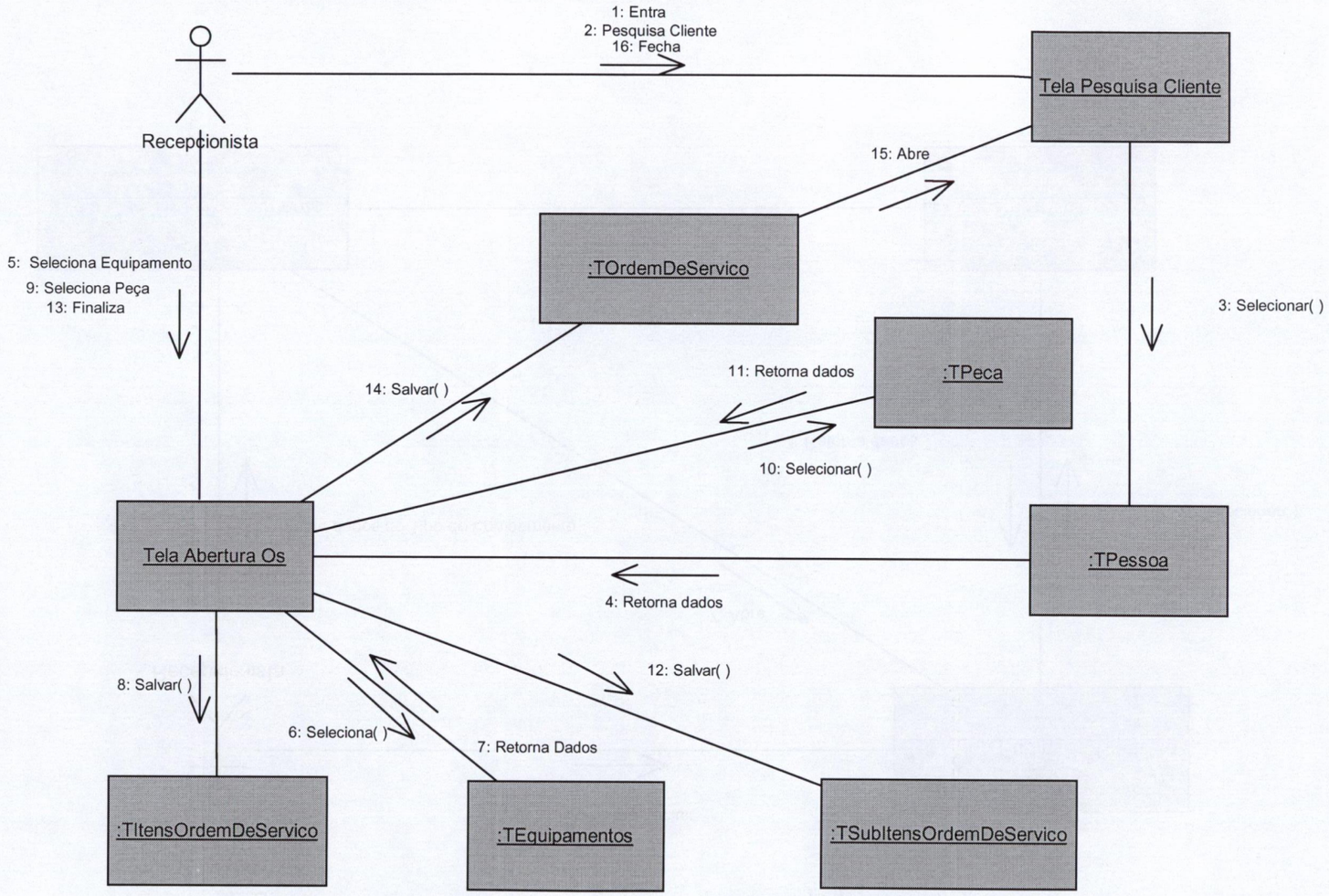


Diagrama de Colaboração - Alterar / Liberar para Orçar

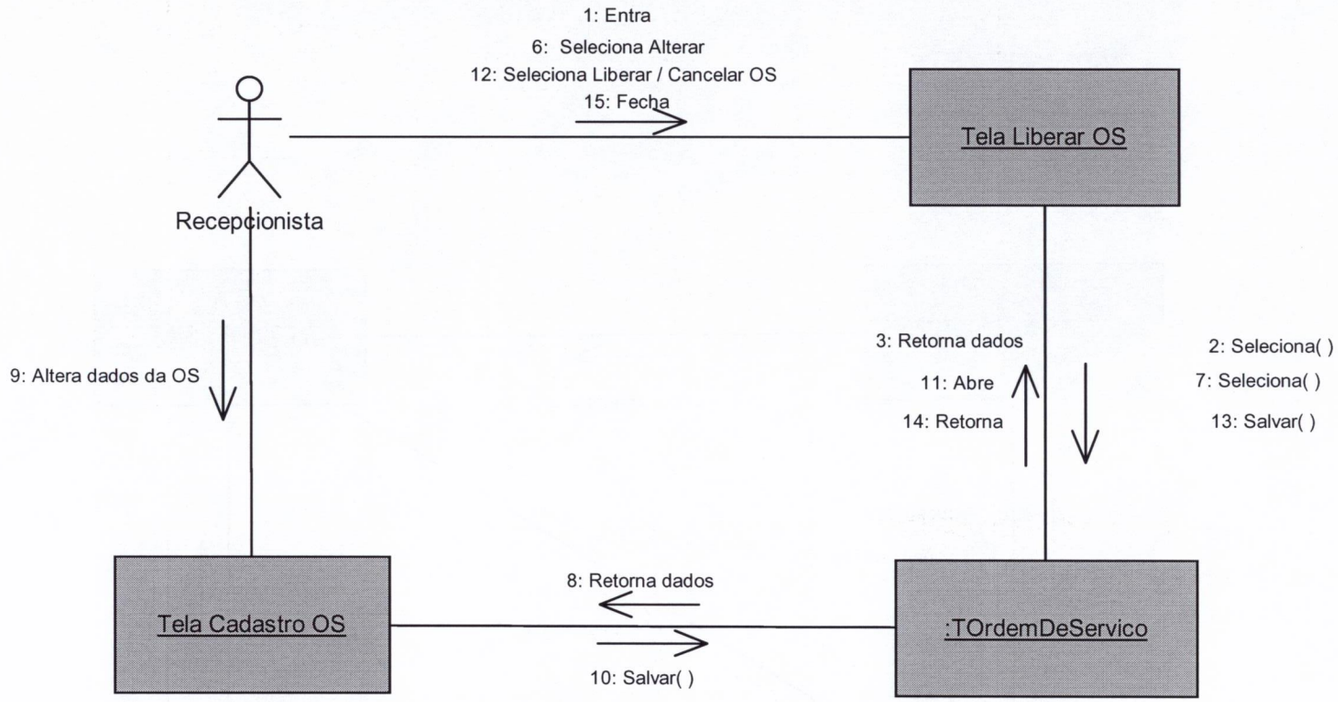


Diagrama de Colaboração - Aprovar Orçamento

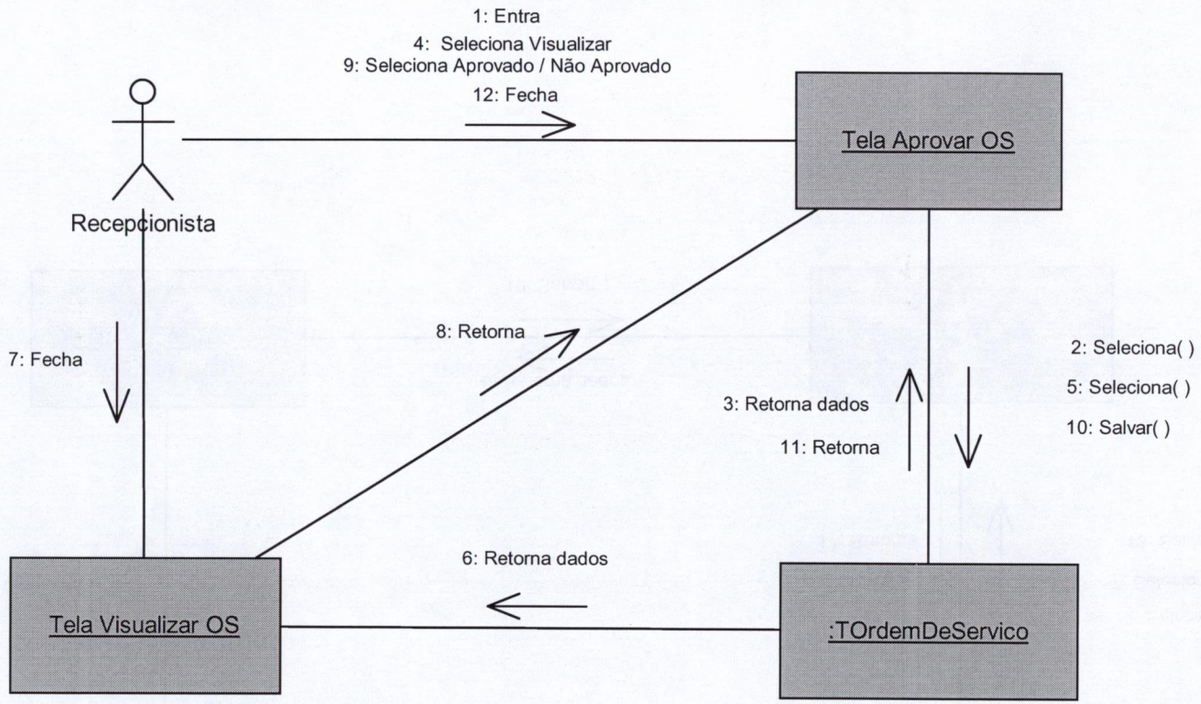


Diagrama de Colaboração - Fechar OS

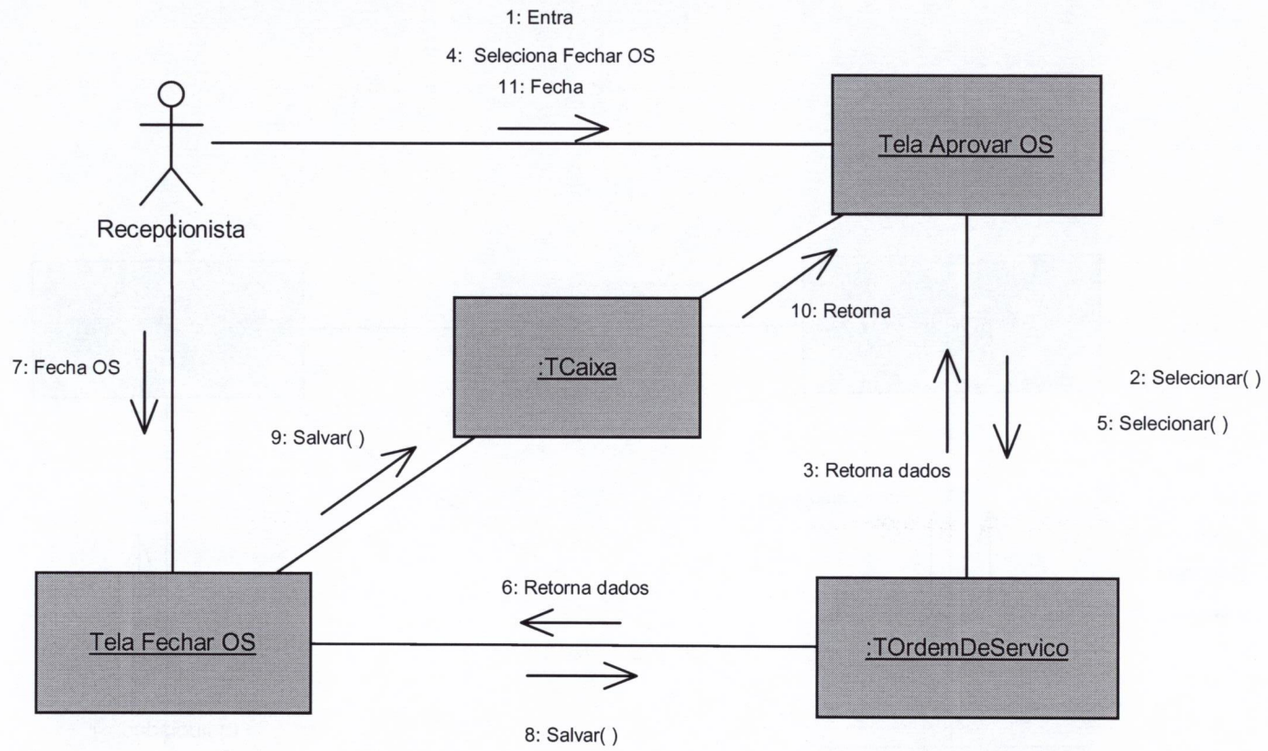


Diagrama de Colaboração - Pesquisar OS

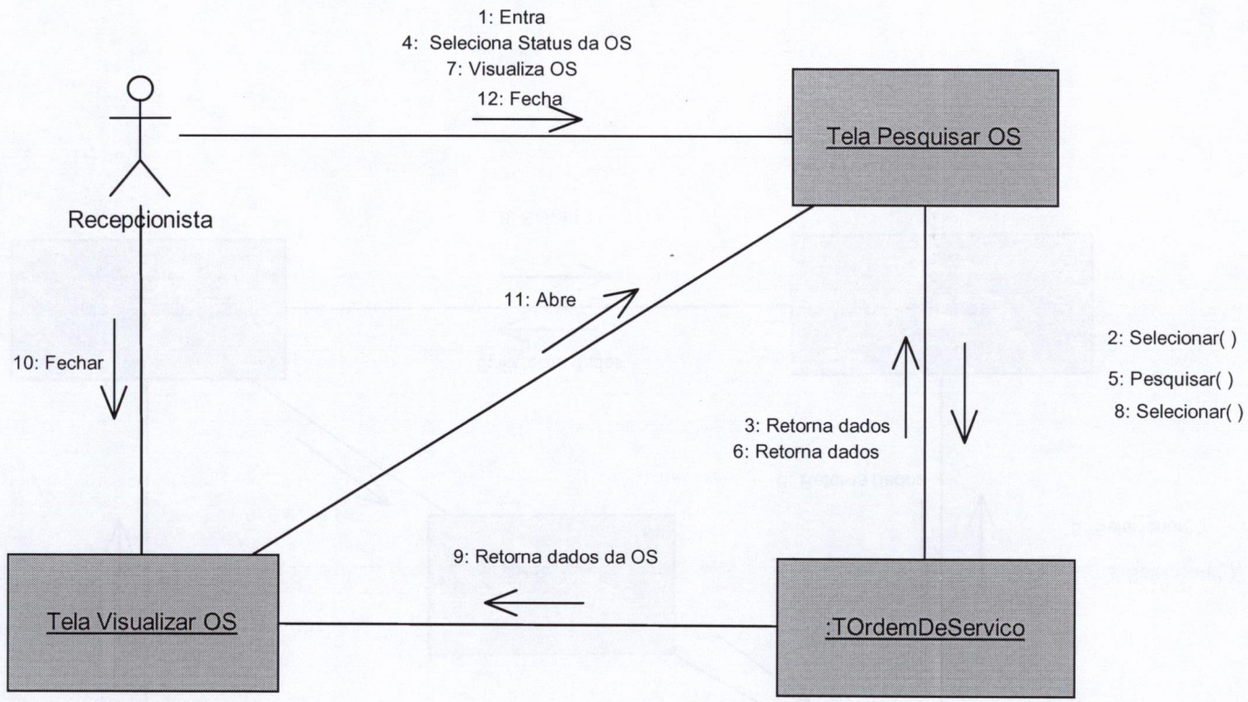


Diagrama de Colaboração - Efetuar Orçamento

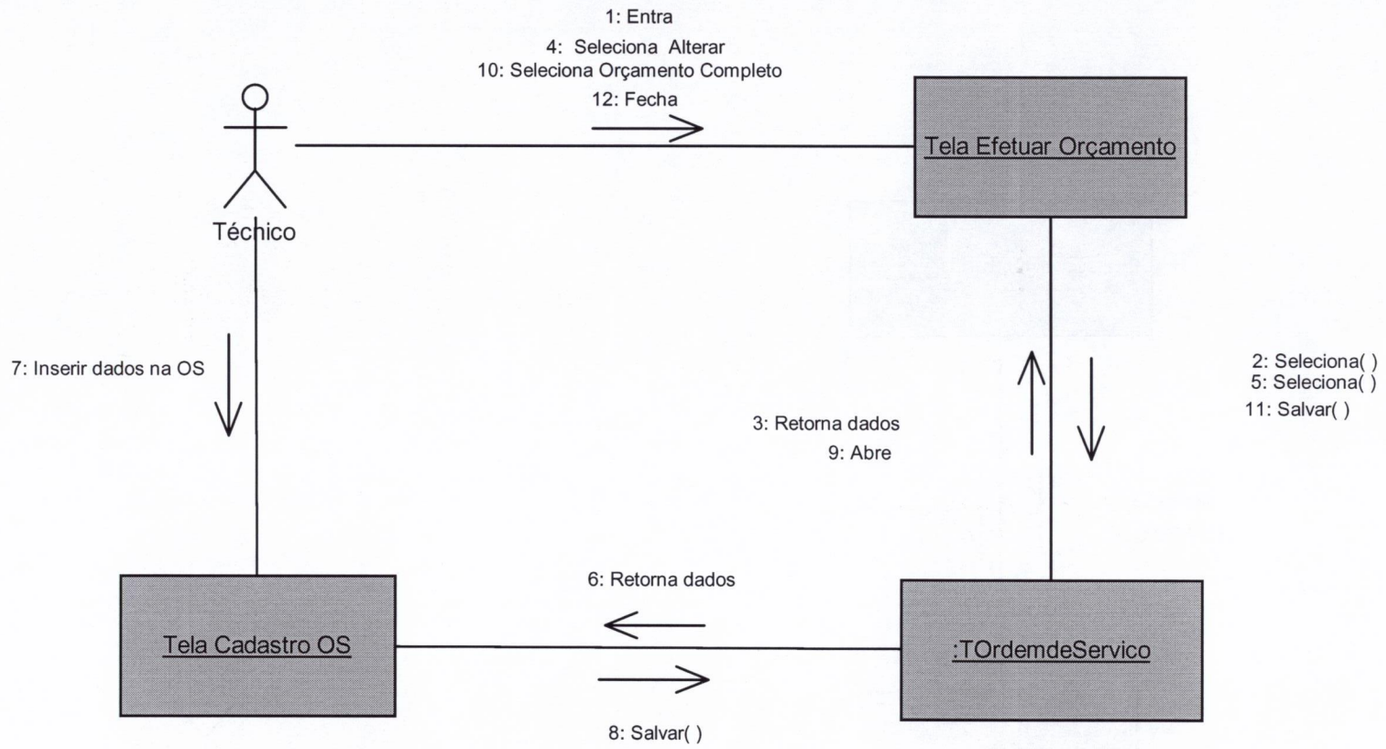
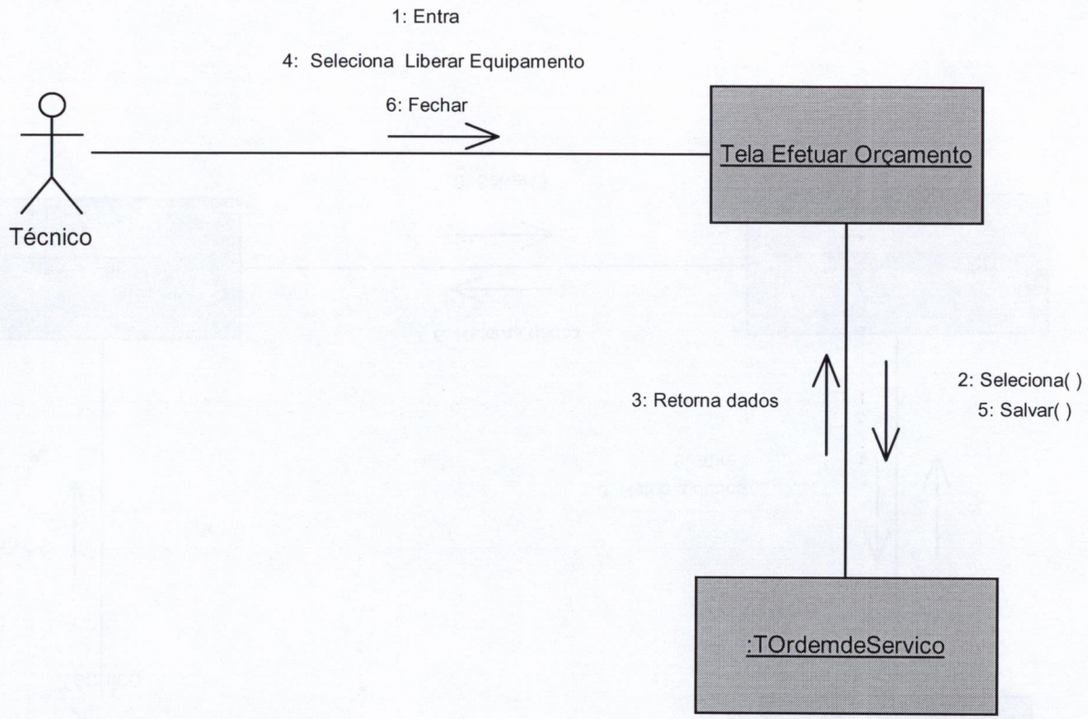
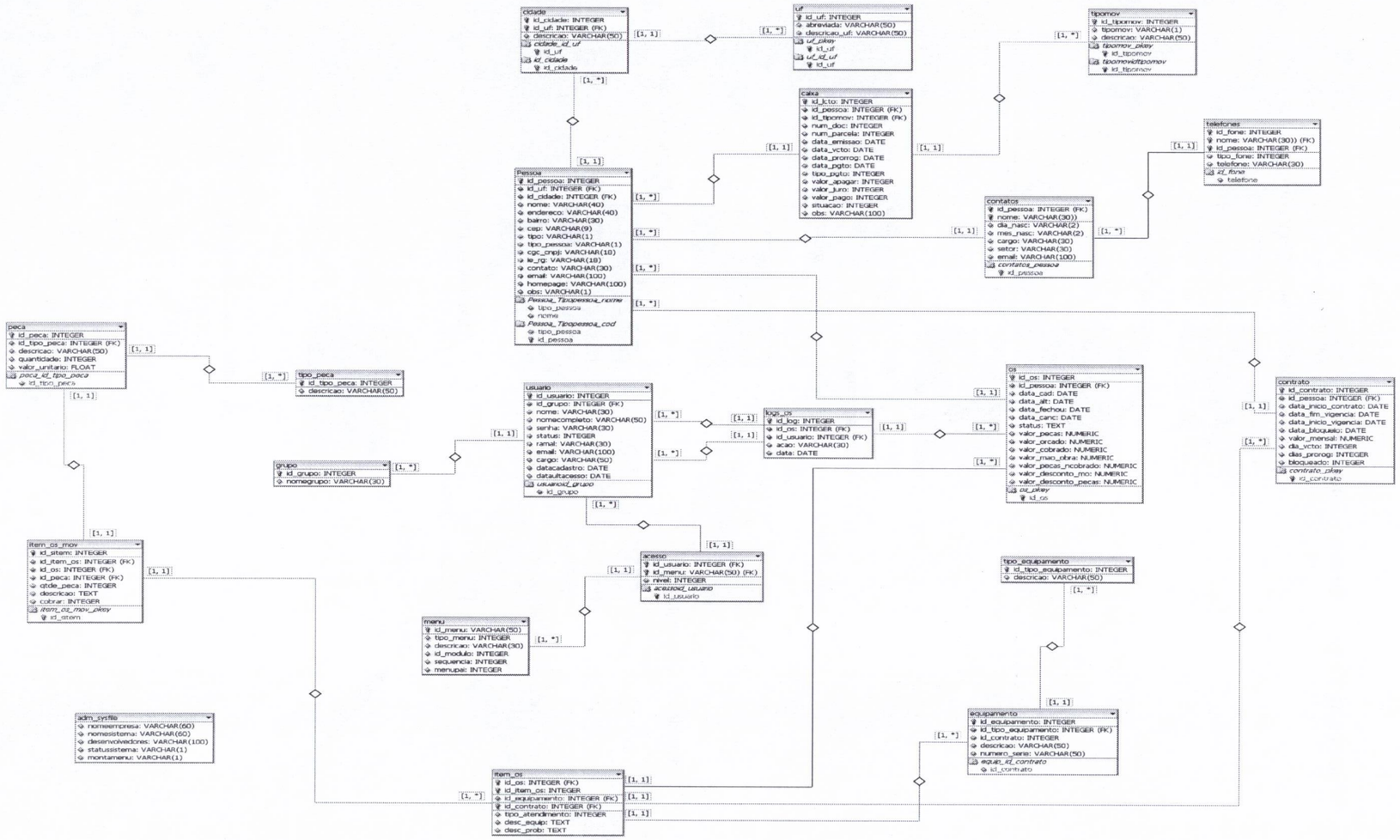


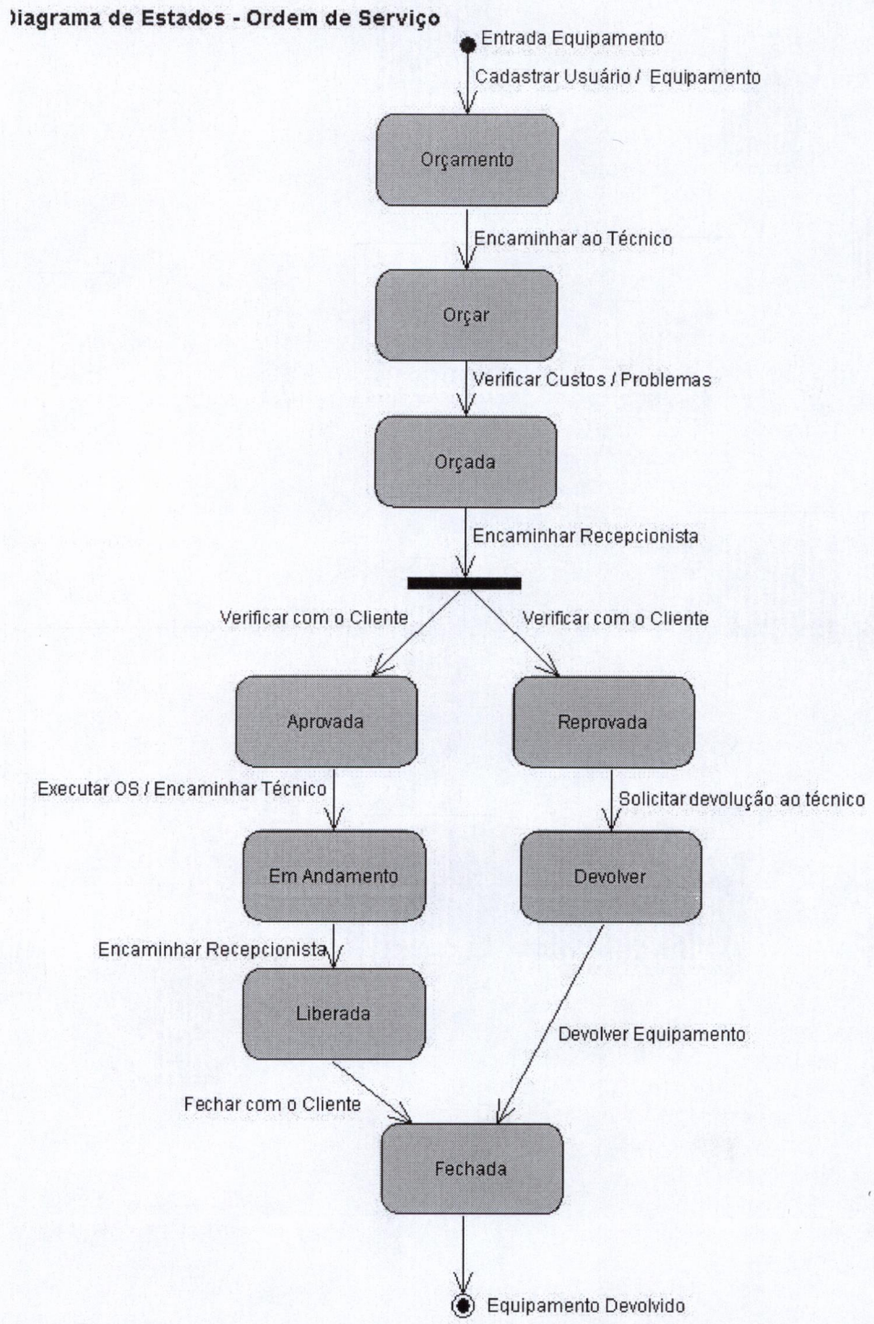
Diagrama de Colaboração - Orçamentos Não Aprovados



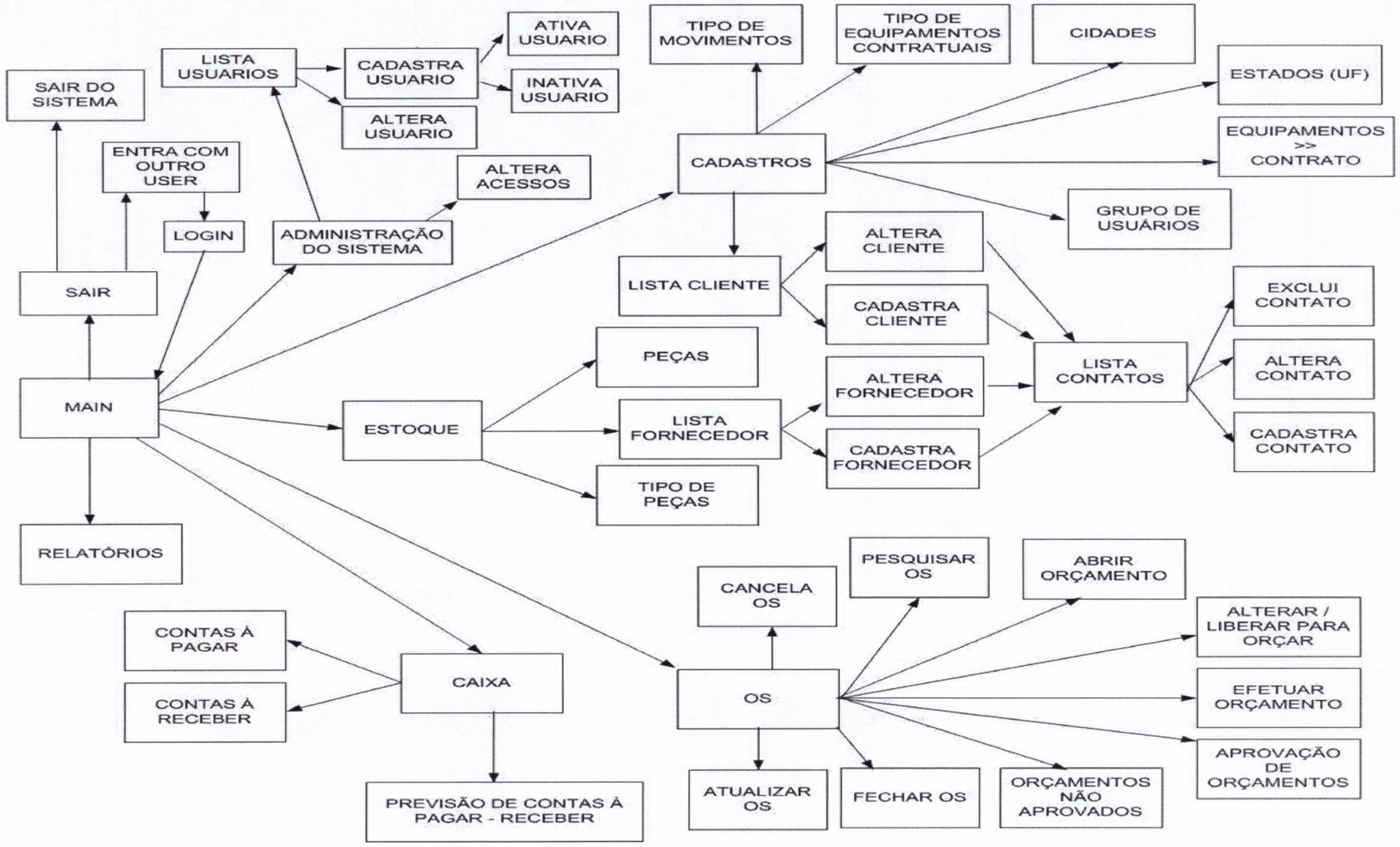
11 DIAGRAMA DE ENTIDADE RELACIONAMENTO



12 DIAGRAMA DE ESTADOS

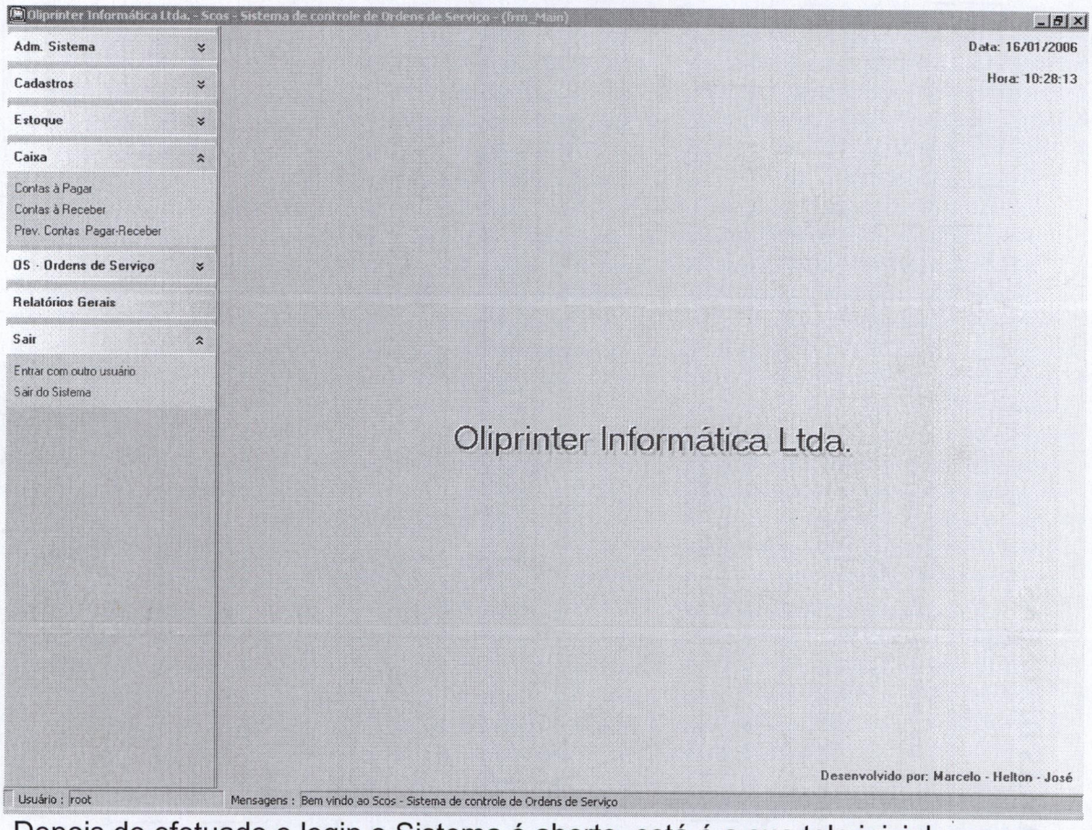


13 DIAGRAMAS DE TELAS

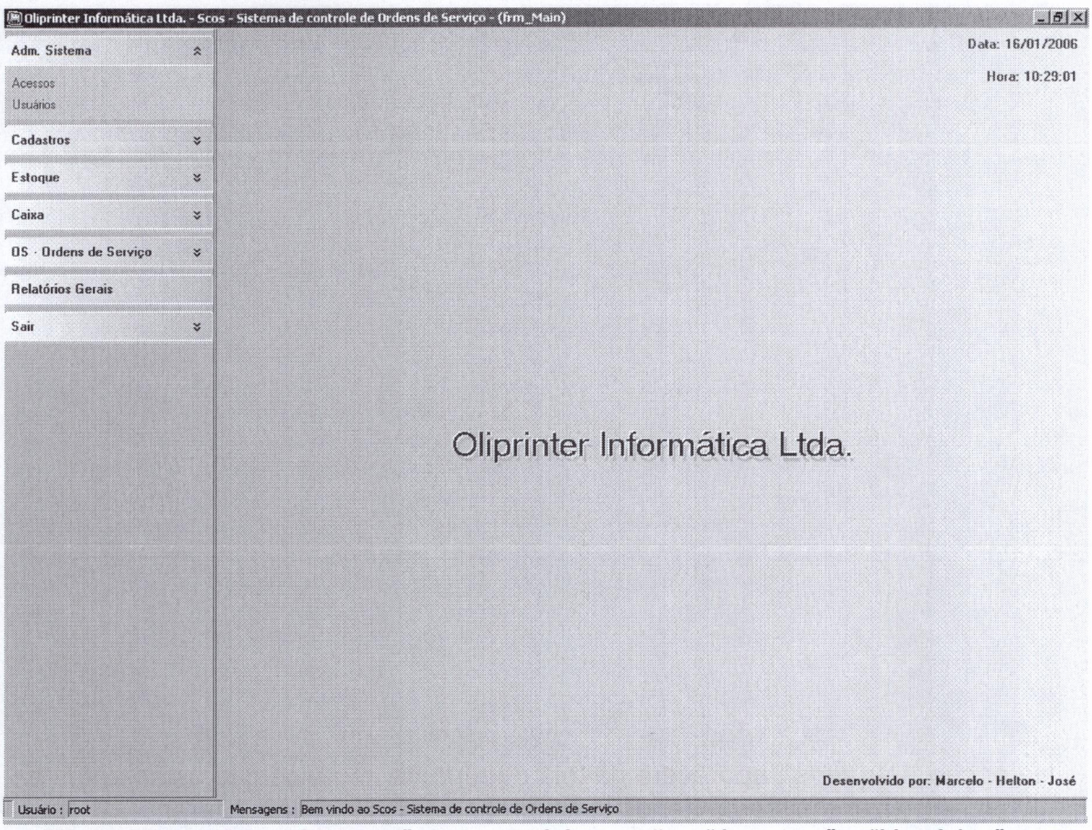


14 TELAS DO SISTEMA

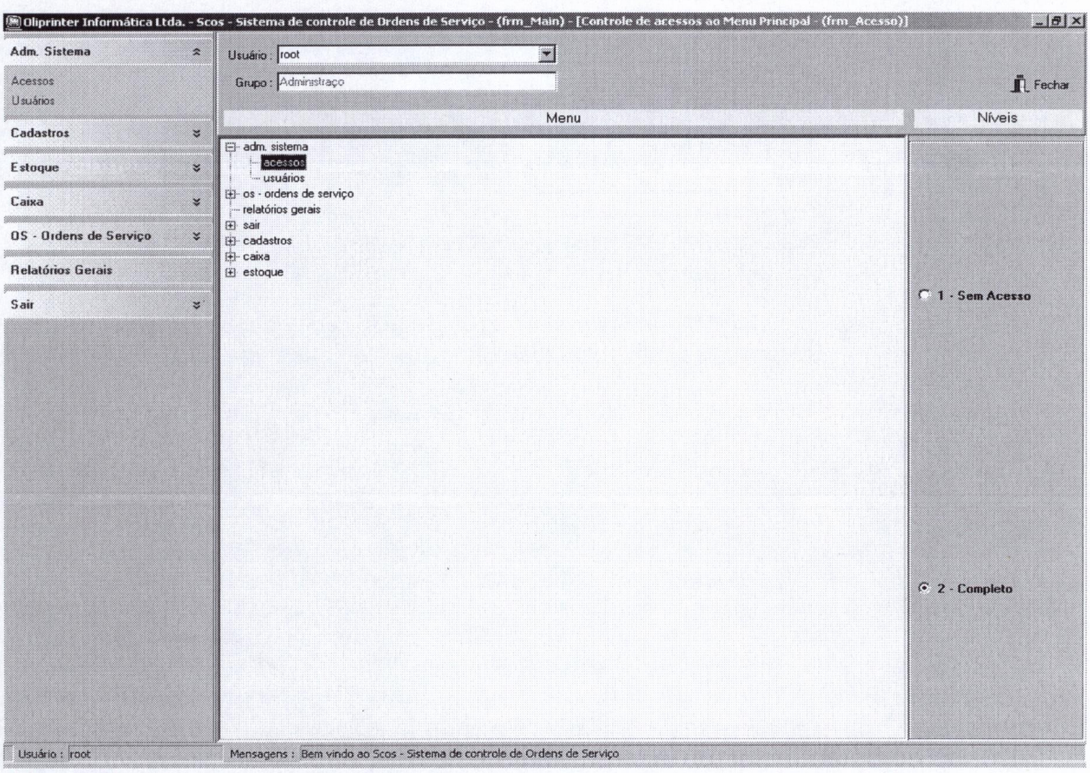
Tela de abertura, nesta tela o usuário entra com seu login e senha para acessar o Sistema.



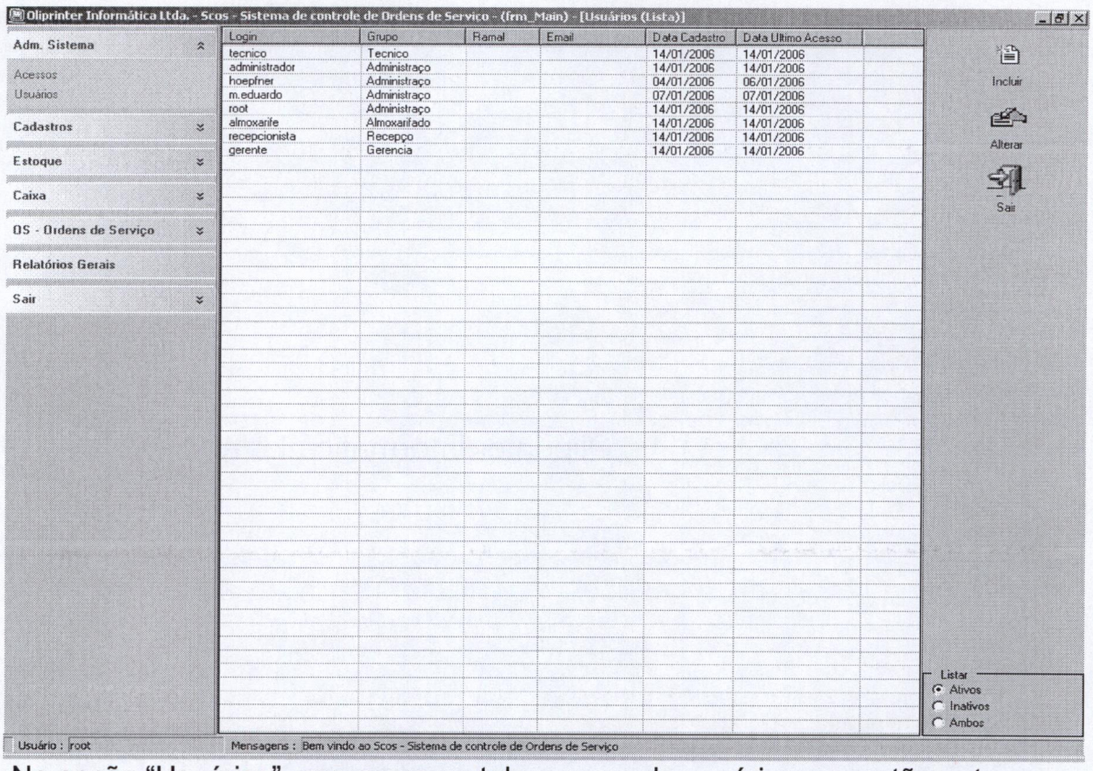
Depois de efetuado o login o Sistema é aberto, está é a sua tela inicial com as opções ao lado para o usuário começar a trabalhar.



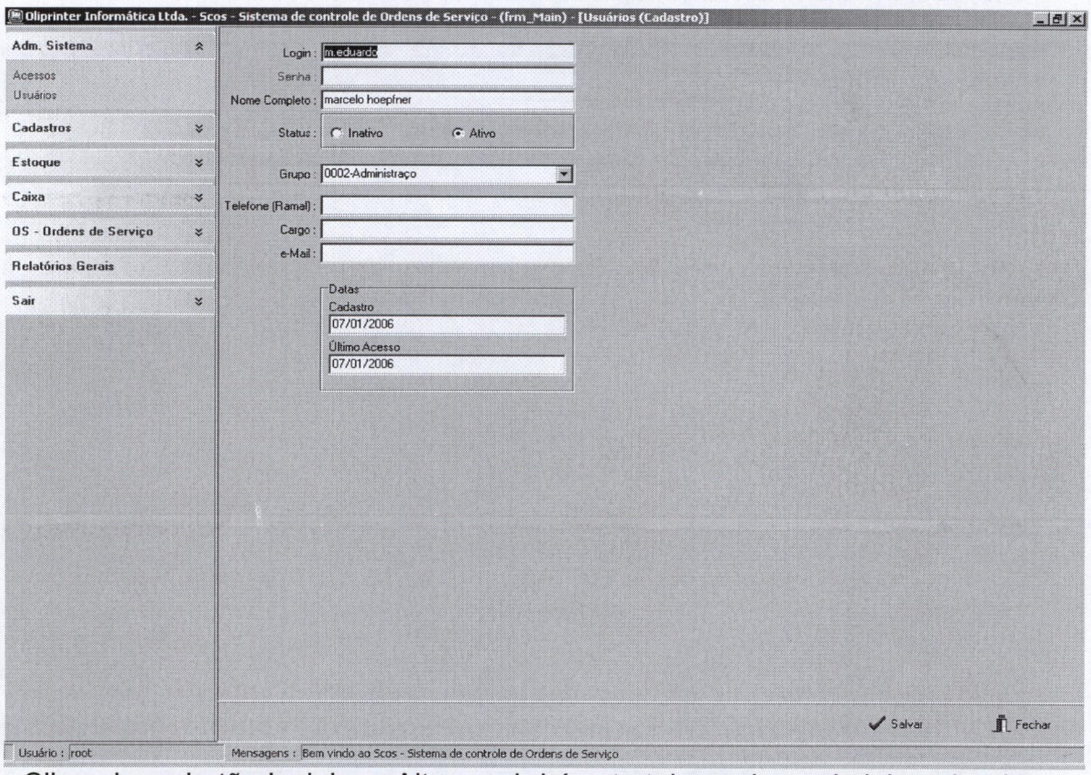
Clicando em “Adm. Sistema” aparecerá às opções “Acessos” e “Usuários”.



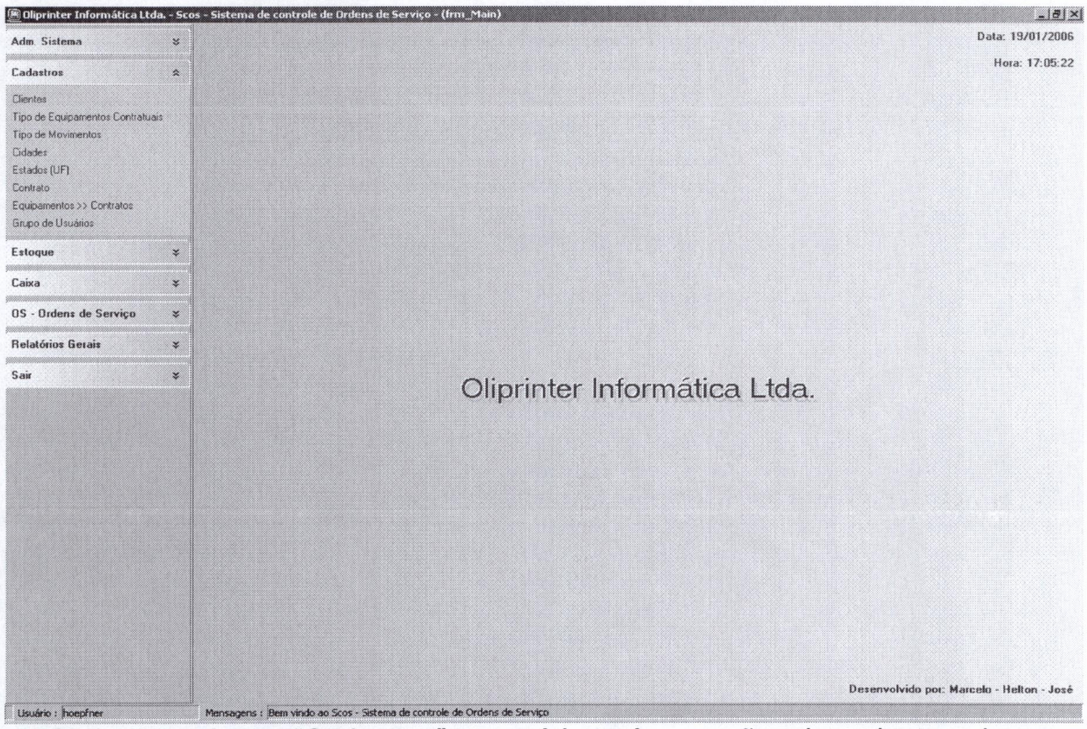
Na opção “Acessos pode-se visualizar os acessos que cada usuário possui no sistema.



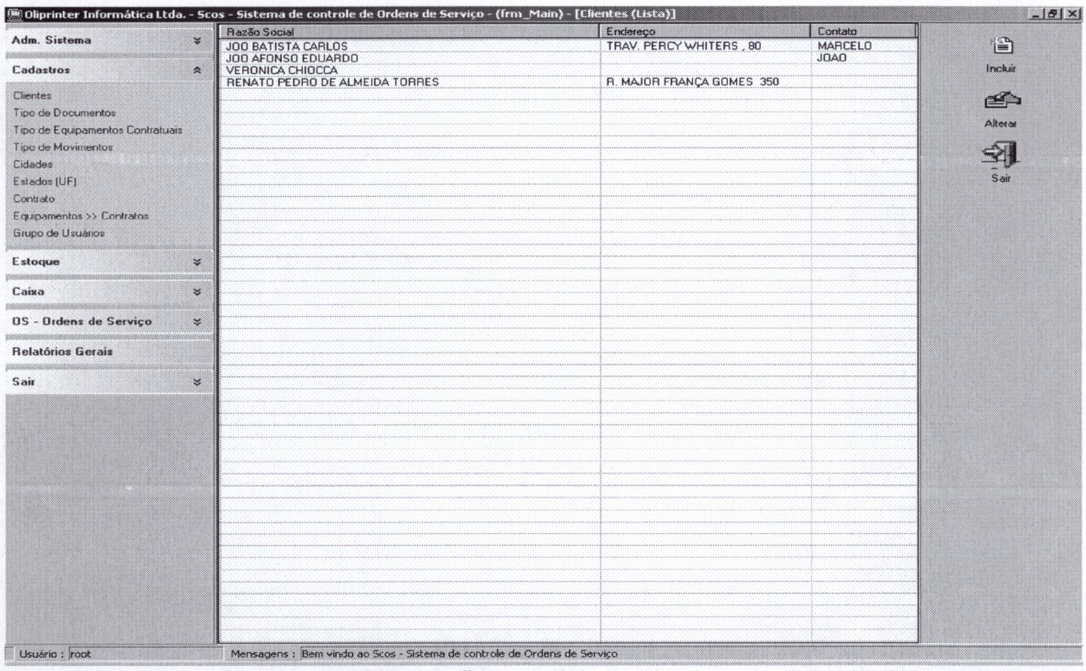
Na opção “Usuários”, aparecera na tela o grupo de usuários que estão aptos a utilizar o sistema, tendo também os botões Incluir e Alterar.



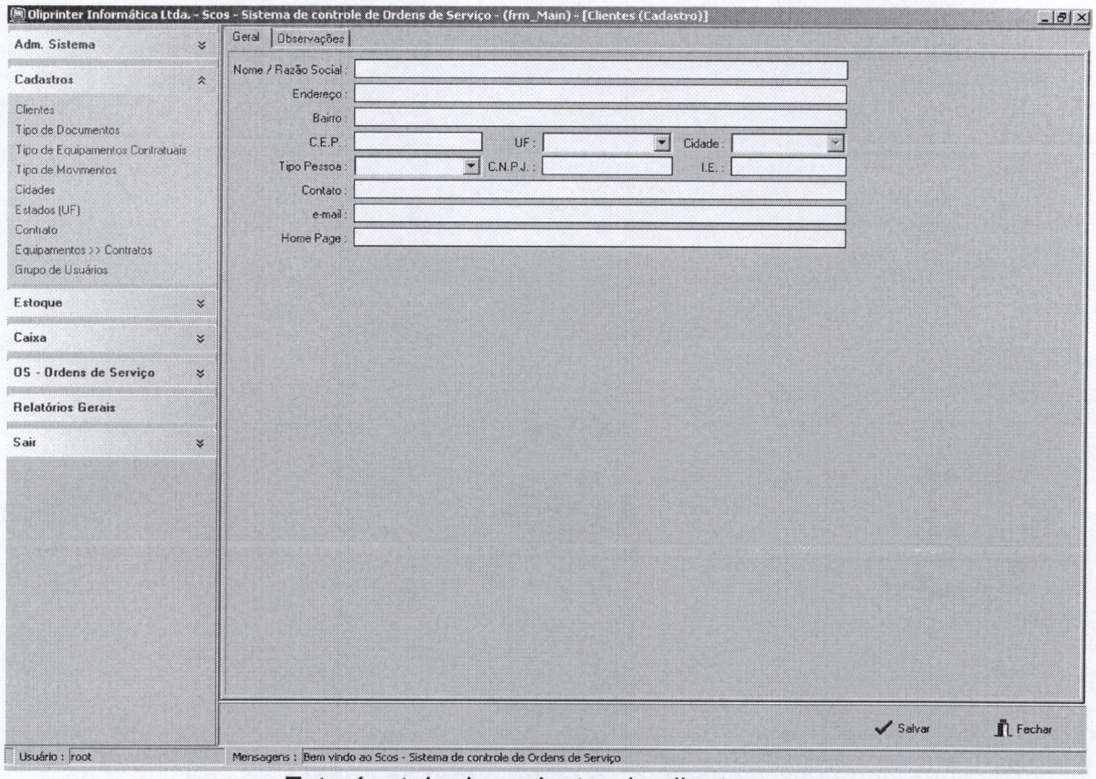
Clicando no botão Incluir ou Alterar, abrirá esta tela, onde o administrador do sistema pode incluir um novo usuário ou alterar os dados de um usuário já cadastrado.



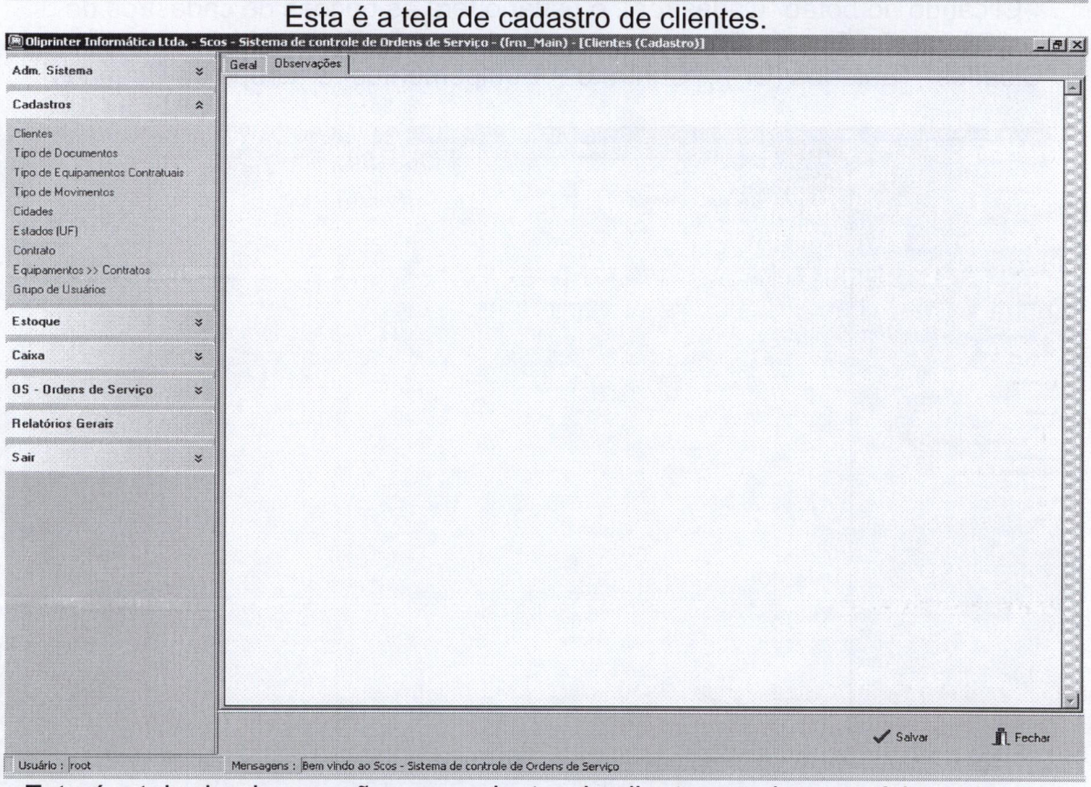
Clicando no botão “Cadastros”, o usuário terá as opções de cadastros de “Clientes”, “Tipo de Equipamentos Contratuais”, “Tipo de Movimentos”, “Cidades”, “Estados (UF)”, “Contrato”, “Equipamentos” e “Grupo de Usuários”.



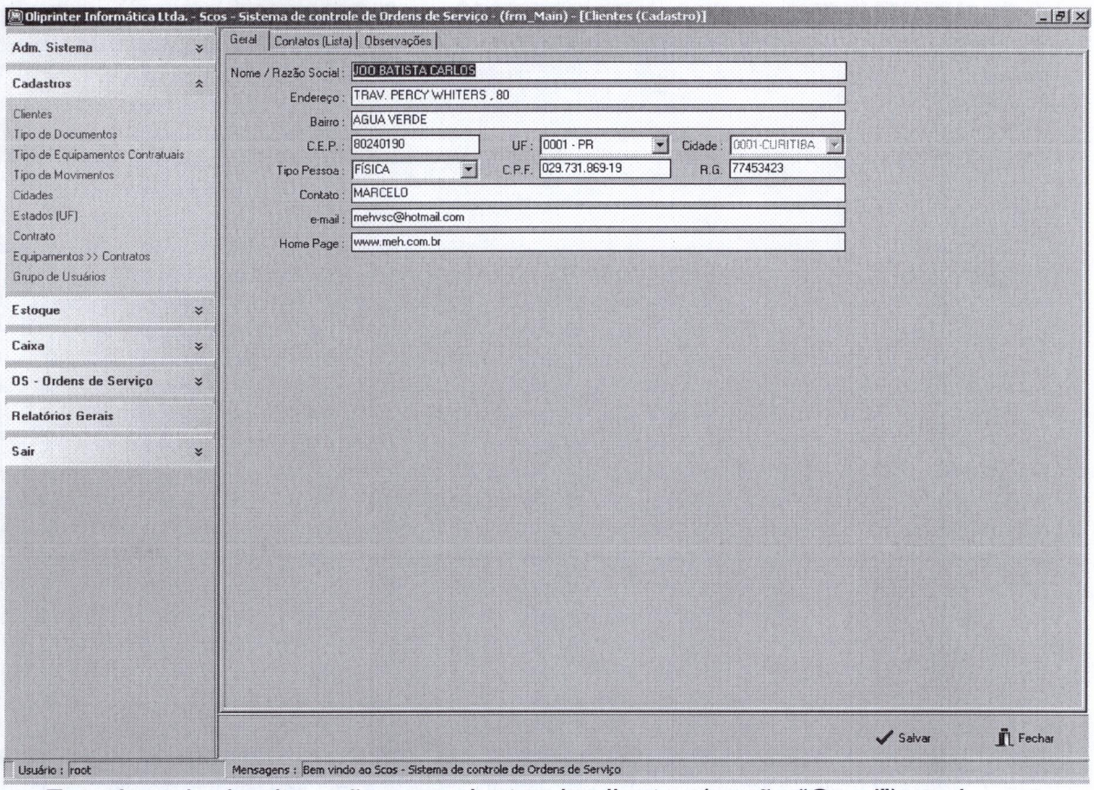
Na opção de cadastro de “Clientes” o usuário poderá cadastrar, incluir e alterar clientes.



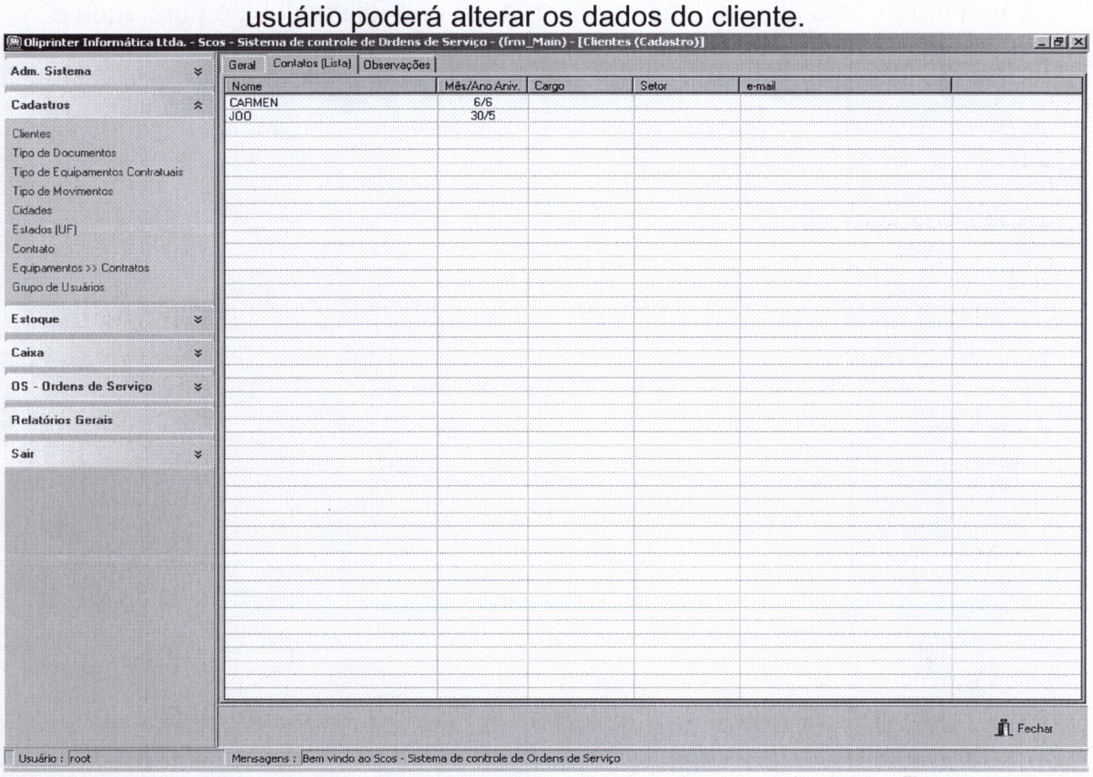
Esta é a tela de cadastro de clientes.



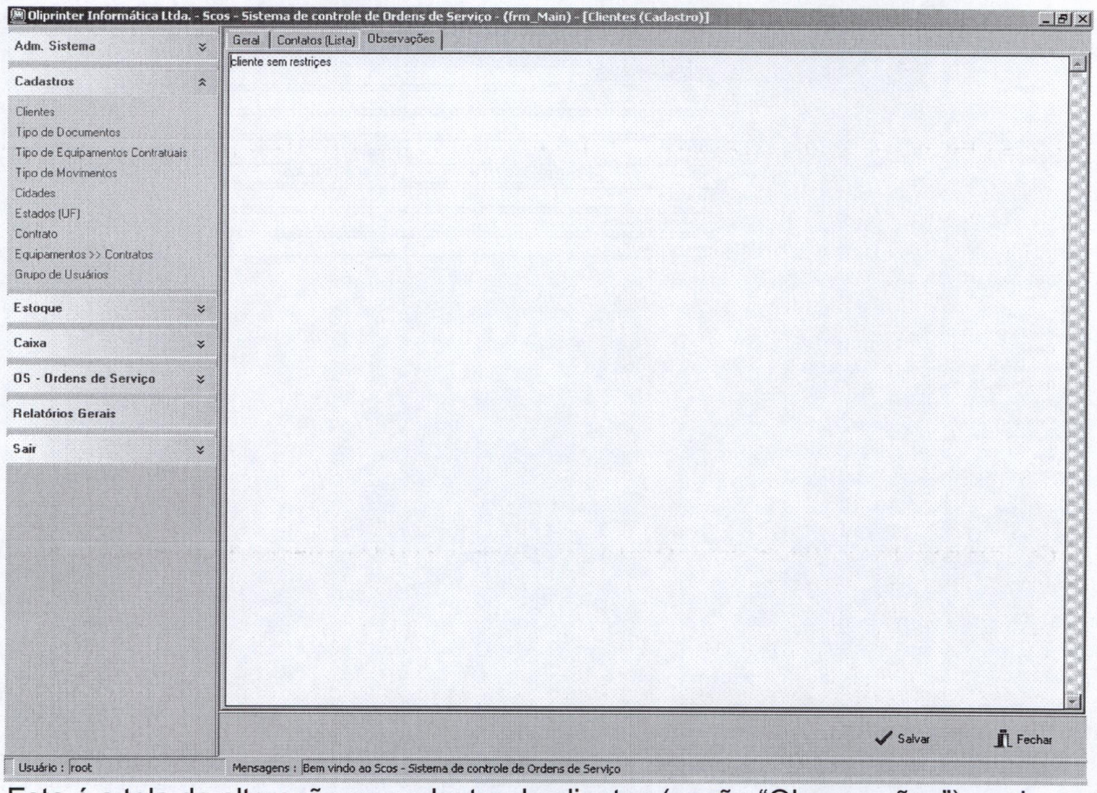
Esta é a tela de observações no cadastro de clientes, onde o usuário poderá colocar alguma observação a respeito do cliente.



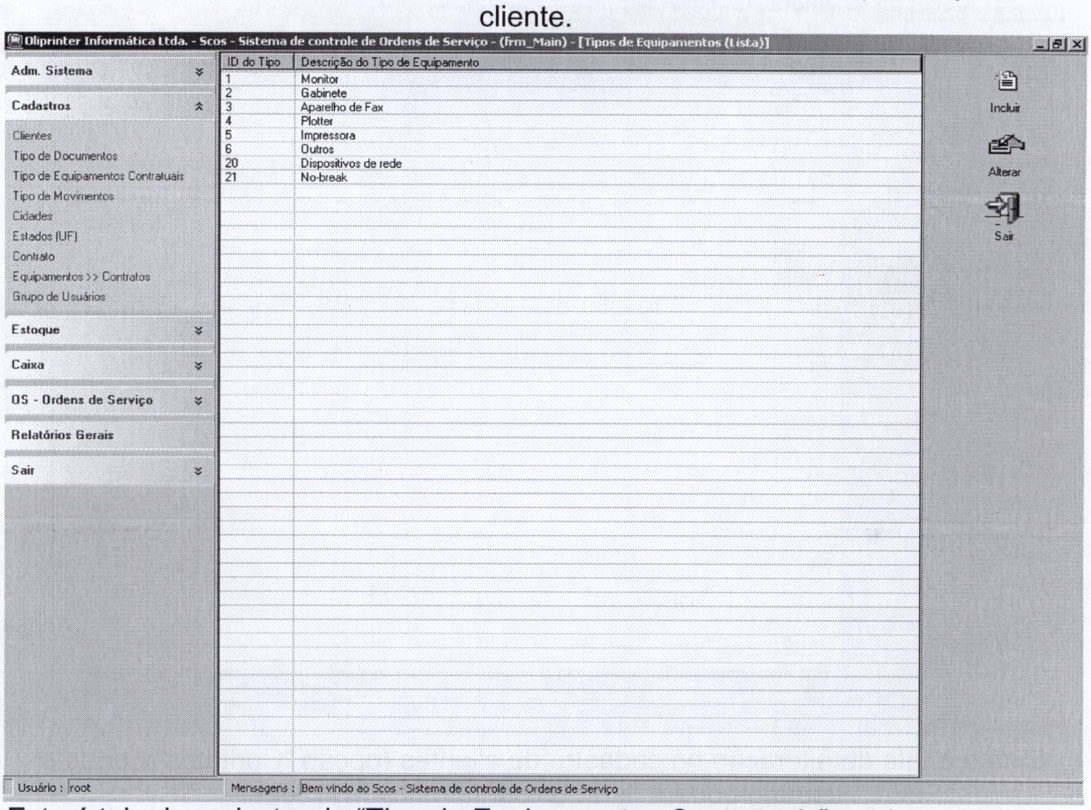
Esta é a tela de alteração no cadastro de clientes (opção “Geral”), onde o usuário poderá alterar os dados do cliente.



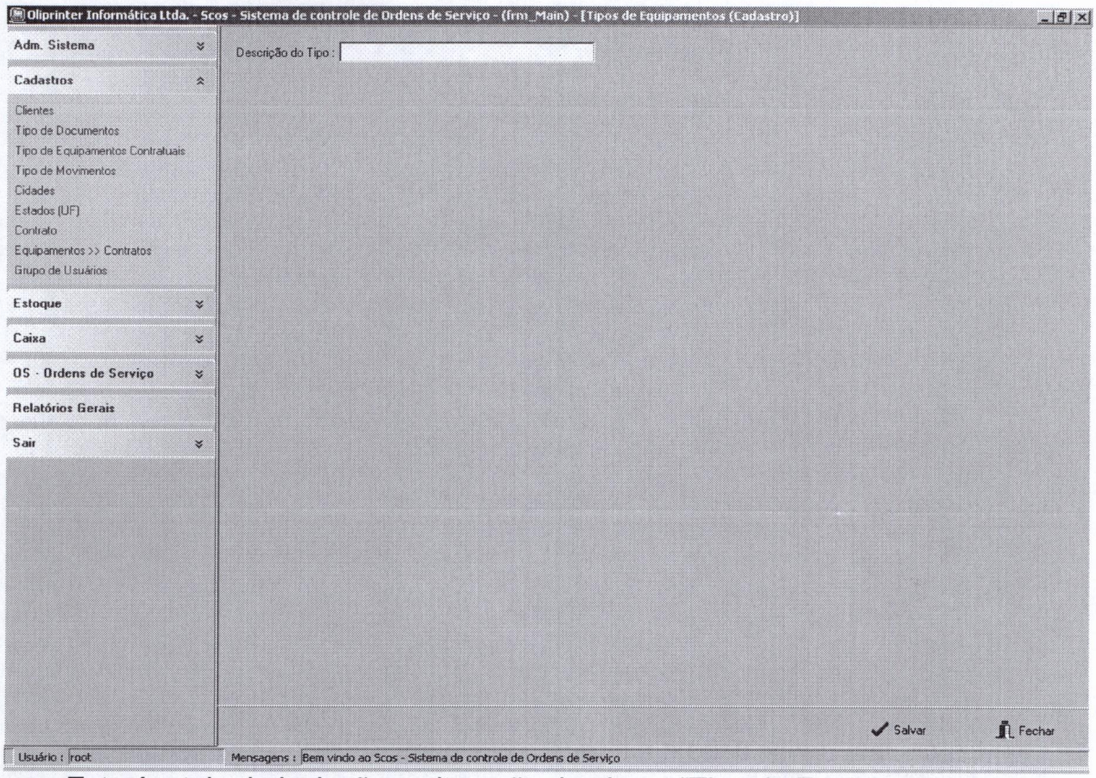
Esta é a tela de alteração no cadastro de clientes (opção “Contatos”), onde o usuário poderá listar os contatos cadastrados.



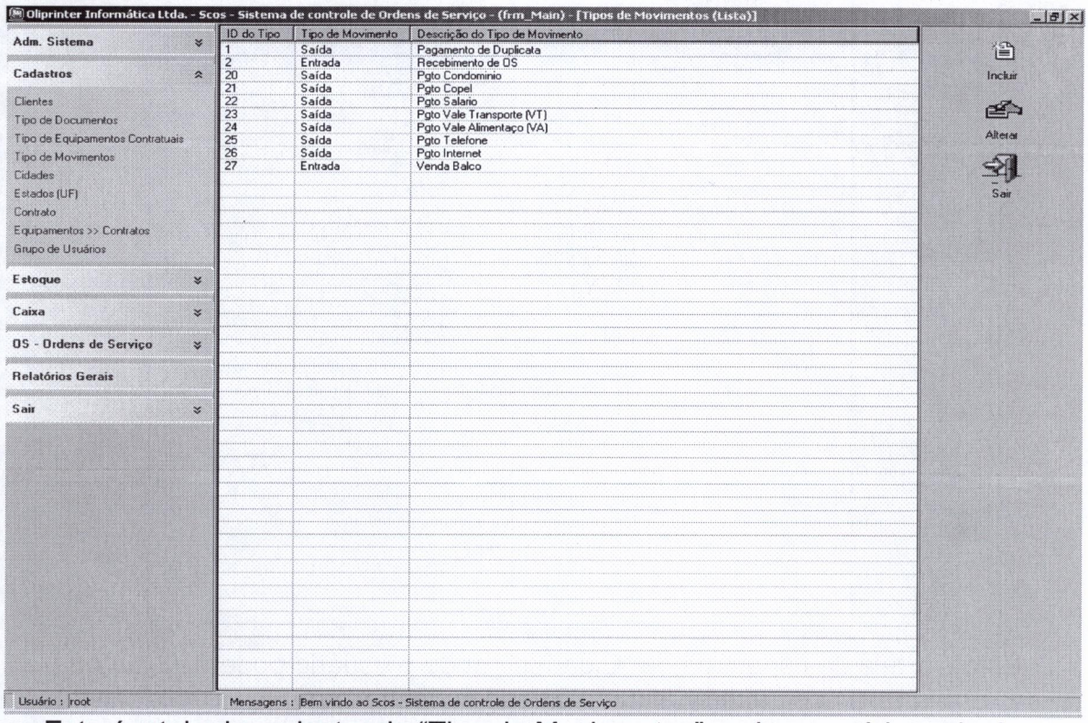
Esta é a tela de alteração no cadastro de clientes (opção "Observações"), onde o usuário poderá alterar alguma observação que foi concebida para aquele cliente.



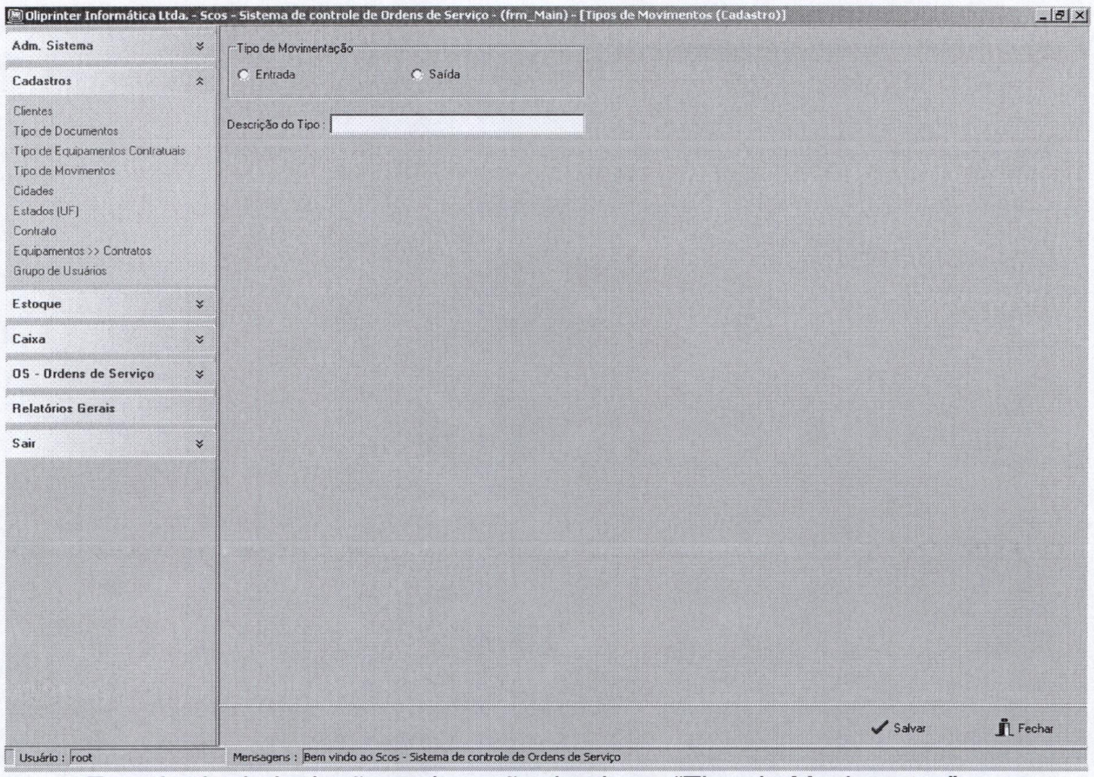
Esta é tela de cadastro de "Tipo de Equipamentos Contratuais", onde o usuário poderá incluir ou alterar algum tipo de equipamento contratual.



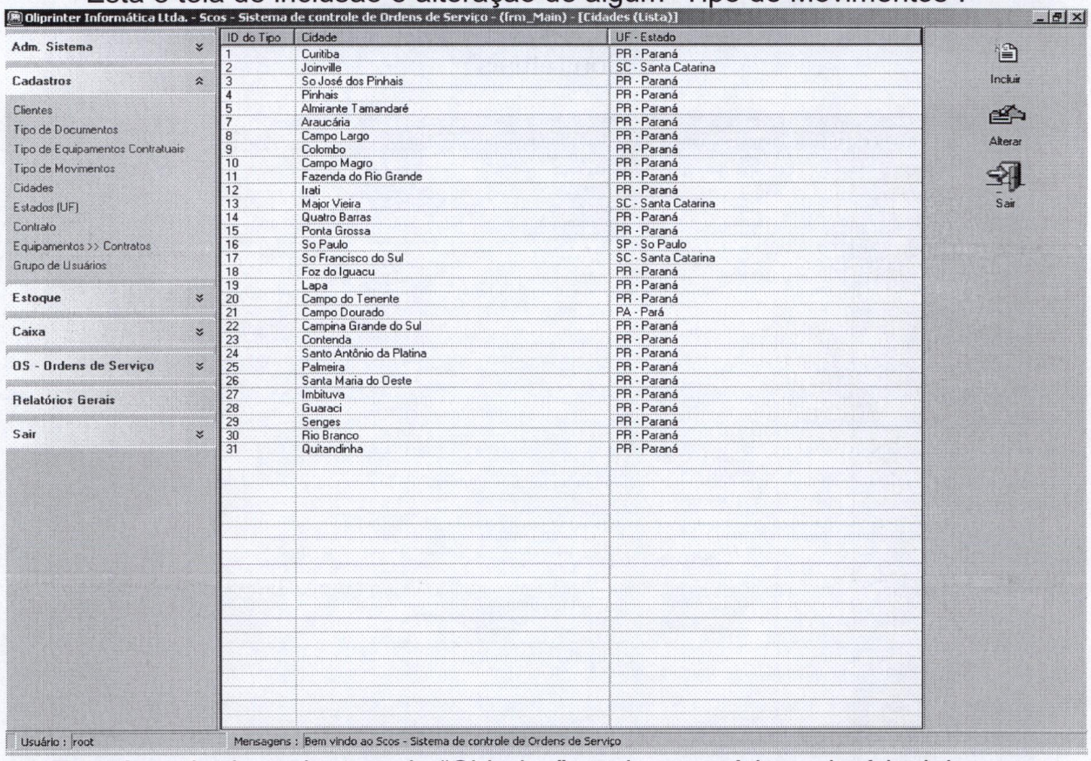
Esta é a tela de inclusão e alteração de algum “Tipo de Equipamentos Contratuais”.



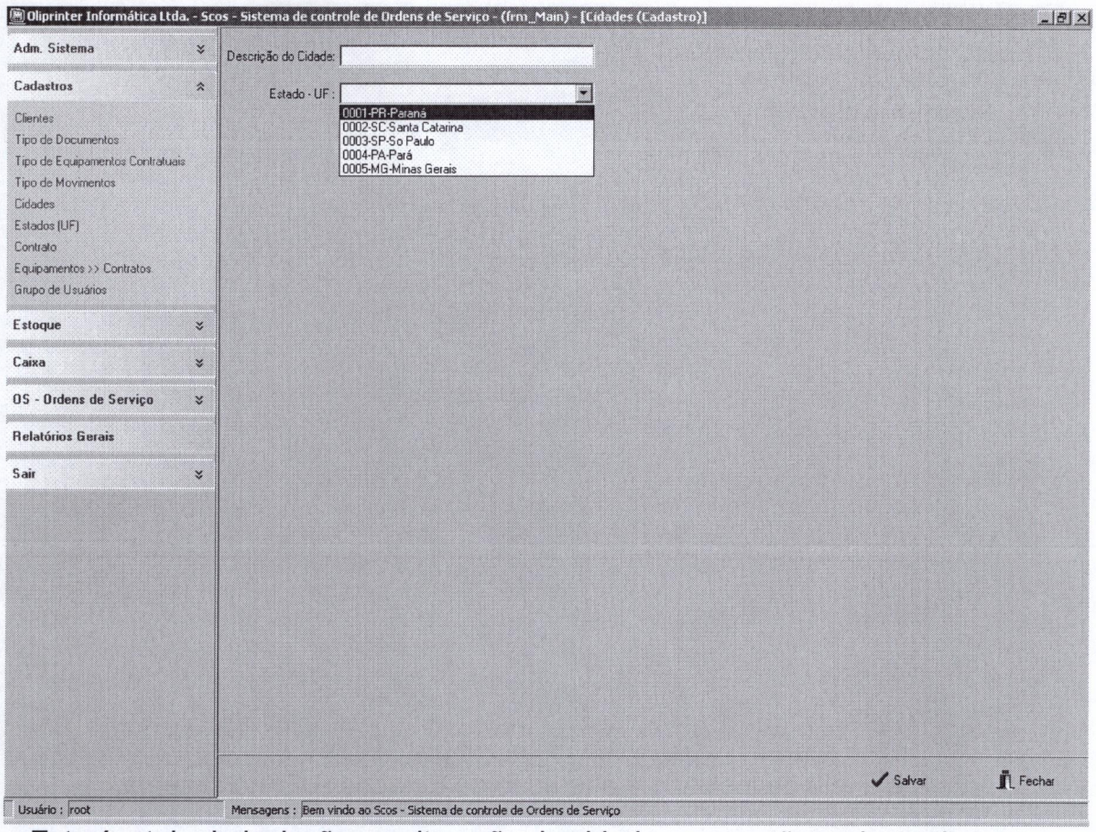
Esta é a tela de cadastro de “Tipo de Movimentos” onde o usuário poderá incluir ou alterar algum tipo de movimento.



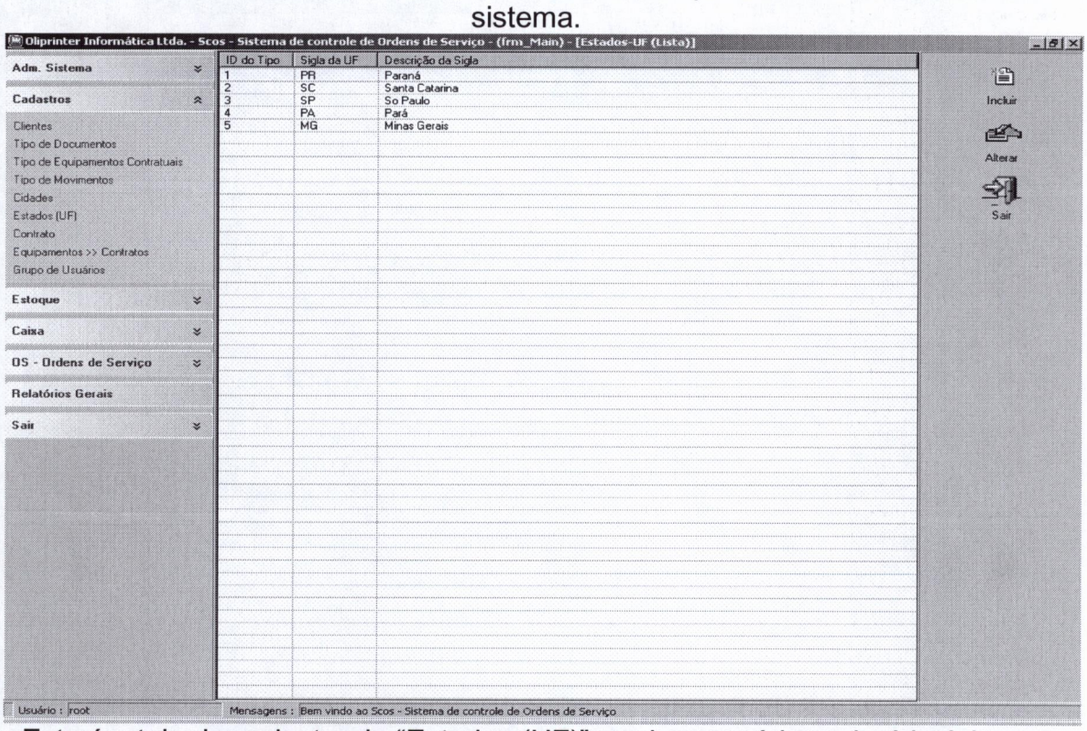
Esta é tela de inclusão e alteração de algum “Tipo de Movimentos”.



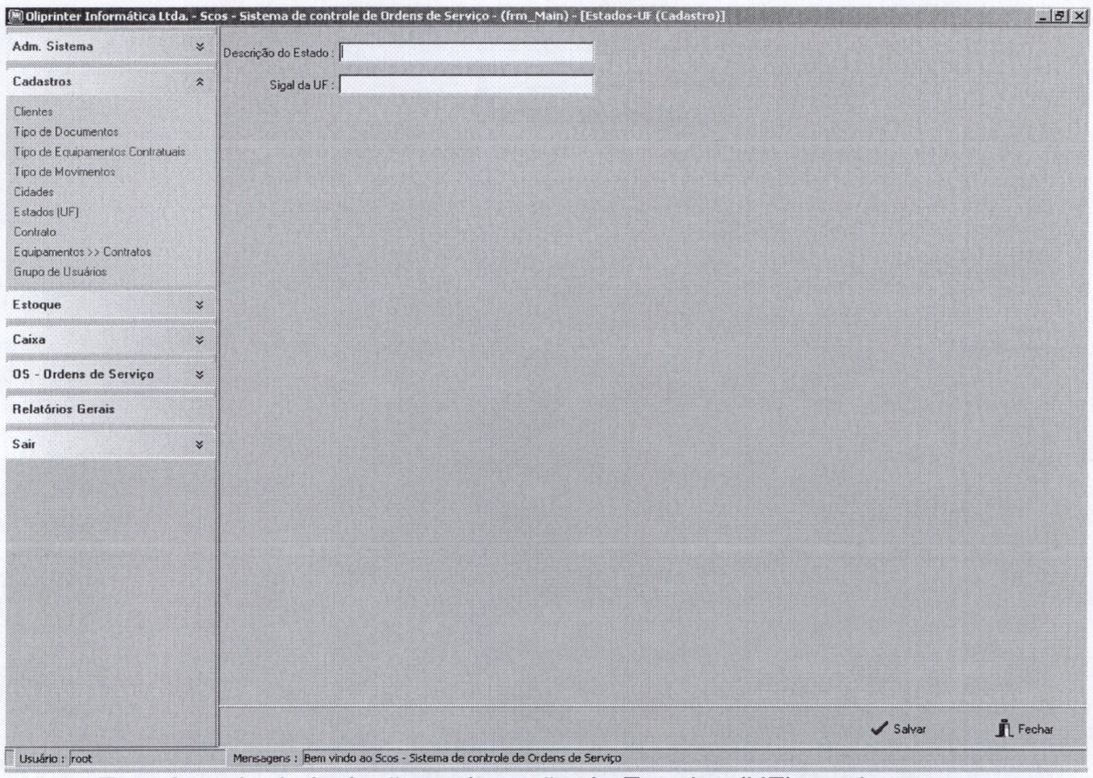
Esta é a tela de cadastros de “Cidades”, onde o usuário poderá incluir ou alterar alguma cidade cadastrada.



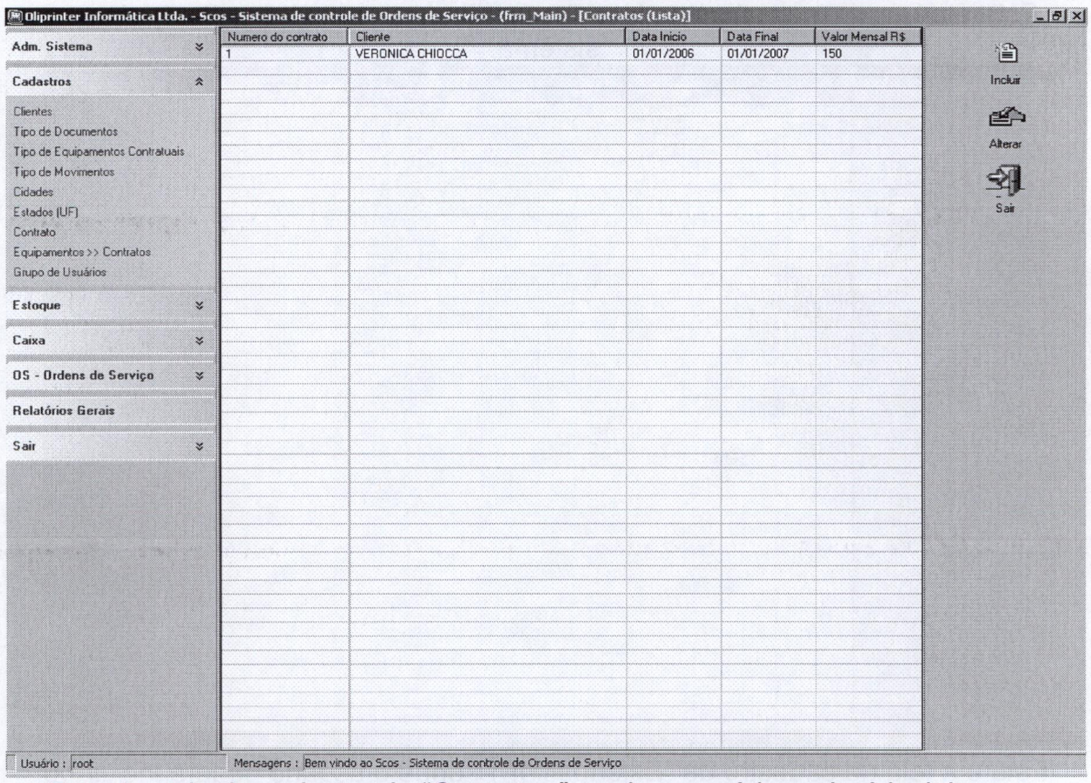
Esta é a tela de inclusão ou alteração de cidades que estão cadastradas no sistema.



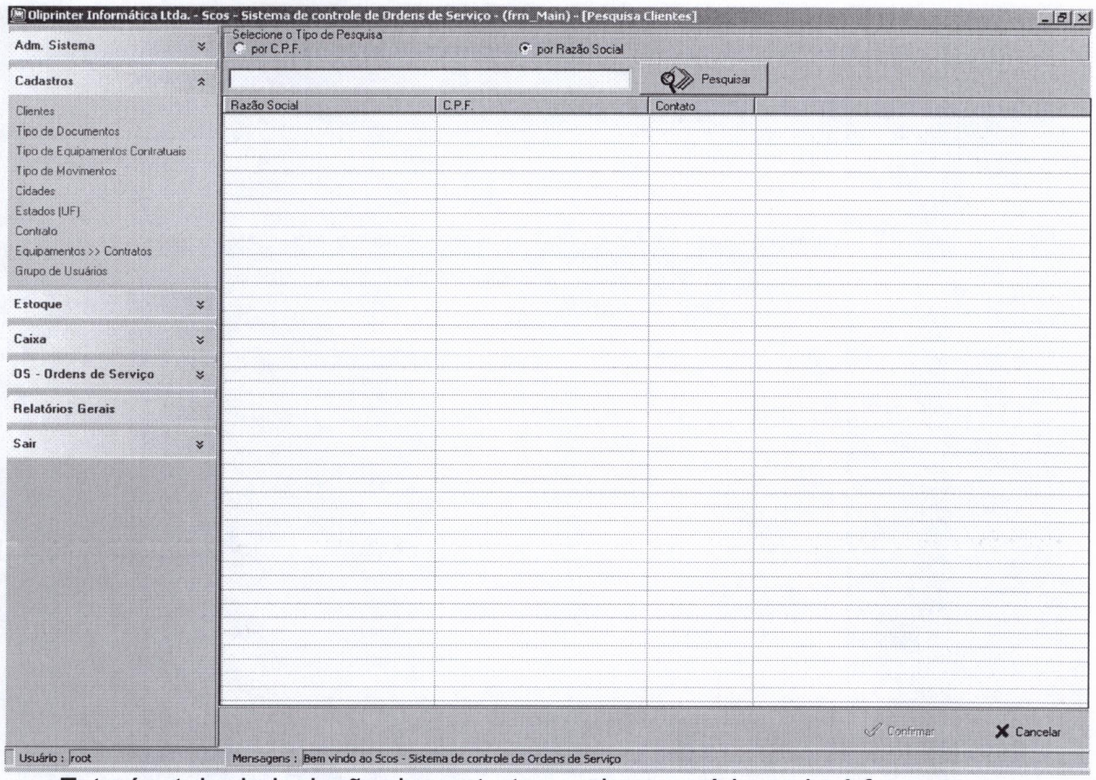
Esta é a tela de cadastro de “Estados (UF)”, onde o usuário poderá incluir ou alterar Estados (UF) no sistema.



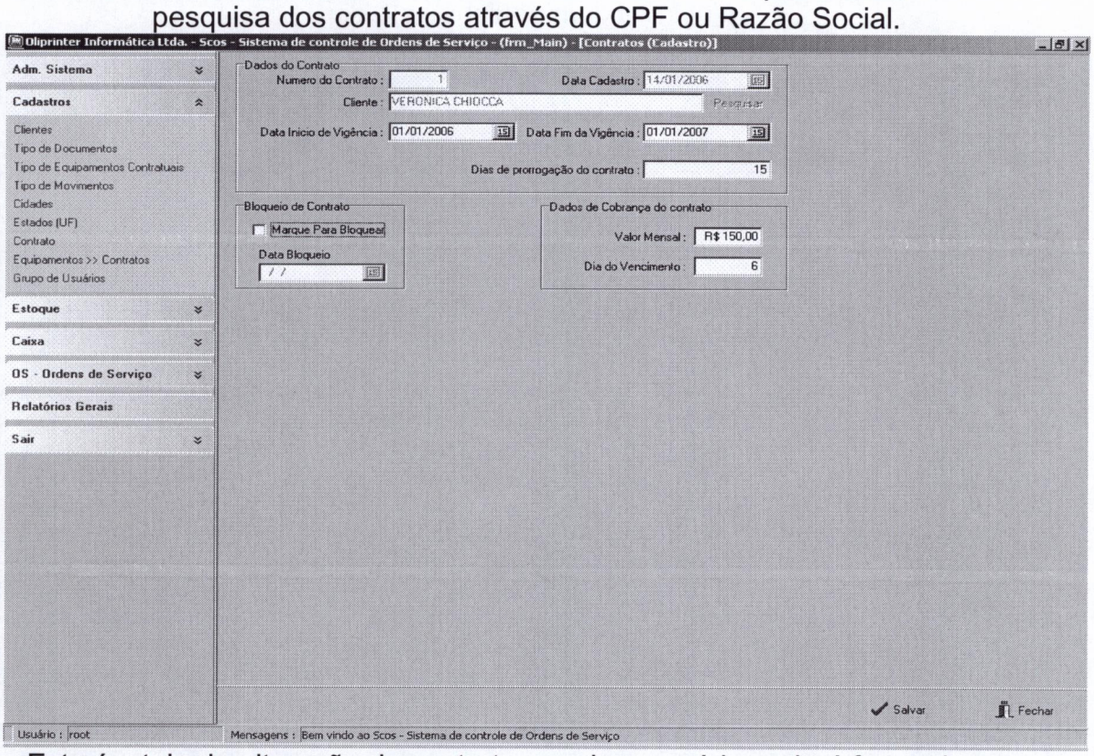
Esta é a tela de inclusão e alteração de Estados (UF) no sistema.



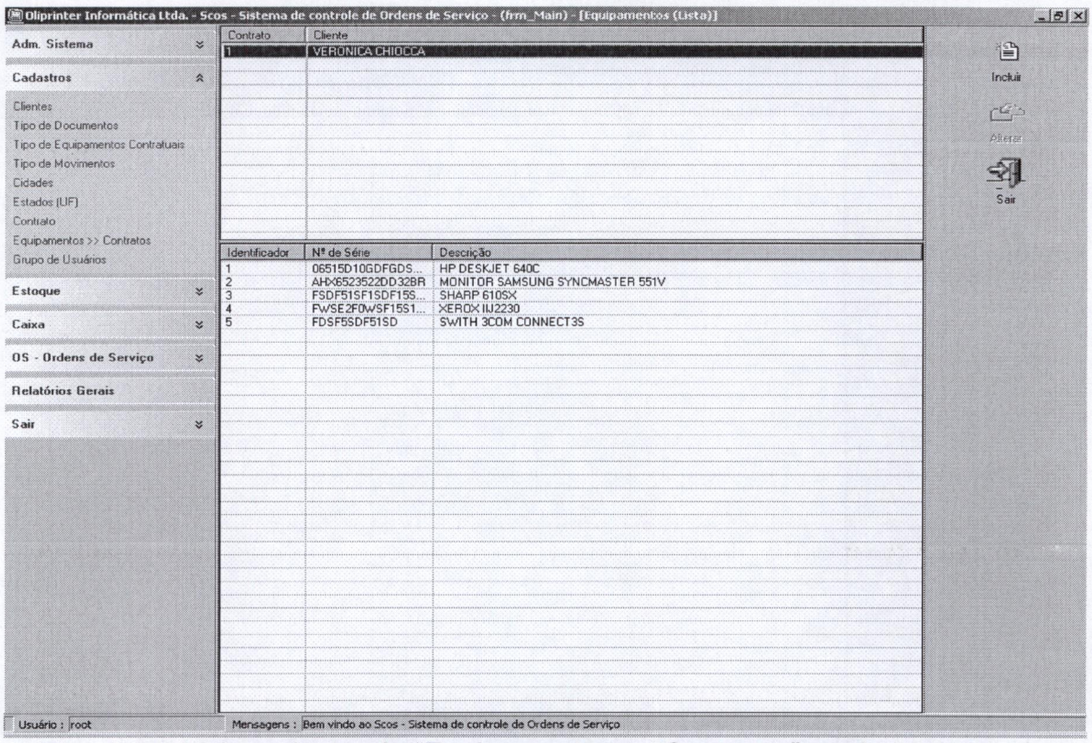
Esta é a tela de cadastro de “Contratos”, onde o usuário poderá incluir ou alterar algum contrato.



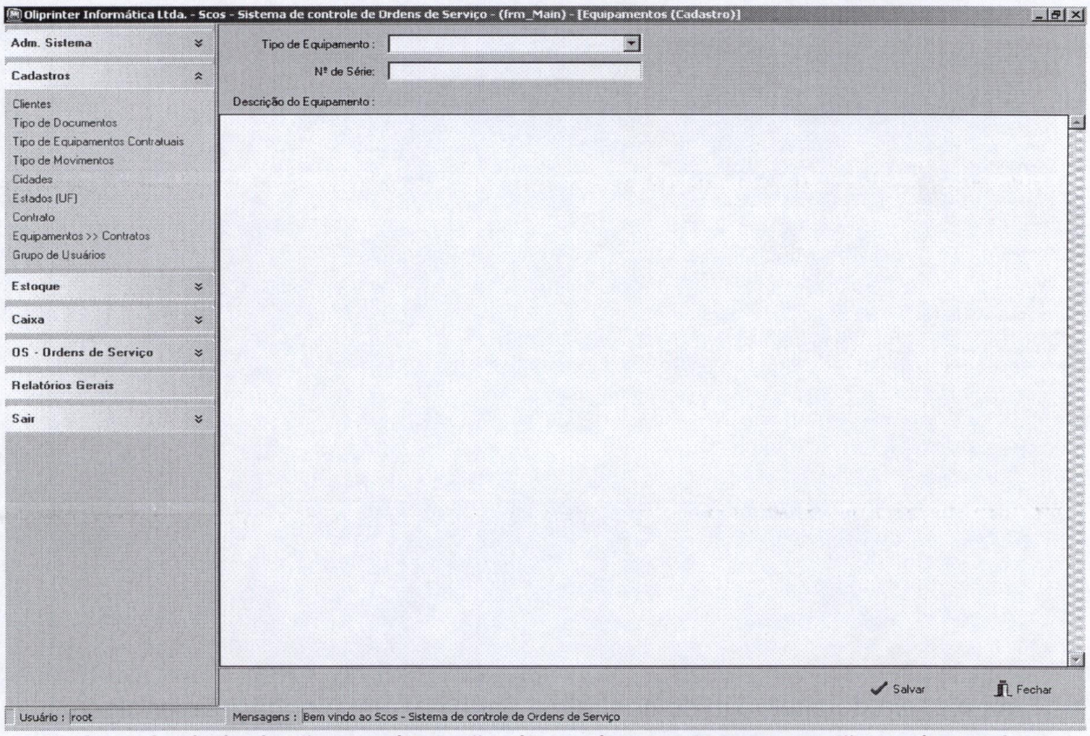
Esta é a tela de inclusão de contratos onde o usuário poderá fazer uma pesquisa dos contratos através do CPF ou Razão Social.



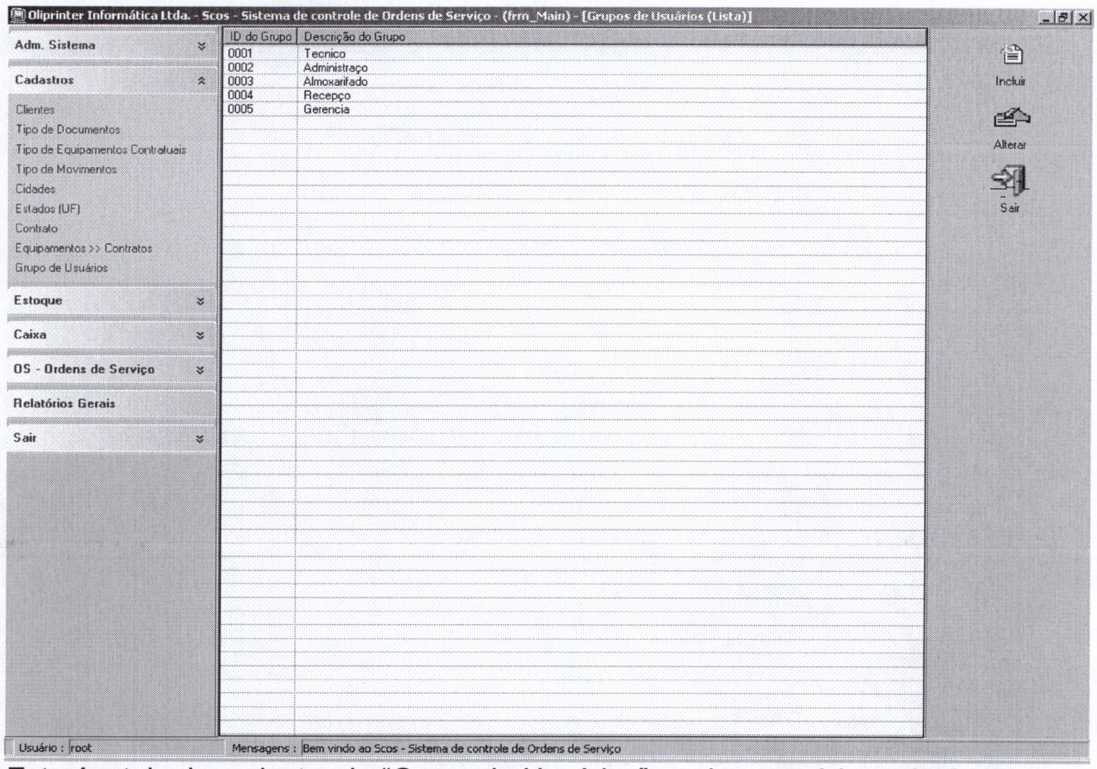
Esta é a tela de alteração de contratos, onde o usuário poderá fazer alguma alteração no contrato cadastrado.



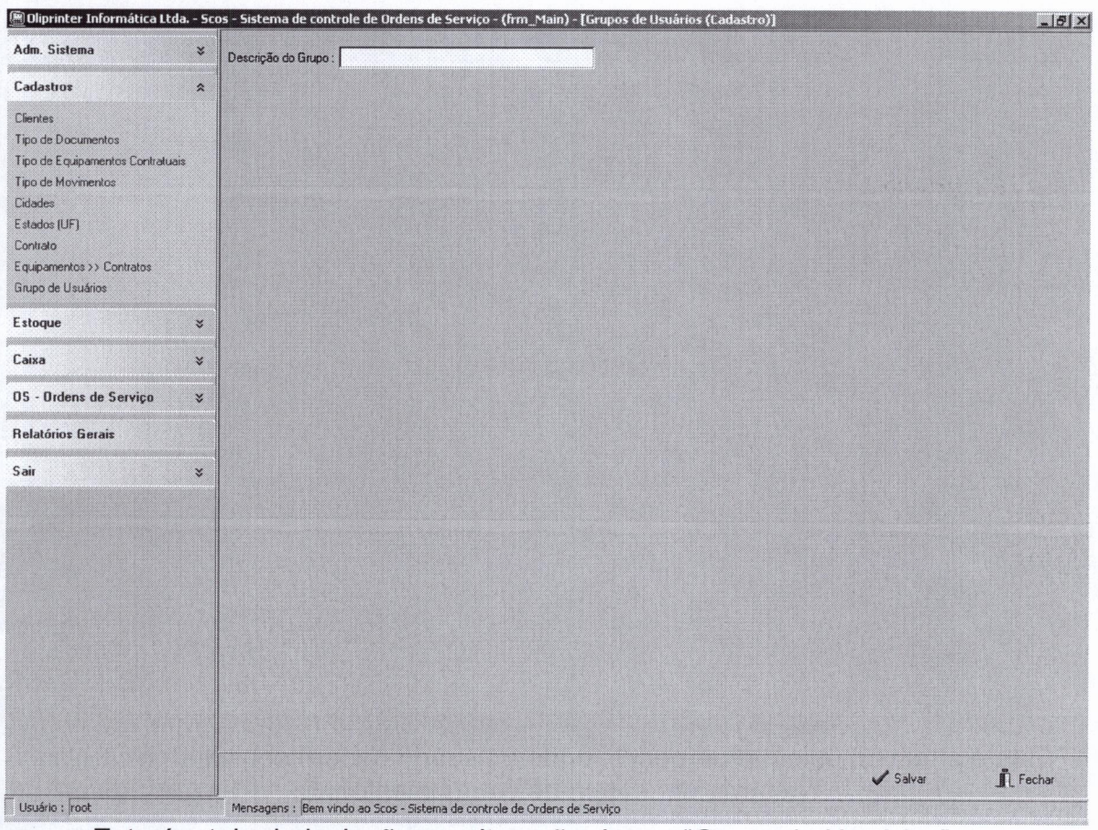
Esta é a tela de cadastro de “Equipamentos >> Contratos”, onde o usuário poderá cadastrar equipamentos que estão no contrato. Podendo alterar ou incluir algum equipamento.



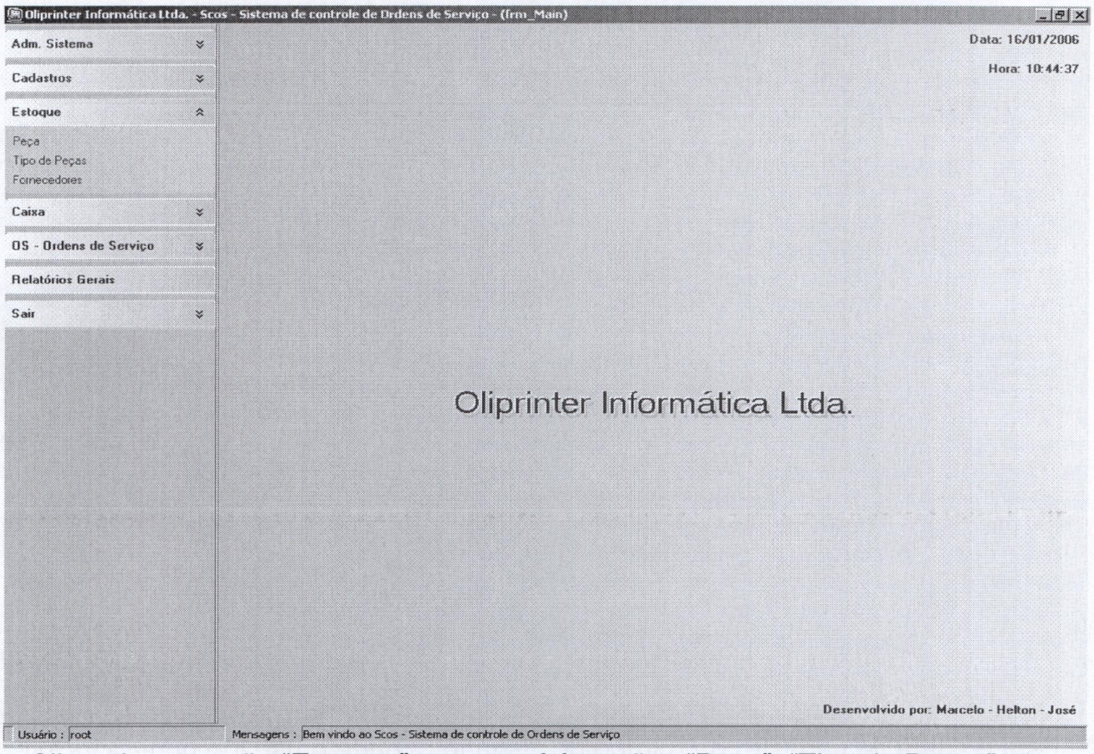
Esta é a tela de inclusão ou alteração de equipamentos que estão cadastrados como contrato.



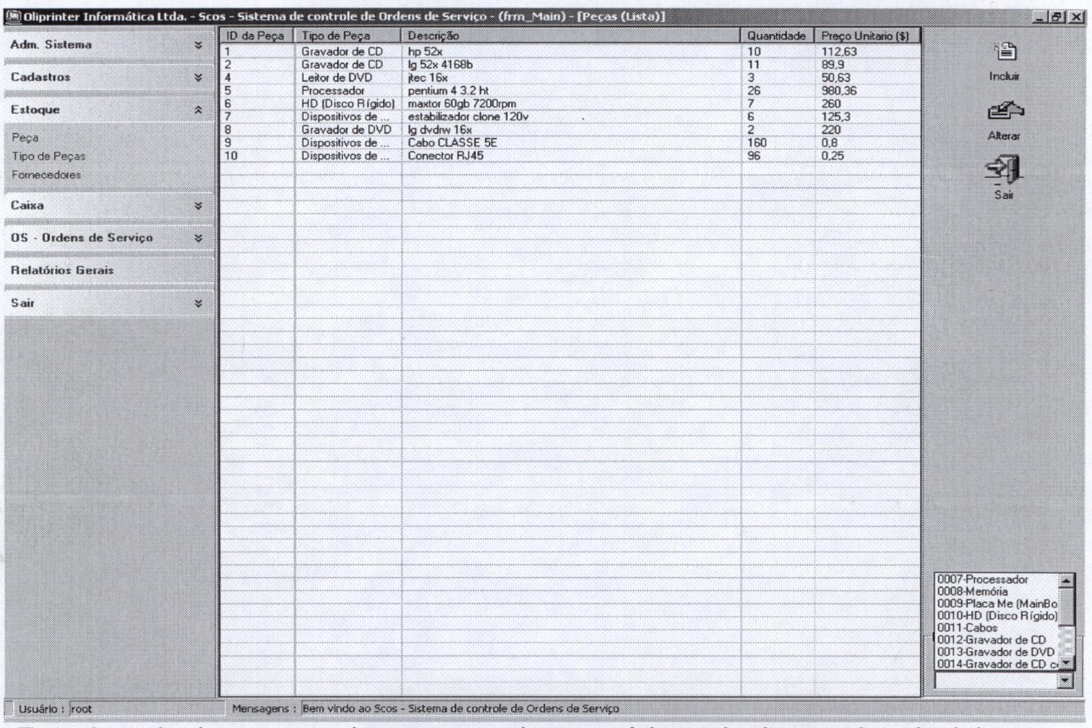
Esta é a tela de cadastro de "Grupo de Usuários", onde o usuário poderá incluir ou alterar os tipos de usuários.



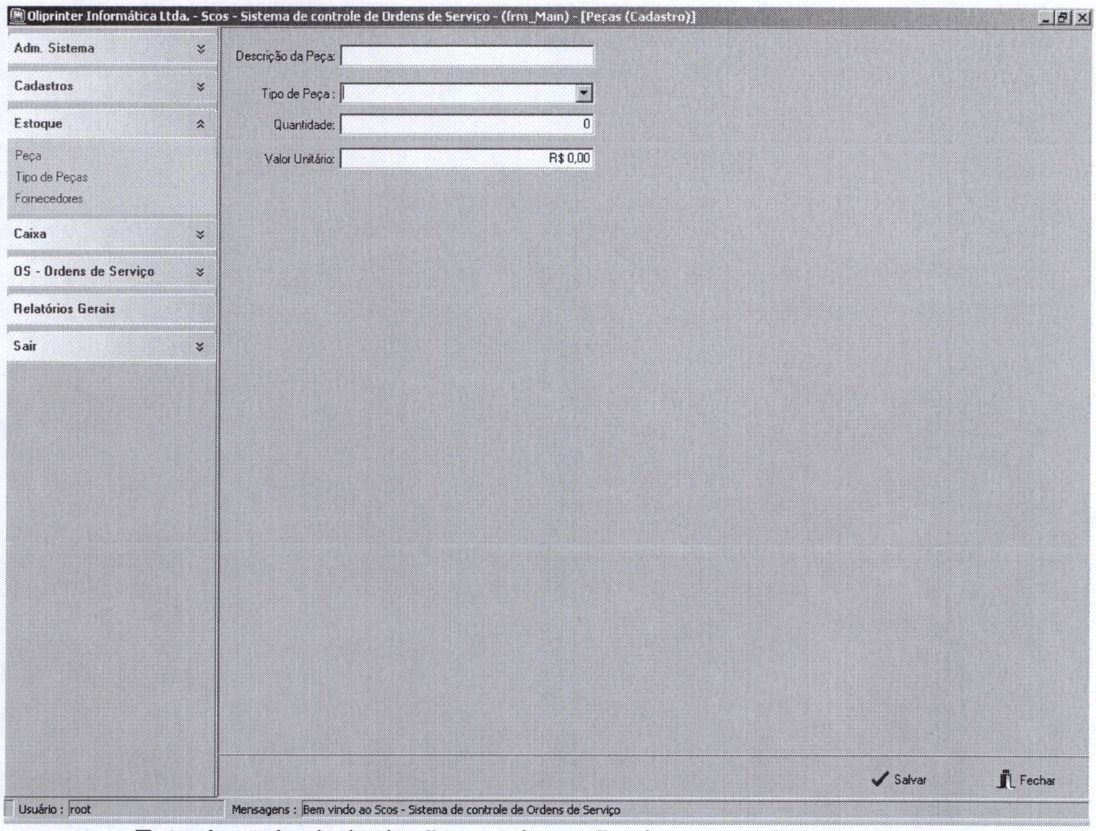
Esta é a tela de inclusão ou alteração de um "Grupo de Usuários".



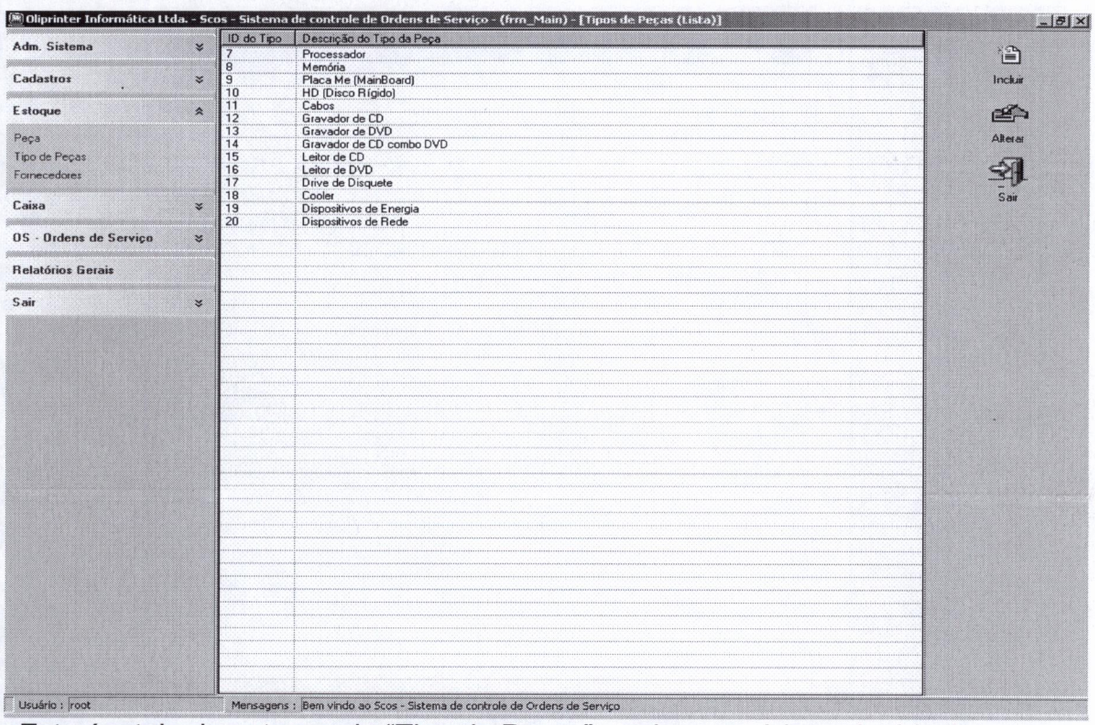
Clicando na opção “Estoque” aparecerá às opões “Peça”, “Tipo de Peças” e “Fornecedores”.



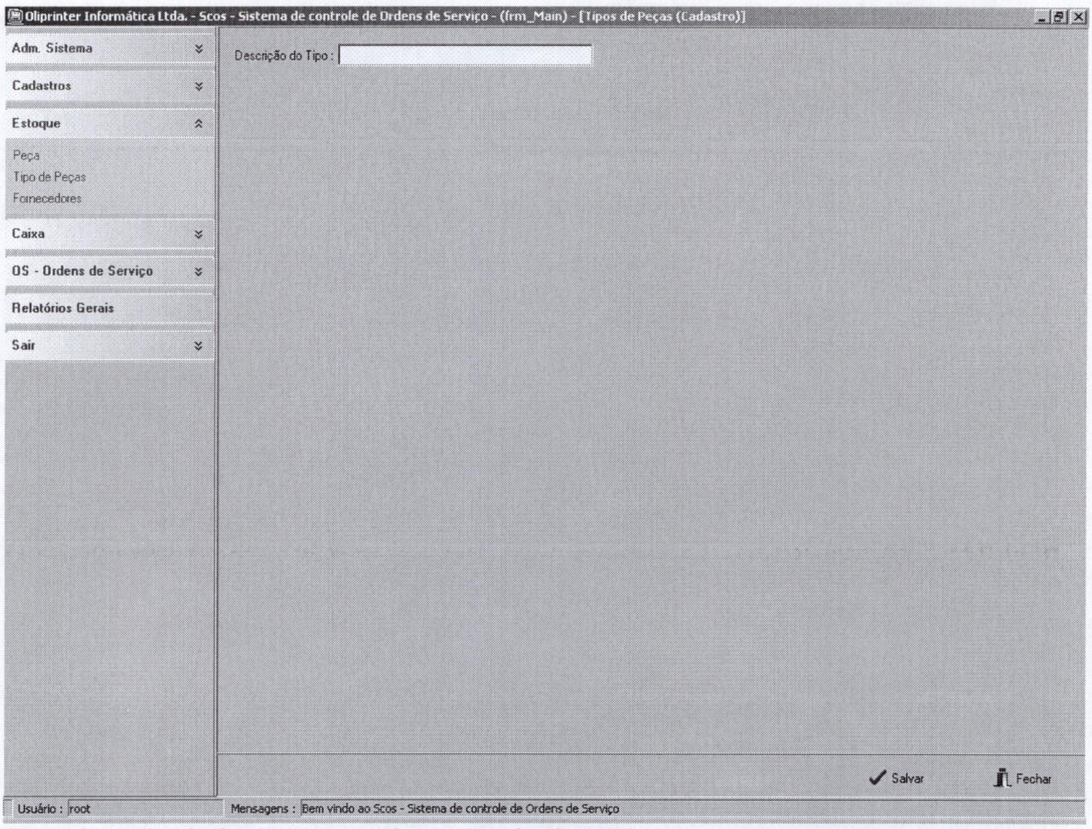
Esta é a tela do estoque de peças, onde o usuário poderá consultar, incluir ou alterar algum tipo de peça do estoque.



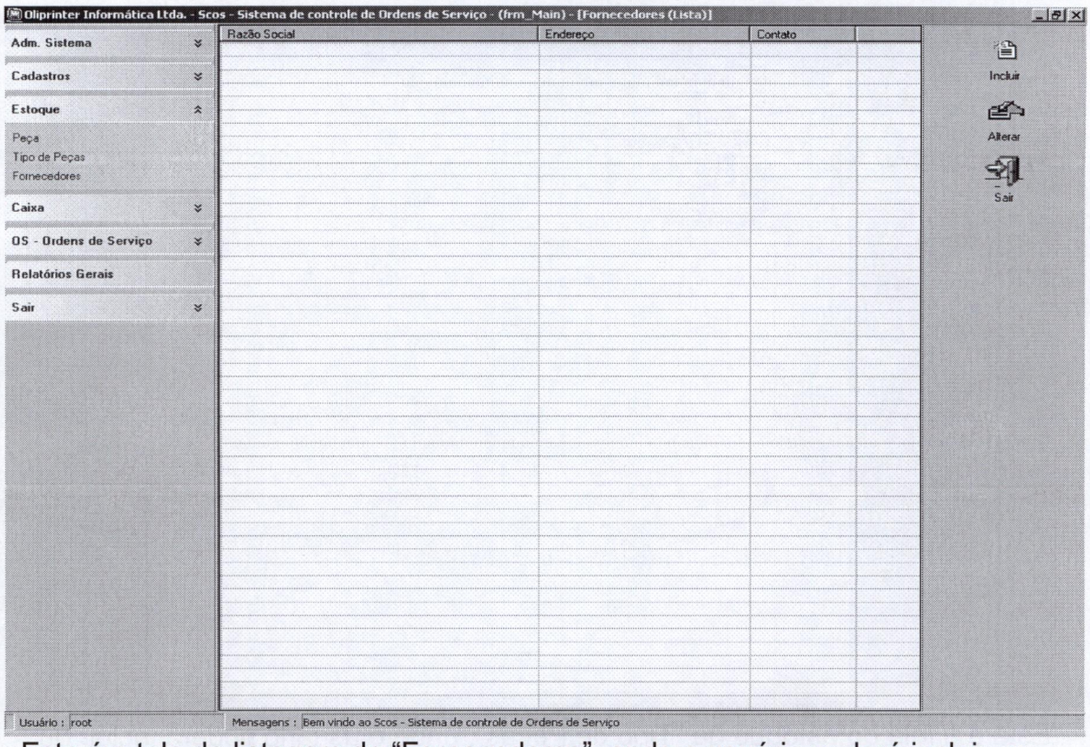
Esta é a tela de inclusão ou alteração de peças do estoque.



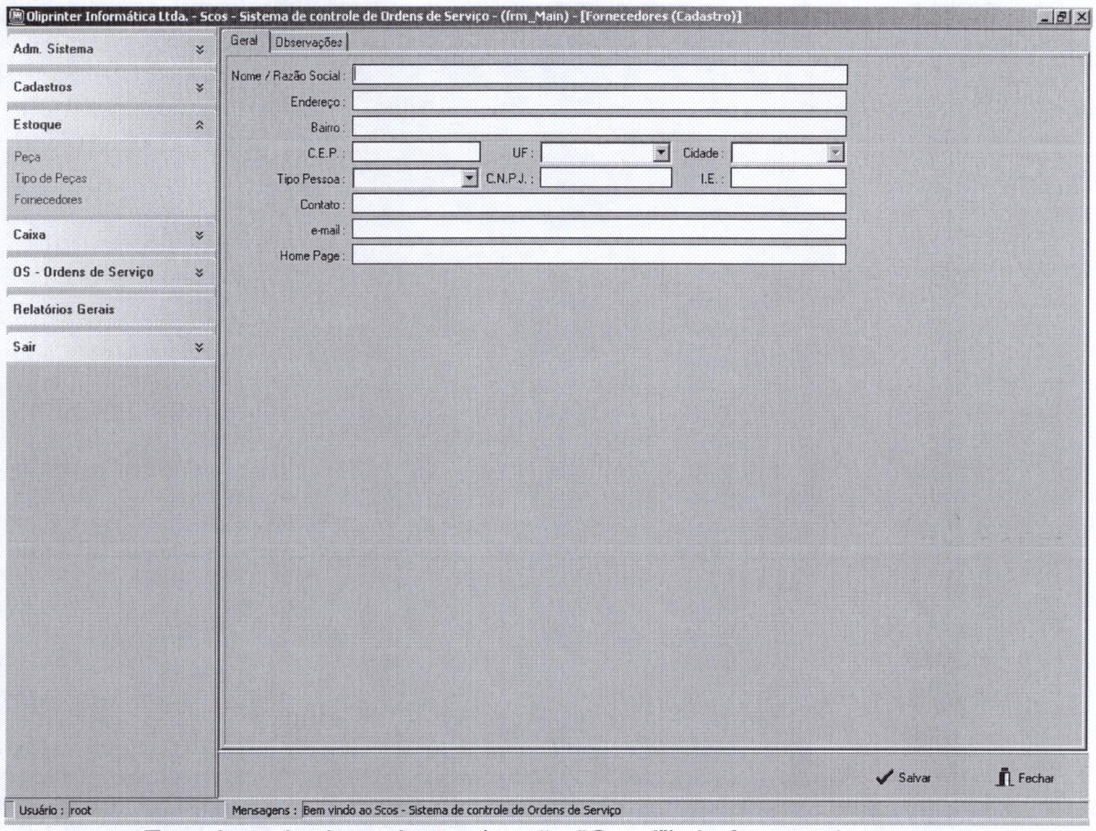
Esta é a tela do estoque de “Tipo de Peças”, onde o usuário poderá incluir ou alterar certos tipos de peças.



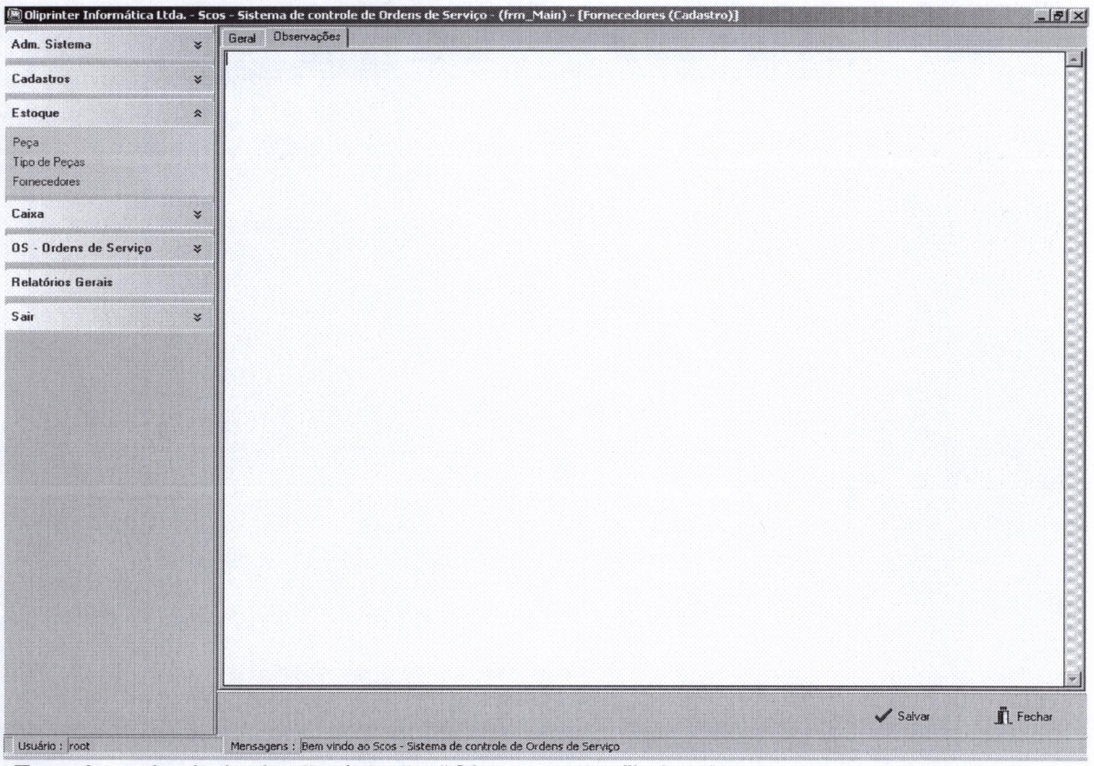
Esta é a tela de inclusão ou alteração de tipos de peças no estoque do sistema



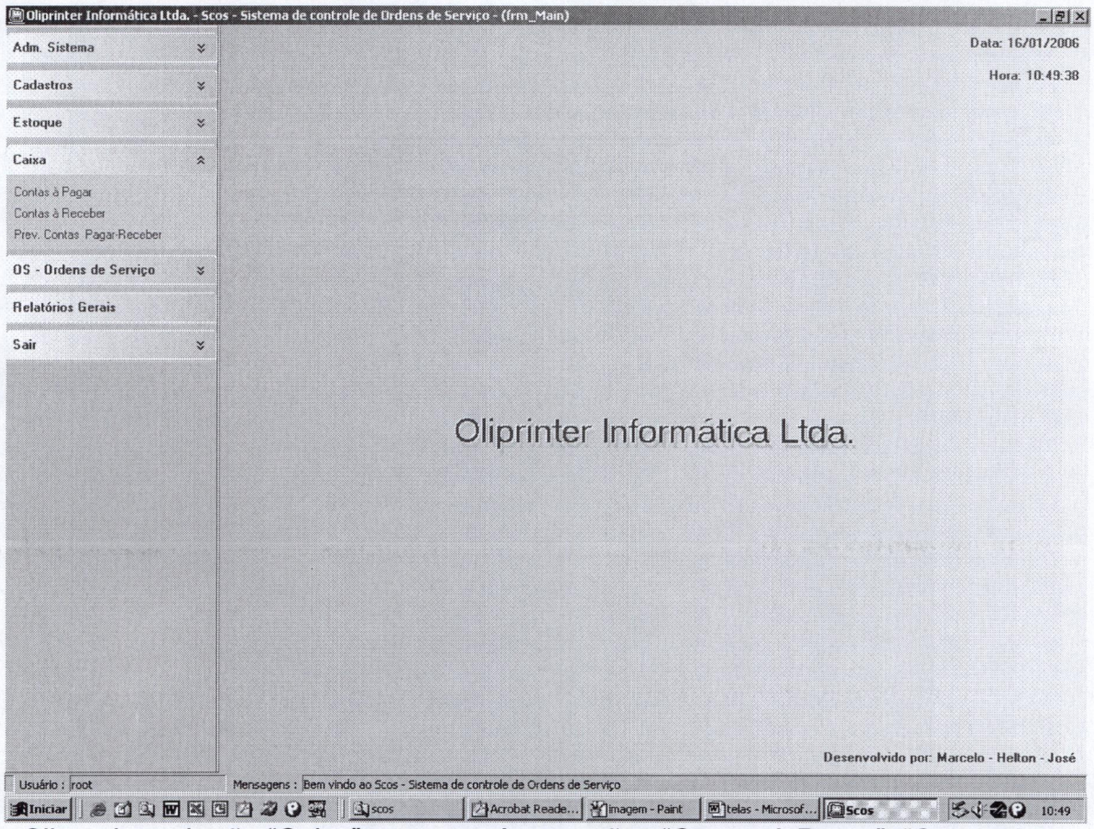
Esta é a tela de listagem de “Fornecedores”, onde o usuário poderá incluir ou alterar fornecedores.



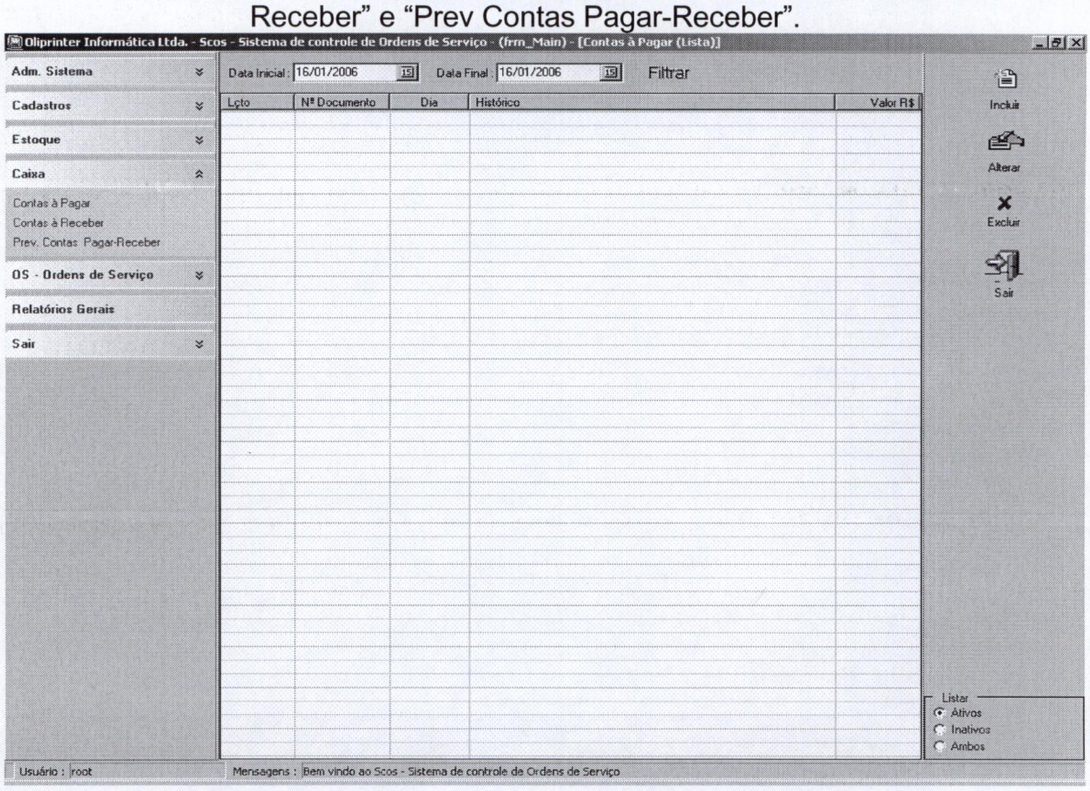
Esta é a tela de cadastro (opção “Geral”) de fornecedores.

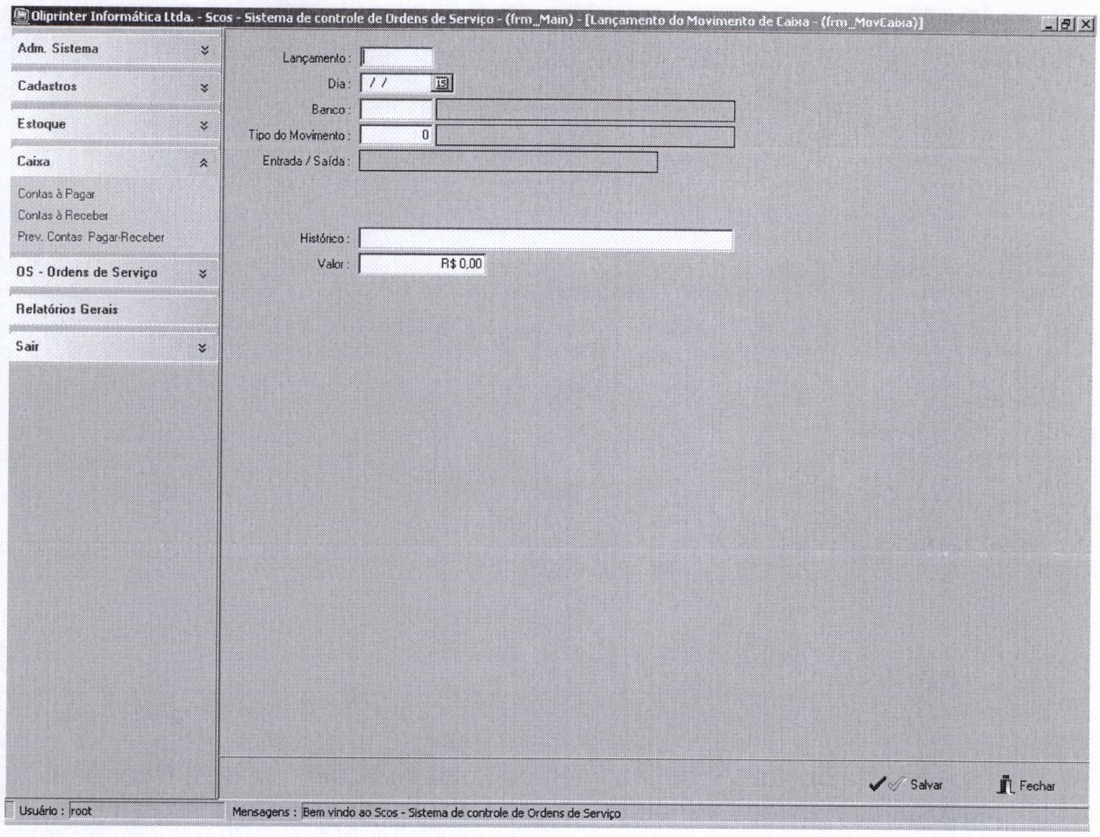


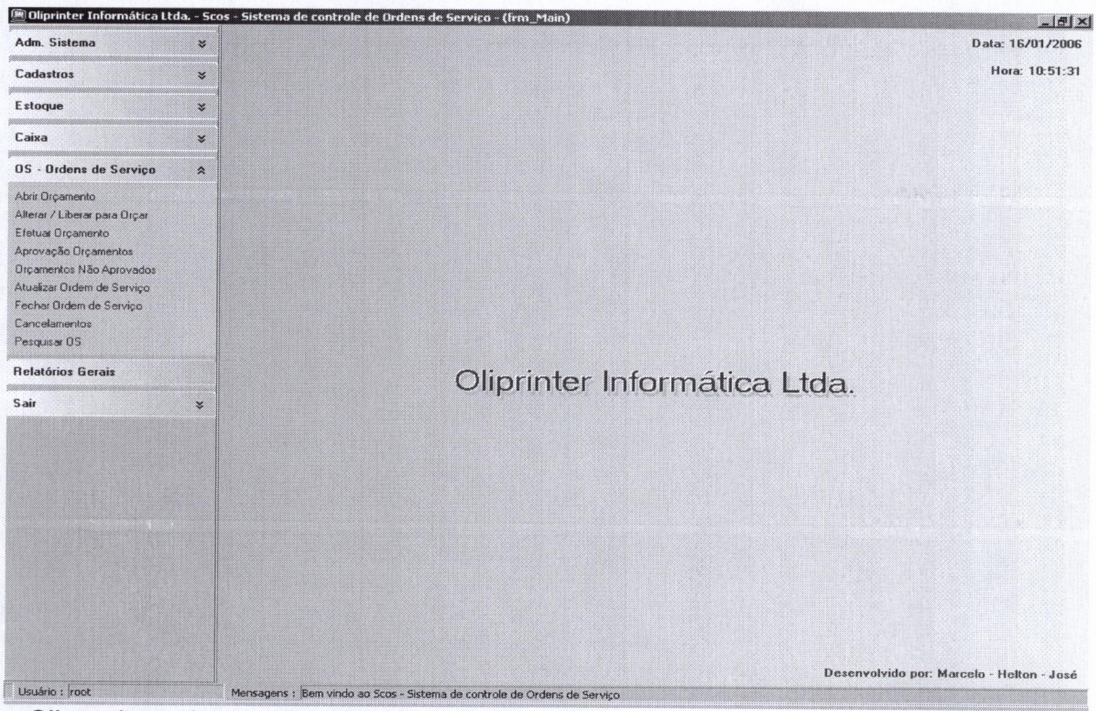
Esta é a tela de inclusão (opção “Observações”) de observações a respeito do fornecedor.



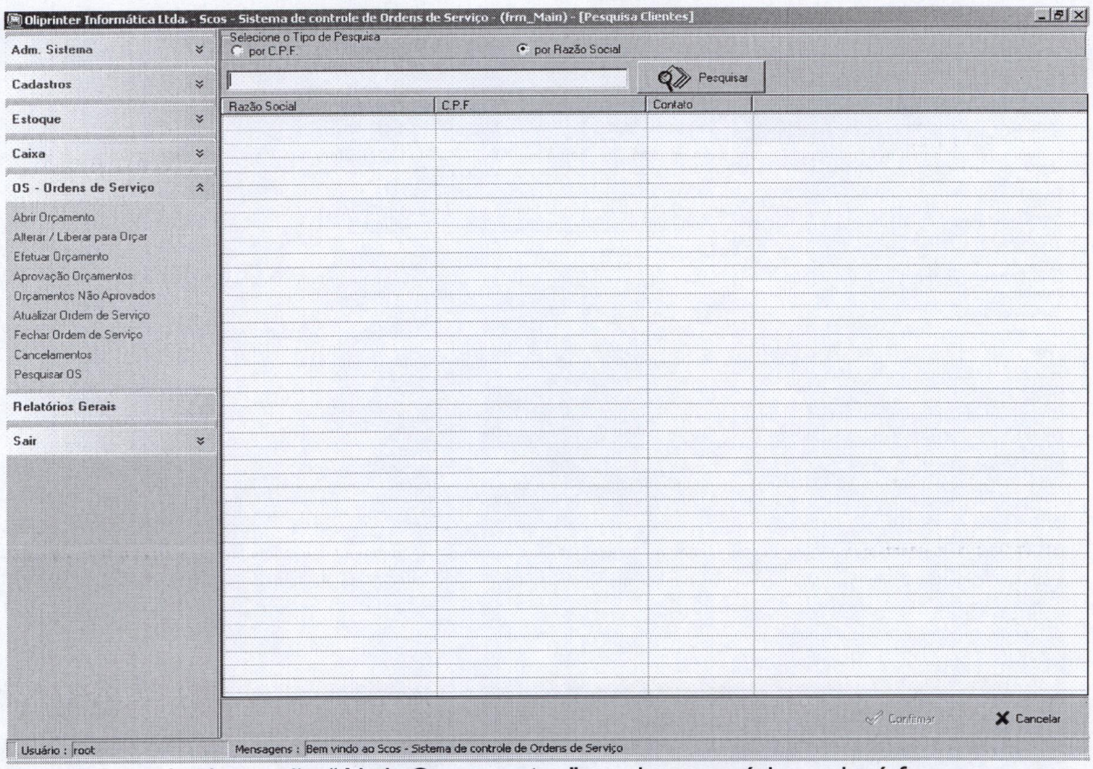
Clicando no botão “Caixa”, aparecerá as opções “Contas à Pagar”, “Contas à Receber” e “Prev Contas Pagar-Receber”.



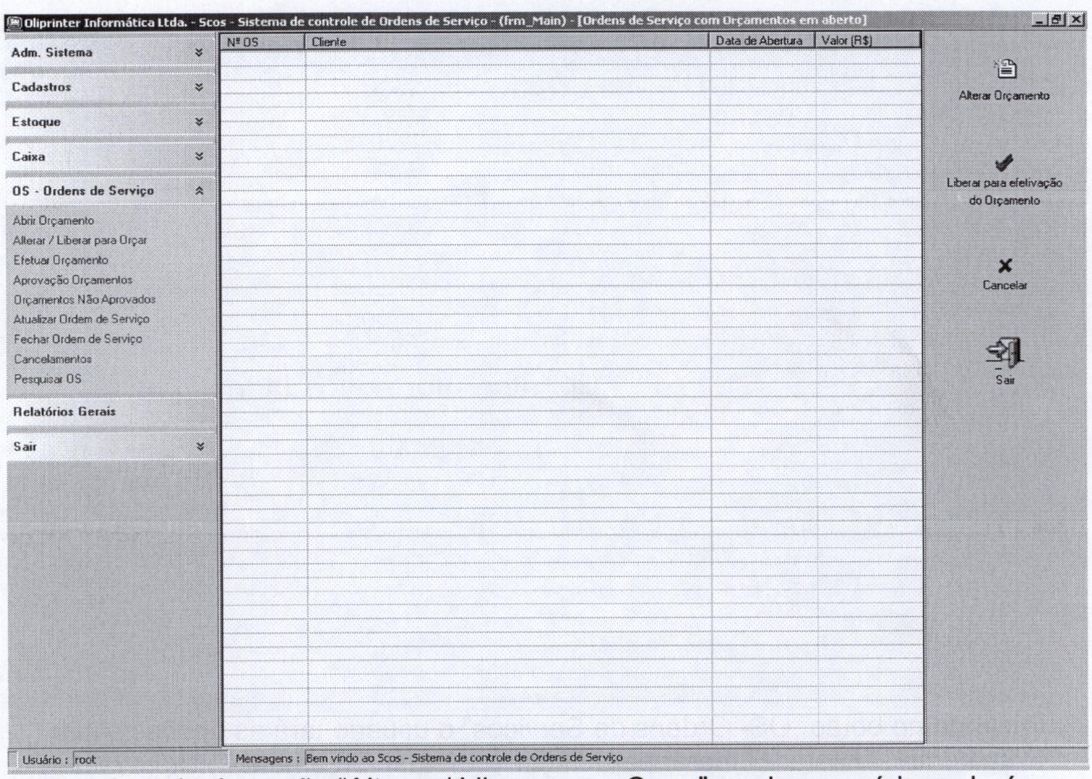




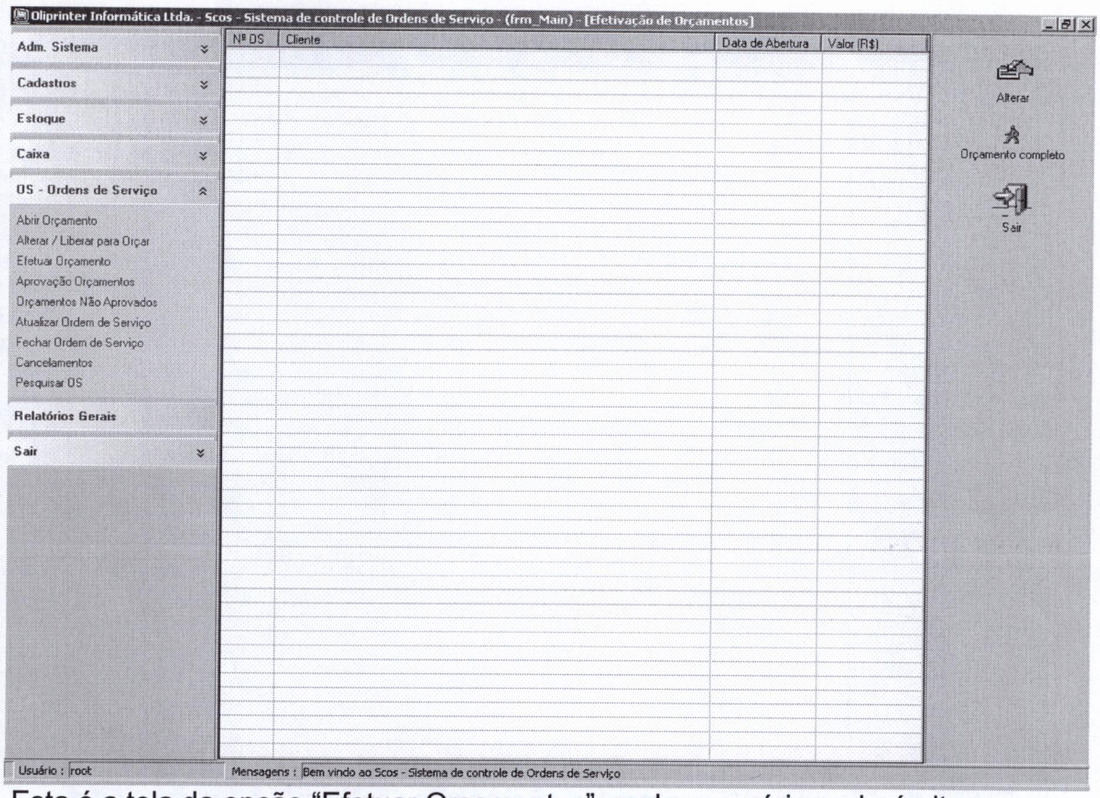
Clicando no botão “OS- Ordens de Serviços” o usuário terá as opções “Abrir Orçamento”, “Alterar / Liberar para Orçar”, “Efetuar Orçamento”, “Aprovação Orçamentos”, Orçamentos Não Aprovados”, “Atualizar Ordens de Serviço”, “Fechar Ordem de Serviço”, “Cancelamentos” e “Pesquisar OS”.



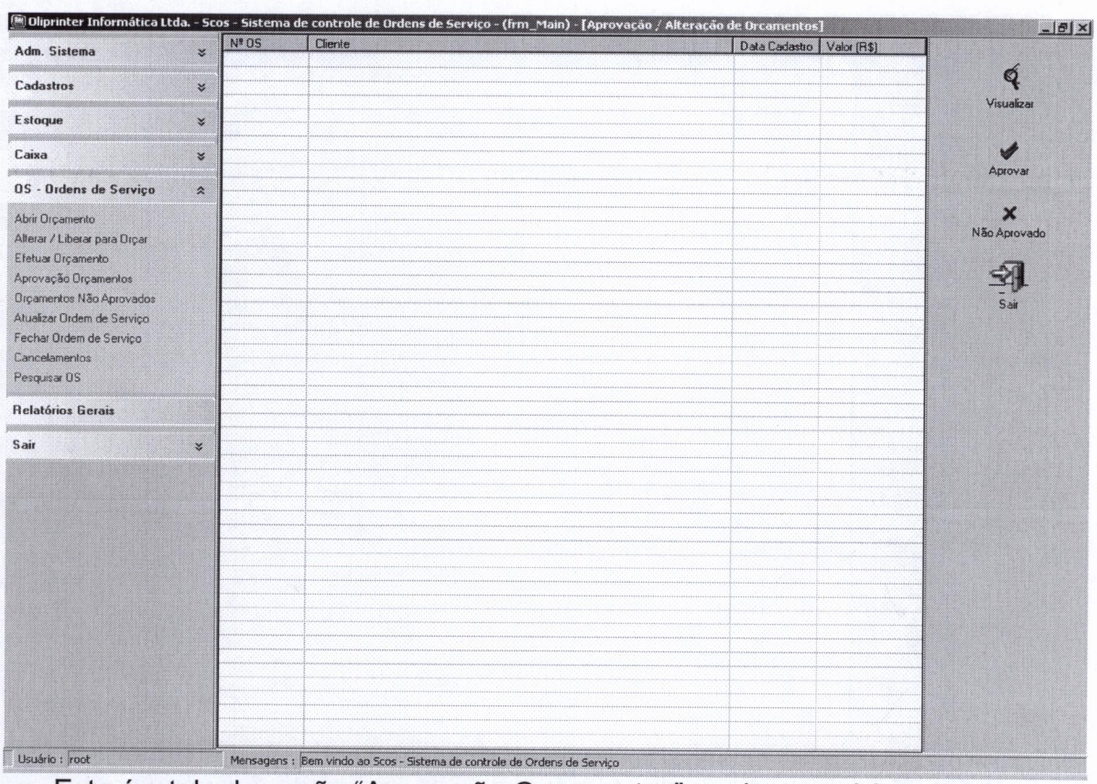
Está é a tela da opção “Abrir Orçamentos”, onde o usuário poderá fazer uma pesquisa através do CPF ou da Razão Social para selecionar o cliente.



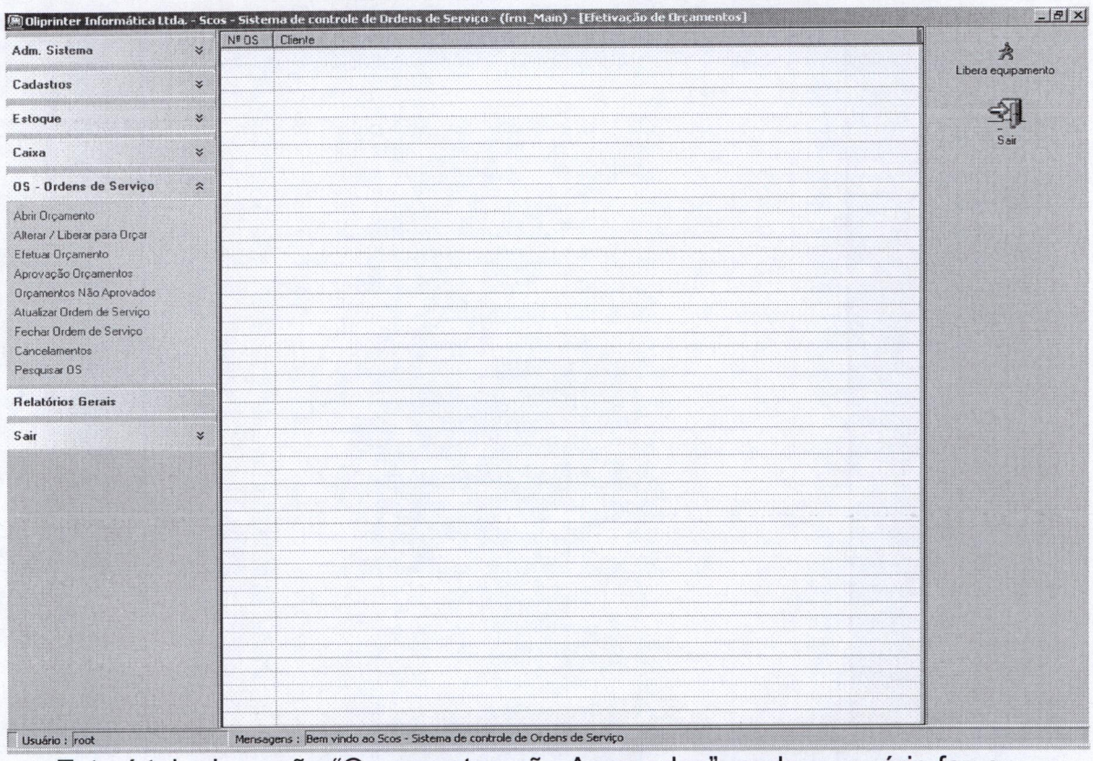
Esta é a tela da opção “Alterar / Liberar para Orçar”, onde o usuário poderá alterar, liberar ou até cancelar orçamentos.



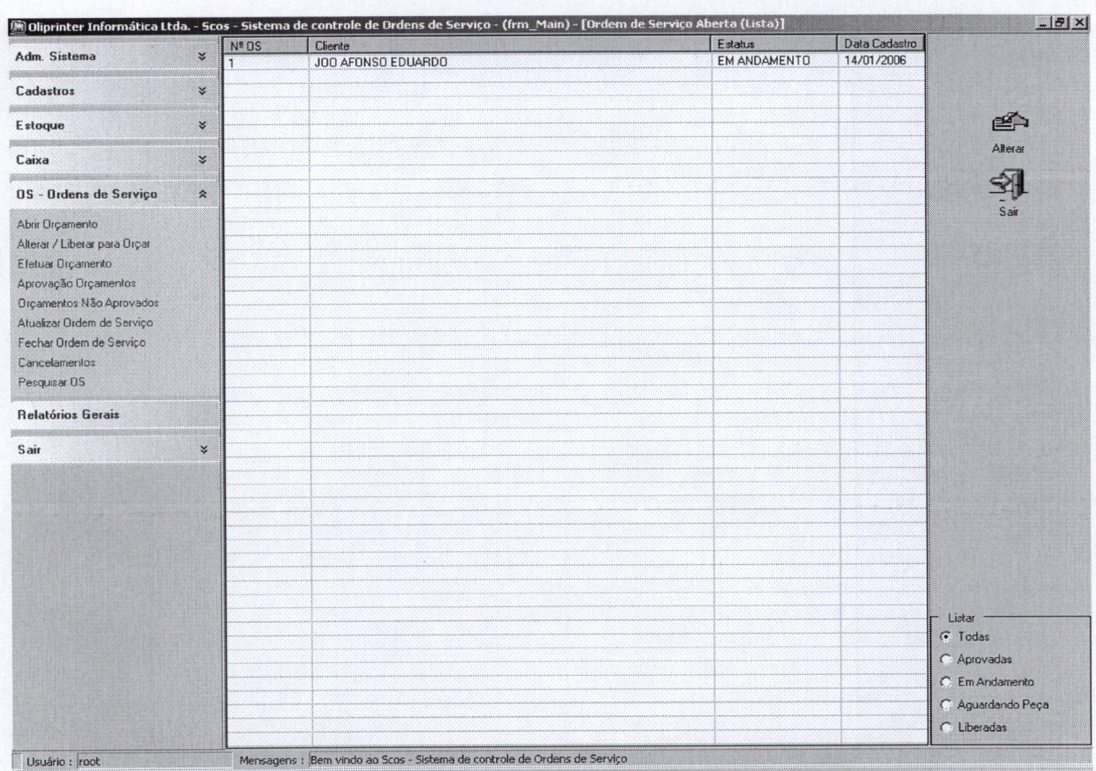
Esta é a tela da opção “Efetuar Orçamentos”, onde o usuário poderá alterar ou descrever que o orçamento esta completo.



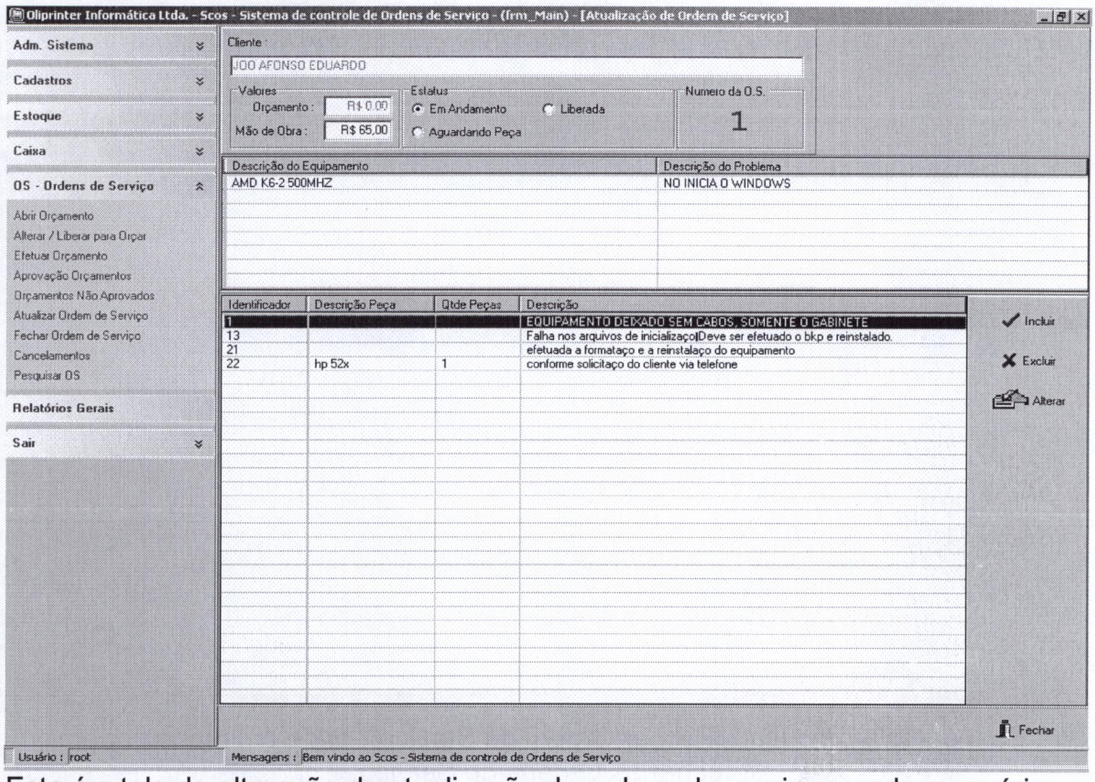
Esta é a tela da opção “Aprovação Orçamentos”, onde o usuário poderá visualizar, definir como aprovado ou não aprovado os orçamentos.



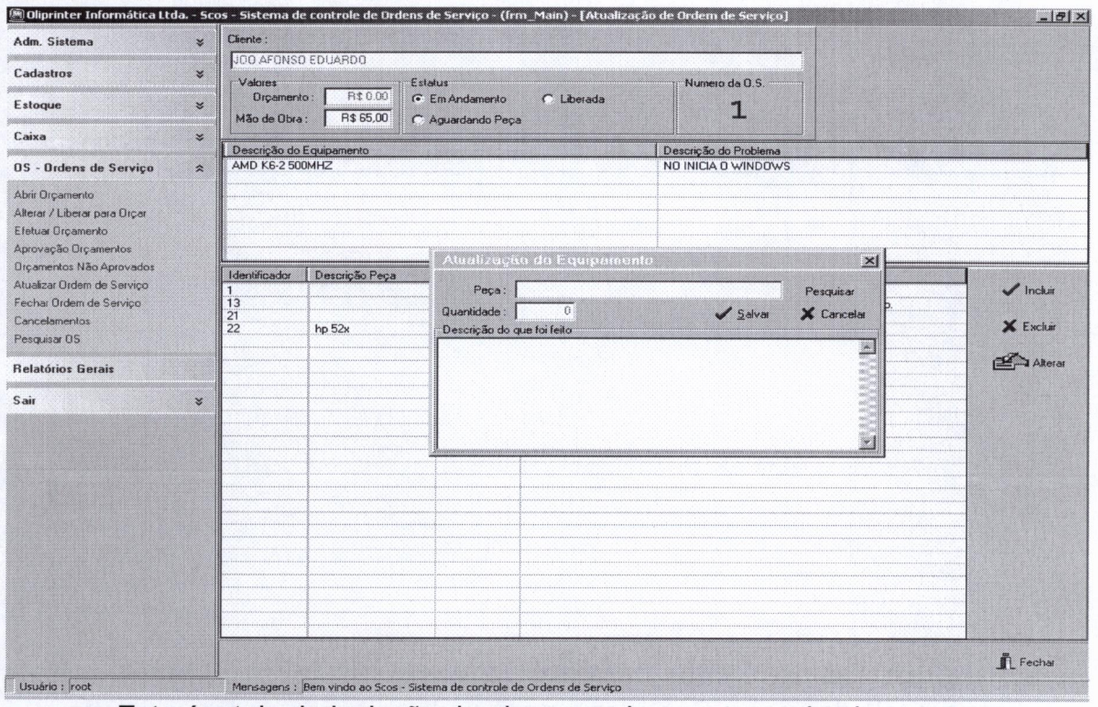
Esta é tela da opção “Orçamentos não Aprovados”, onde o usuário faz a liberação dos equipamentos que não foram aprovados.



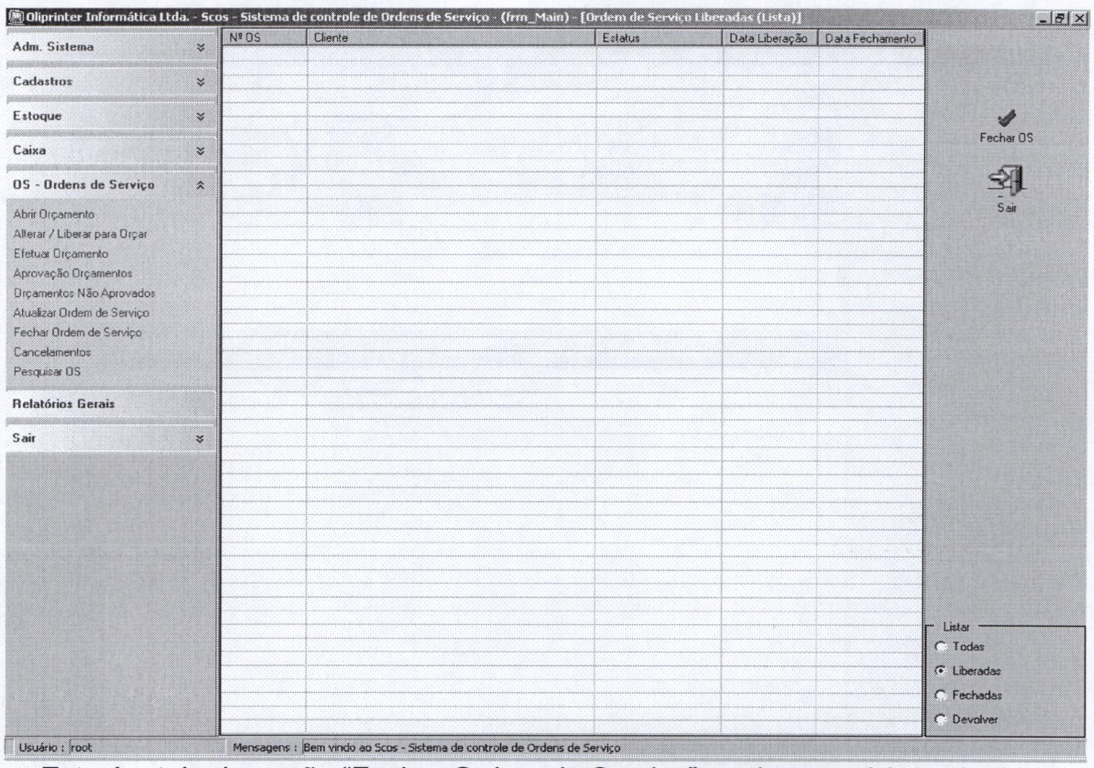
Esta é a tela da opção “Atualizar Ordem de Serviço”, onde o usuário poderá atualizar e alterar ordens de serviços.



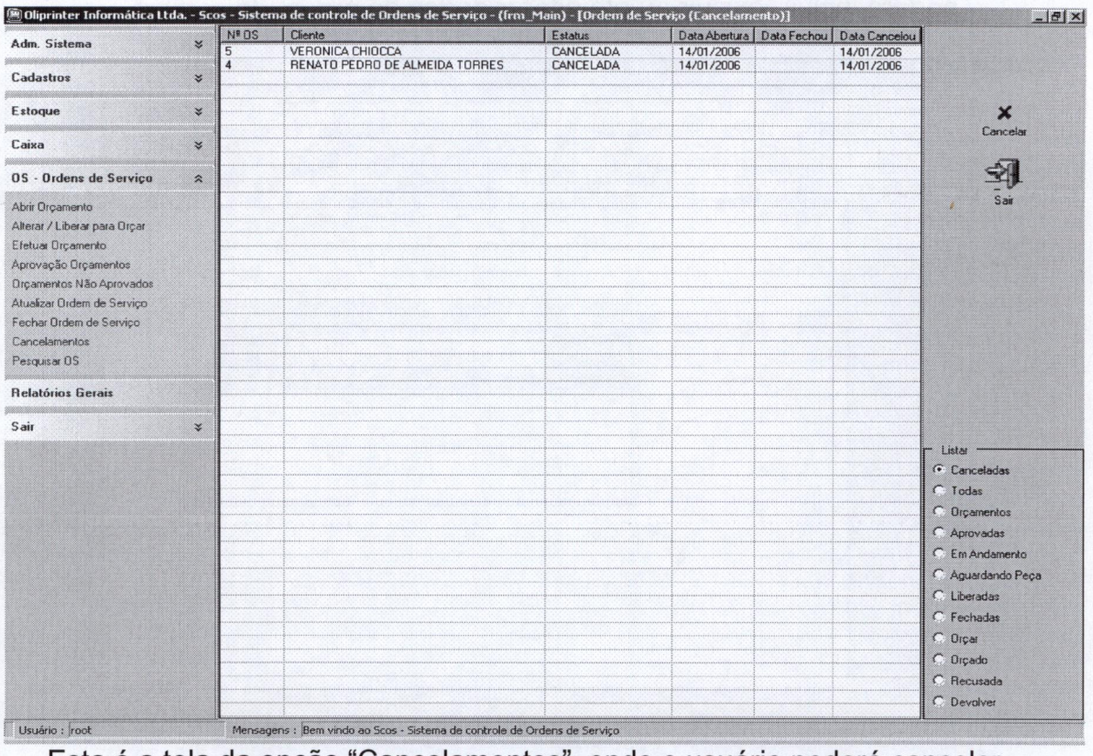
Esta é a tela de alteração da atualização de ordens de serviços, onde o usuário poderá incluir, excluir ou até alterar dados da ordem de serviço.



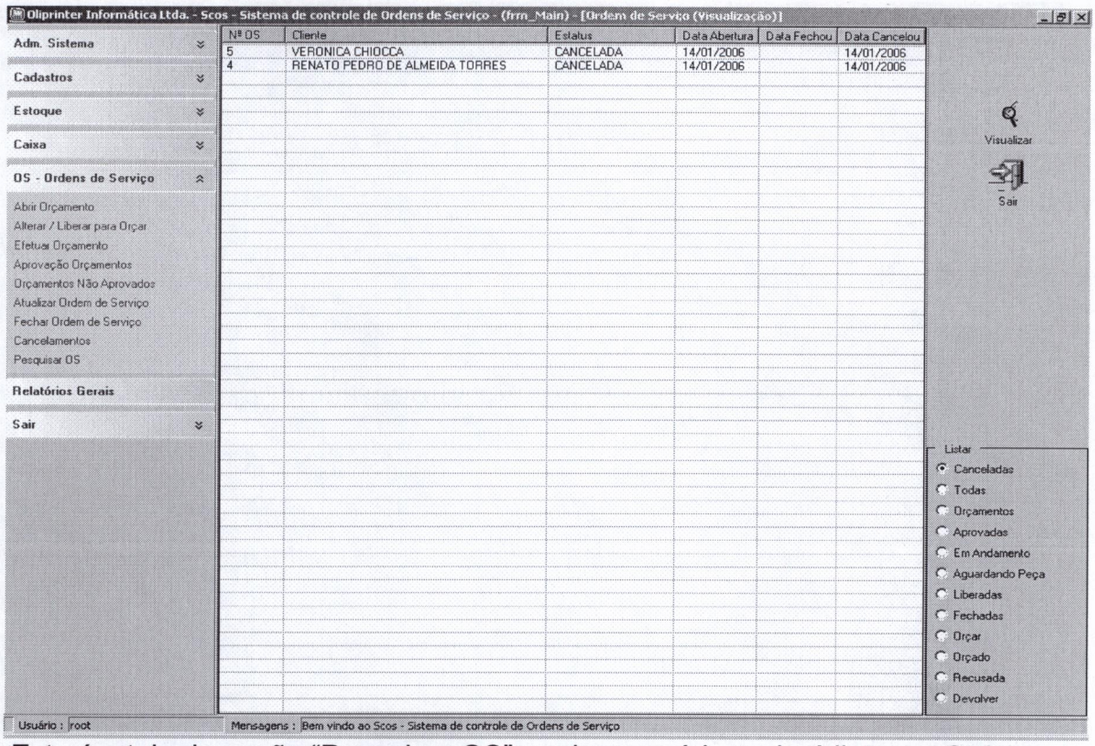
Esta é a tela de inclusão de algum equipamento ou de alguma anotação/descrição dentro da ordem de serviço.



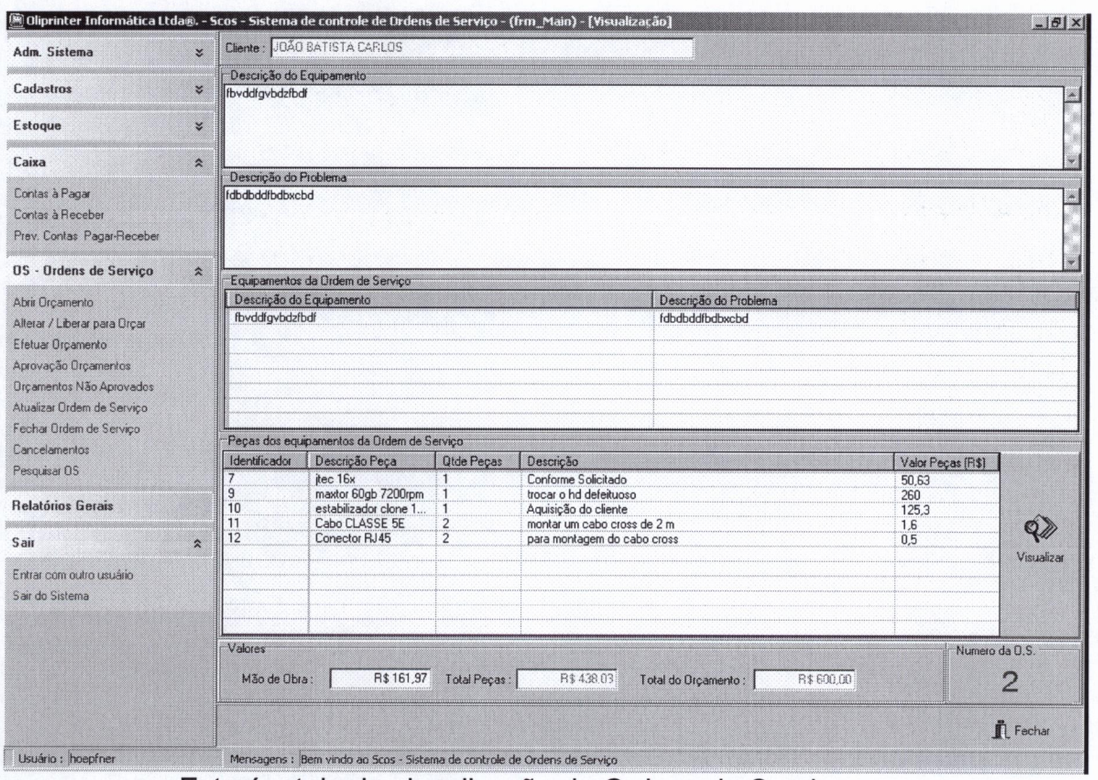
Esta é a tela da opção “Fechar Ordem de Serviço”, onde o usuário poderá fechar as Ordens de Serviço.



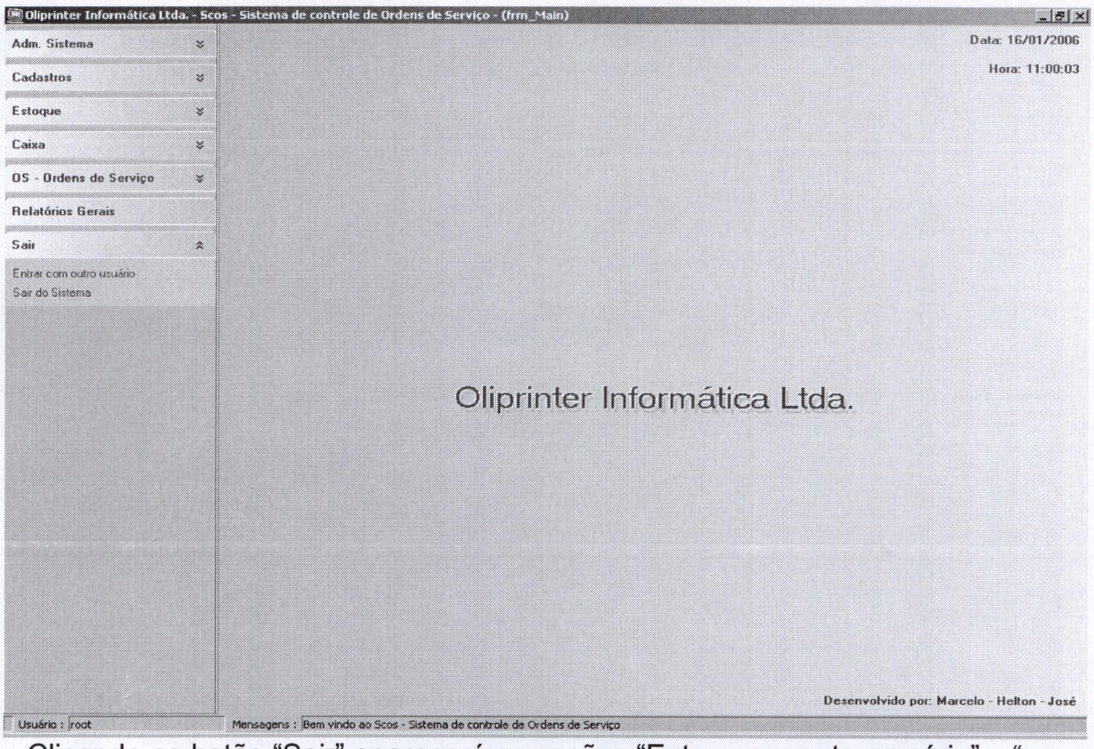
Esta é a tela da opção “Cancelamentos”, onde o usuário poderá cancelar Ordens de Serviço.



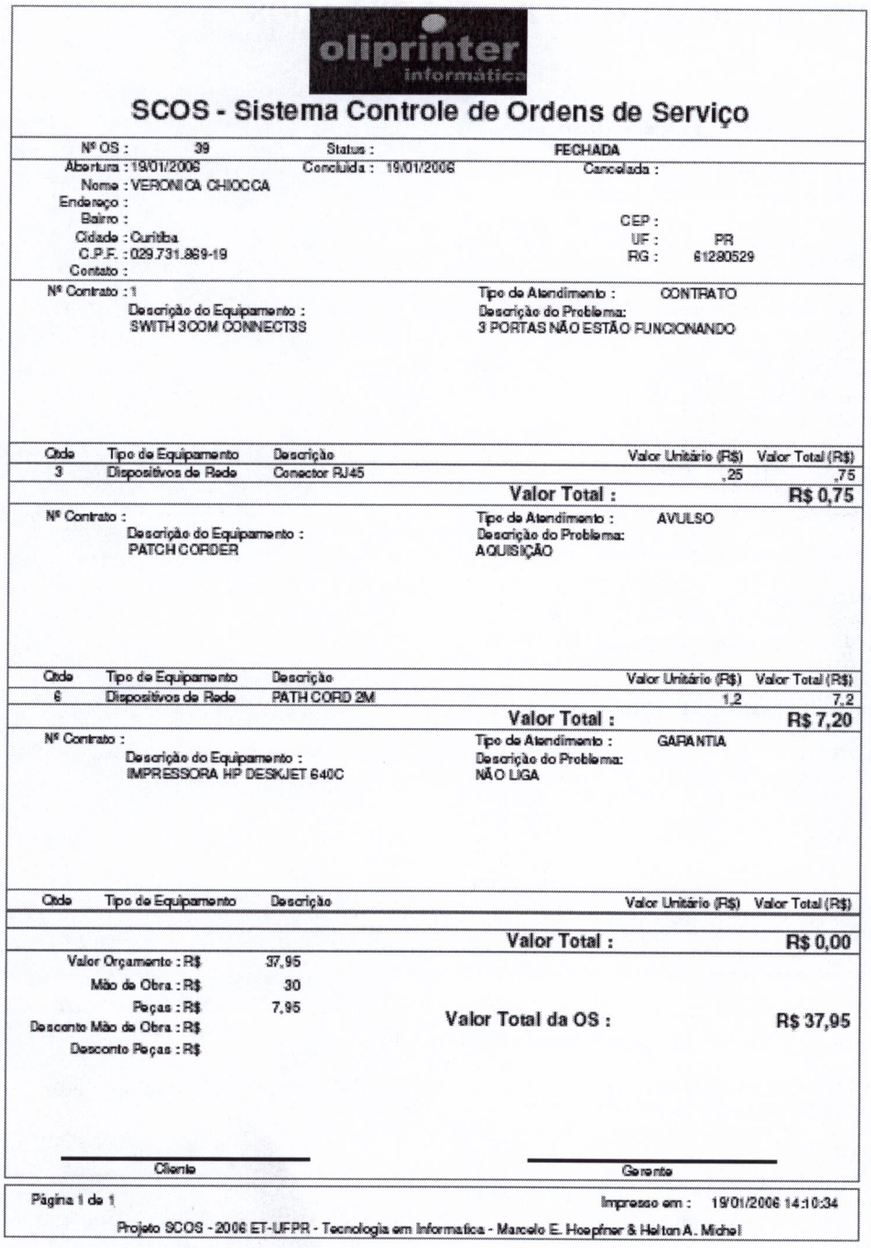
Esta é a tela da opção “Pesquisar OS”, onde o usuário poderá listar as Ordens de Serviços e visualizas.



Esta é a tela da visualização de Ordens de Serviços.



Clicando no botão “Sair” aparecerá as opções “Entrar com outro usuário” e “ Sair do Sistema”.



oliprinter informática

SCOS - Sistema Controle de Ordens de Serviço

Nº OS : 39 Status : FECHADA

Abertura : 19/01/2006 Concluída : 19/01/2006 Cancelada :

Nome : VERONICA CHIOCCA

Endereço :

Bairro : CEP :

Cidade : Curitiba UF : PR

C.P.F. : 029.731.889-19 RG : 61280529

Contato :

Nº Contrato : 1 Tipo de Atendimento : CONTRATO

Descrição do Equipamento : SWITH 3COM CONNECT3S

Descrição do Problema: 3 PORTAS NÃO ESTÃO FUNCIONANDO

Qtde	Tipo de Equipamento	Descrição	Valor Unitário (R$)	Valor Total (R$)
3	Dispositivos de Rede	Conector RJ45	,25	,75

Valor Total : R$ 0,75

Nº Contrato : Tipo de Atendimento : AVULSO

Descrição do Equipamento : PATCH CORDER

Descrição do Problema: AQUISIÇÃO

Qtde	Tipo de Equipamento	Descrição	Valor Unitário (R$)	Valor Total (R$)
6	Dispositivos de Rede	PATH CORD 2M	1,2	7,2

Valor Total : R$ 7,20

Nº Contrato : Tipo de Atendimento : GARANTIA

Descrição do Equipamento : IMPRESSORA HP DESKJET 640C

Descrição do Problema: NÃO LIGA

Qtde	Tipo de Equipamento	Descrição	Valor Unitário (R$)	Valor Total (R$)

Valor Total : R$ 0,00

Valor Orçamento : R$ 37,95

Mão de Obra : R$ 30

Peças : R$ 7,95

Desconto Mão de Obra : R$

Desconto Peças : R$

Valor Total da OS : R$ 37,95

Cliente

Gerente

Página 1 de 1 Impresso em : 19/01/2006 14:10:34

Projeto SCOS - 2006 ET-UFPR - Tecnologia em Informatica - Marcelo E. Hoepfner & Helton A. Michel

Este é o relatório emitido pelo sistema a respeito das OS

15 SCRIPT DE CRIAÇÃO DO BANCO DE DADOS

```
--
-- Structure for table tipomov (OID = 39073) :
--
CREATE TABLE tipomov (
    id_tipomov sequence NOT NULL,
    tipomov integer,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table menu (OID = 39076) :
--
CREATE TABLE menu (
    id_menu character varying(50) NOT NULL,
    tipo_menu integer NOT NULL,
    id_modulo integer NOT NULL,
    sequencia integer NOT NULL,
    menupai integer NOT NULL,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table grupo (OID = 39080) :
--
CREATE TABLE grupo (
    id_grupo sequence NOT NULL,
    nomegrupo character varying(30) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table uf (OID = 39085) :
--
CREATE TABLE uf (
    id_uf sequence NOT NULL,
    abreviada character varying(50) NOT NULL,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table tipo_peca (OID = 39090) :
--
CREATE TABLE tipo_peca (
    id_tipo_peca sequence NOT NULL,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table tipo_equipamento (OID = 39095) :
--
CREATE TABLE tipo_equipamento (
    id_tipo_equipamento sequence NOT NULL,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table usuario (OID = 39100) :
--
CREATE TABLE usuario (
    id_usuario sequence NOT NULL,
    id_grupo integer NOT NULL,
    nome character varying(30) NOT NULL,
    nomecompleto character varying(50) NOT NULL,
    senha character varying(30) NOT NULL,
    status integer NOT NULL,
    ramal character varying(30) NOT NULL,
    email character varying(100) NOT NULL,
    cargo character varying(50) NOT NULL,
    datacadastro date NOT NULL,
    dataultacesso date NOT NULL
) WITHOUT OIDS;
--
-- Structure for table peca (OID = 39105) :
--
CREATE TABLE peca (
    id_peca sequence NOT NULL,
    id_tipo_peca integer NOT NULL,
    descricao character varying(50) NOT NULL,
```

```
    quantidade integer NOT NULL,
    valor_unitario double precision
) WITHOUT OIDS;
--
-- Structure for table equipamento (OID = 39110) :
--
CREATE TABLE equipamento (
    id_equipamento sequence NOT NULL,
    id_tipo_equipamento integer NOT NULL,
    id_contrato integer NOT NULL,
    descricao text NOT NULL,
    numero_serie character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table cidade (OID = 39118) :
--
CREATE TABLE cidade (
    id_cidade sequence NOT NULL,
    id_uf integer NOT NULL,
    descricao character varying(50) NOT NULL
) WITHOUT OIDS;
--
-- Structure for table acesso (OID = 39121) :
--
CREATE TABLE acesso (
    id_usuario integer NOT NULL,
    id_menu character varying(50) NOT NULL,
    nivel integer NOT NULL
) WITHOUT OIDS;
--
-- Structure for table pessoa (OID = 39125) :
--
CREATE TABLE pessoa (
    id_pessoa sequence NOT NULL,
    id_cidade integer NOT NULL,
    nome character varying(40) NOT NULL,
    endereco character varying(40),
    bairro character varying(30),
    cep character varying(9),
    cgc_cnpj character varying(18),
    ie_rg character varying(18),
    contato character varying(30),
    email character varying(100),
    homepage character varying(100),
    tipo_pessoa integer NOT NULL,
    tipo integer NOT NULL,
    id_uf integer NOT NULL,
    obs text
) WITHOUT OIDS;
--
-- Structure for table os (OID = 39133) :
--
CREATE TABLE os (
    id_os sequence NOT NULL,
    data_cad date NOT NULL,
    data_alt date,
    data_fechou date,
    data_canc date,
    id_pessoa integer NOT NULL,
    status text NOT NULL,
    valor_pecas numeric,
    valor_orcado numeric,
    valor_cobrado numeric,
    valor_mao_obra numeric,
    valor_pecas_ncobrado numeric,
    valor_desconto_mo numeric,
    valor_desconto_pecas numeric
) WITHOUT OIDS;
--
-- Structure for table peca_fornec (OID = 39141) :
--
CREATE TABLE peca_fornec (
    id_pessoa sequence NOT NULL,
    id_peca integer NOT NULL
) WITHOUT OIDS;
--
-- Structure for table adm_sysfile (OID = 39144) :
```

```
--
CREATE TABLE adm_sysfile (
    nomeempresa character varying(60),
    nomesistema character varying(60),
    desenvolvedores character varying(100),
    statussistema character varying(1) DEFAULT 1,
    montamenu character varying(1) DEFAULT 0
) WITHOUT OIDS;
--
-- Structure for table contatos (OID = 39148) :
--
CREATE TABLE contatos (
    id_pessoa integer NOT NULL,
    nome character varying(30) NOT NULL,
    dia_nasc character varying(2),
    mes_nasc character varying(2),
    cargo character varying(30),
    setor character varying(30),
    email character varying(100)
) WITHOUT OIDS;
--
-- Structure for table contrato (OID = 39152) :
--
CREATE TABLE contrato (
    id_contrato sequence NOT NULL,
    id_pessoa integer NOT NULL,
    data_inicio_contrato date NOT NULL,
    data_fim_vigencia date NOT NULL,
    data_inicio_vigencia date NOT NULL,
    data_bloqueio date,
    valor_mensal numeric(10,2) DEFAULT 0,
    dia_vcto integer NOT NULL,
    dias_prorog integer DEFAULT 0,
    bloqueado integer DEFAULT 0 NOT NULL
) WITHOUT OIDS;
--
-- Structure for table item_os (OID = 39160) :
--
CREATE TABLE item_os (
    id_os integer NOT NULL,
    id_item_os sequence NOT NULL,
    id_contrato integer,
    id_equipamento integer,
    tipo_atendimento integer NOT NULL,
    desc_equip text,
    desc_prob text
) WITHOUT OIDS;
--
-- Structure for table item_os_mov (OID = 39168) :
--
CREATE TABLE item_os_mov (
    id_os integer NOT NULL,
    id_item_os integer NOT NULL,
    id_sitem sequence NOT NULL,
    id_peca integer,
    qtde_peca integer,
    descricao text,
    cobrar integer DEFAULT 1
) WITHOUT OIDS;
--
-- Structure for table telefones (OID = 39177) :
--
CREATE TABLE telefones (
    id_fone sequence NOT NULL,
    nome character varying(30) NOT NULL,
    telefone character varying(20) NOT NULL,
    tipo_fone character varying(20) NOT NULL,
    id_pessoa integer NOT NULL
) WITHOUT OIDS;
--
-- Structure for table caixa (OID = 39187) :
--
CREATE TABLE caixa (
    id_lcto sequence NOT NULL,
    num_doc integer,
    id_tipomov integer NOT NULL,
    id_pessoa integer,
```

```
    num_parcela integer,
    data_emissao date NOT NULL,
    data_vcto date NOT NULL,
    data_pgto date,
    data_prorrog date,
    tipo_pgto integer NOT NULL,
    valor_apagar numeric NOT NULL,
    valor_juro numeric,
    valor_pago numeric,
    situacao character varying(50) NOT NULL,
    obs text
) WITHOUT OIDS;
--
-- Structure for table logs_os (OID = 39336) :
--
CREATE TABLE logs_os (
    id_log sequence NOT NULL,
    id_os integer NOT NULL,
    id_usuario integer NOT NULL,
    acao text NOT NULL,
    data date NOT NULL
) WITHOUT OIDS;
--
-- Definition for index cidade_id_cidade_key (OID = 39197) :
--
CREATE UNIQUE INDEX cidade_id_cidade_key ON cidade USING btree (id_cidade);
--
-- Definition for index spatial_ref_sys_pkey (OID = 17287) :
--
ALTER TABLE ONLY spatial_ref_sys
    ADD CONSTRAINT spatial_ref_sys_pkey PRIMARY KEY (srid);
--
-- Definition for index geometry_columns_pk (OID = 17291) :
--
ALTER TABLE ONLY geometry_columns
    ADD CONSTRAINT geometry_columns_pk PRIMARY KEY (f_table_catalog, f_table_schema,
f_table_name, f_geometry_column);
--
-- Definition for index tipomov_pkey (OID = 39198) :
--
ALTER TABLE ONLY tipomov
    ADD CONSTRAINT tipomov_pkey PRIMARY KEY (id_tipomov);
--
-- Definition for index menu_pkey (OID = 39200) :
--
ALTER TABLE ONLY menu
    ADD CONSTRAINT menu_pkey PRIMARY KEY (id_menu);
--
-- Definition for index grupo_pkey (OID = 39202) :
--
ALTER TABLE ONLY grupo
    ADD CONSTRAINT grupo_pkey PRIMARY KEY (id_grupo);
--
-- Definition for index uf_pkey (OID = 39204) :
--
ALTER TABLE ONLY uf
    ADD CONSTRAINT uf_pkey PRIMARY KEY (id_uf);
--
-- Definition for index tipo_peca_pkey (OID = 39206) :
--
ALTER TABLE ONLY tipo_peca
    ADD CONSTRAINT tipo_peca_pkey PRIMARY KEY (id_tipo_peca);
--
-- Definition for index tipo_equipamento_pkey (OID = 39208) :
--
ALTER TABLE ONLY tipo_equipamento
    ADD CONSTRAINT tipo_equipamento_pkey PRIMARY KEY (id_tipo_equipamento);
--
-- Definition for index usuario_pkey (OID = 39210) :
--
ALTER TABLE ONLY usuario
    ADD CONSTRAINT usuario_pkey PRIMARY KEY (id_usuario);
--
-- Definition for index peca_pkey (OID = 39212) :
--
ALTER TABLE ONLY peca
    ADD CONSTRAINT peca_pkey PRIMARY KEY (id_peca);
```

```
--
-- Definition for index equipamento_pkey (OID = 39214) :
--
ALTER TABLE ONLY equipamento
    ADD CONSTRAINT equipamento_pkey PRIMARY KEY (id_equipamento);
--
-- Definition for index cidade_pkey (OID = 39216) :
--
ALTER TABLE ONLY cidade
    ADD CONSTRAINT cidade_pkey PRIMARY KEY (id_cidade, id_uf);
--
-- Definition for index acesso_pkey (OID = 39218) :
--
ALTER TABLE ONLY acesso
    ADD CONSTRAINT acesso_pkey PRIMARY KEY (id_usuario, id_menu);
--
-- Definition for index pessoa_pkey (OID = 39220) :
--
ALTER TABLE ONLY pessoa
    ADD CONSTRAINT pessoa_pkey PRIMARY KEY (id_pessoa);
--
-- Definition for index os_pkey (OID = 39222) :
--
ALTER TABLE ONLY os
    ADD CONSTRAINT os_pkey PRIMARY KEY (id_os);
--
-- Definition for index peca_fornec_pkey (OID = 39224) :
--
ALTER TABLE ONLY peca_fornec
    ADD CONSTRAINT peca_fornec_pkey PRIMARY KEY (id_pessoa, id_peca);
--
-- Definition for index rel_01 (OID = 39226) :
--
ALTER TABLE ONLY usuario
    ADD CONSTRAINT rel_01 FOREIGN KEY (id_grupo) REFERENCES grupo(id_grupo) ON UPDATE
RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_02 (OID = 39230) :
--
ALTER TABLE ONLY peca
    ADD CONSTRAINT rel_02 FOREIGN KEY (id_tipo_peca) REFERENCES tipo_peca(id_tipo_peca)
ON UPDATE RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_04 (OID = 39234) :
--
ALTER TABLE ONLY cidade
    ADD CONSTRAINT rel_04 FOREIGN KEY (id_uf) REFERENCES uf(id_uf) ON UPDATE RESTRICT ON
DELETE RESTRICT;
--
-- Definition for index rel_17 (OID = 39238) :
--
ALTER TABLE ONLY peca_fornec
    ADD CONSTRAINT rel_17 FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa) ON UPDATE
RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_18 (OID = 39242) :
--
ALTER TABLE ONLY peca_fornec
    ADD CONSTRAINT rel_18 FOREIGN KEY (id_peca) REFERENCES peca(id_peca) ON UPDATE
RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_07 (OID = 39246) :
--
ALTER TABLE ONLY acesso
    ADD CONSTRAINT rel_07 FOREIGN KEY (id_menu) REFERENCES menu(id_menu);
--
-- Definition for index contatos_pkey (OID = 39250) :
--
ALTER TABLE ONLY contatos
    ADD CONSTRAINT contatos_pkey PRIMARY KEY (id_pessoa, nome);
--
-- Definition for index contrato_pkey (OID = 39252) :
--
ALTER TABLE ONLY contrato
    ADD CONSTRAINT contrato_pkey PRIMARY KEY (id_contrato);
--
-- Definition for index os_id_pessoa_fkey (OID = 39254) :
```

```
--
ALTER TABLE ONLY os
    ADD CONSTRAINT os_id_pessoa_fkey FOREIGN KEY (id_pessoa) REFERENCES
pessoa(id_pessoa) ON UPDATE RESTRICT ON DELETE RESTRICT;
--
-- Definition for index item_os_pkey (OID = 39258) :
--
ALTER TABLE ONLY item_os
    ADD CONSTRAINT item_os_pkey PRIMARY KEY (id_os, id_item_os);
--
-- Definition for index item_os_id_contrato_fkey (OID = 39260) :
--
ALTER TABLE ONLY item_os
    ADD CONSTRAINT item_os_id_contrato_fkey FOREIGN KEY (id_contrato) REFERENCES
contrato(id_contrato) ON UPDATE RESTRICT ON DELETE RESTRICT;
--
-- Definition for index item_os_id_equipamento_fkey (OID = 39264) :
--
ALTER TABLE ONLY item_os
    ADD CONSTRAINT item_os_id_equipamento_fkey FOREIGN KEY (id_equipamento) REFERENCES
equipamento(id_equipamento);
--
-- Definition for index item_os_mov_id_peca_fkey (OID = 39268) :
--
ALTER TABLE ONLY item_os_mov
    ADD CONSTRAINT item_os_mov_id_peca_fkey FOREIGN KEY (id_peca) REFERENCES
peca(id_peca);
--
-- Definition for index item_os_mov_id_os_fkey (OID = 39272) :
--
ALTER TABLE ONLY item_os_mov
    ADD CONSTRAINT item_os_mov_id_os_fkey FOREIGN KEY (id_os, id_item_os) REFERENCES
item_os(id_os, id_item_os) ON UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for index item_os_mov_pkey (OID = 39276) :
--
ALTER TABLE ONLY item_os_mov
    ADD CONSTRAINT item_os_mov_pkey PRIMARY KEY (id_os, id_item_os, id_sitem);
--
-- Definition for index item_os_id_os_fkey (OID = 39278) :
--
ALTER TABLE ONLY item_os
    ADD CONSTRAINT item_os_id_os_fkey FOREIGN KEY (id_os) REFERENCES os(id_os) ON UPDATE
CASCADE ON DELETE CASCADE;
--
-- Definition for index rel_08 (OID = 39282) :
--
ALTER TABLE ONLY acesso
    ADD CONSTRAINT rel_08 FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario) ON
UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for index telefones_pkey (OID = 39286) :
--
ALTER TABLE ONLY telefones
    ADD CONSTRAINT telefones_pkey PRIMARY KEY (id_fone, id_pessoa, nome);
--
-- Definition for index pessoa_id_cidade_fkey (OID = 39288) :
--
ALTER TABLE ONLY pessoa
    ADD CONSTRAINT pessoa_id_cidade_fkey FOREIGN KEY (id_cidade) REFERENCES
cidade(id_cidade) ON UPDATE RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_25 (OID = 39302) :
--
ALTER TABLE ONLY telefones
    ADD CONSTRAINT rel_25 FOREIGN KEY (id_pessoa, nome) REFERENCES contatos(id_pessoa,
nome) ON UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for index rel_12 (OID = 39306) :
--
ALTER TABLE ONLY contatos
    ADD CONSTRAINT rel_12 FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa) ON UPDATE
CASCADE ON DELETE CASCADE;
--
-- Definition for index rel_11 (OID = 39310) :
--
ALTER TABLE ONLY contrato
```

```
    ADD CONSTRAINT rel_11 FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa) ON UPDATE
CASCADE ON DELETE CASCADE;
--
-- Definition for index rel_03 (OID = 39314) :
--
ALTER TABLE ONLY equipamento
    ADD CONSTRAINT rel_03 FOREIGN KEY (id_tipo_equipamento) REFERENCES
tipo_equipamento(id_tipo_equipamento) ON UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for index equipamento_id_contrato_fkey (OID = 39318) :
--
ALTER TABLE ONLY equipamento
    ADD CONSTRAINT equipamento_id_contrato_fkey FOREIGN KEY (id_contrato) REFERENCES
contrato(id_contrato) ON UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for index caixa_pkey (OID = 39322) :
--
ALTER TABLE ONLY caixa
    ADD CONSTRAINT caixa_pkey PRIMARY KEY (id_lcto);
--
-- Definition for index rel_14 (OID = 39324) :
--
ALTER TABLE ONLY caixa
    ADD CONSTRAINT rel_14 FOREIGN KEY (id_tipomov) REFERENCES tipomov(id_tipomov) ON
UPDATE RESTRICT ON DELETE RESTRICT;
--
-- Definition for index rel_60 (OID = 39328) :
--
ALTER TABLE ONLY caixa
    ADD CONSTRAINT rel_60 FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa) ON UPDATE
RESTRICT ON DELETE RESTRICT;
--
-- Definition for index logs_os_pkey (OID = 39342) :
--
ALTER TABLE ONLY logs_os
    ADD CONSTRAINT logs_os_pkey PRIMARY KEY (id_log);
--
-- Definition for index rel_15 (OID = 39344) :
--
ALTER TABLE ONLY logs_os
    ADD CONSTRAINT rel_15 FOREIGN KEY (id_os) REFERENCES os(id_os) ON UPDATE CASCADE ON
DELETE CASCADE;
--
-- Definition for index rel_16 (OID = 39348) :
--
ALTER TABLE ONLY logs_os
    ADD CONSTRAINT rel_16 FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario) ON
UPDATE CASCADE ON DELETE CASCADE;
--
-- Definition for view rel_os (OID = 39193) :
--
SET search_path = public, pg_catalog;
CREATE VIEW rel_os AS
    SELECT os.id_os, os.data_cad, os.data_fechou, os.status, cli.nome,
        cli.endereco, cli.bairro, cli.cep, cde.descricao AS cde_descr,
        uf.abreviada, cli.cgc_cnpj, cli.ie_rg, cli.contato, ios.id_item_os,
        ios.id_contrato, ios.id_equipamento, ios.tipo_atendimento,
        ios.desc_equip, ios.desc_prob, sos.id_sitem, sos.qtde_peca,
        tpc.descricao AS tpc_descr, pc.descricao AS pc_descr,
        pc.valor_unitario, sos.cobrar, os.data_canc, os.valor_pecas,
        os.valor_orcado, os.valor_cobrado, os.valor_mao_obra,
        os.valor_pecas_ncobrado, os.valor_desconto_mo,
        os.valor_desconto_pecas, eqp.numero_serie, eqp.descricao AS
        eqp_descr, ((sos.qtde_peca)::double precision * pc.valor_unitario)
        AS vlr_itot_peca
    FROM (((((item_os_mov sos RIGHT JOIN item_os ios ON (((sos.id_item_os =
        ios.id_item_os) AND (sos.id_os = ios.id_os)))) JOIN (os os JOIN
        ((pessoa cli JOIN cidade cde ON (((cli.id_uf = cde.id_uf) AND
        (cli.id_cidade = cde.id_cidade)))) JOIN uf uf ON ((cde.id_uf =
        uf.id_uf))) ON ((os.id_pessoa = cli.id_pessoa))) ON ((ios.id_os =
        os.id_os))) LEFT JOIN peca pc ON ((sos.id_peca = pc.id_peca))) LEFT
        JOIN tipo_peca tpc ON ((pc.id_tipo_peca = tpc.id_tipo_peca))) LEFT
        JOIN equipamento eqp ON (((ios.id_contrato = eqp.id_contrato) AND
        (ios.id_equipamento = eqp.id_equipamento))))
    ORDER BY os.id_os, ios.id_item_os, sos.id_sitem;
COMMENT ON SCHEMA public IS 'Standard public schema';
```

16 INSTALAÇÃO DO SISTEMA

1- Execute o arquivo "postgresql-8.0.msi" e siga as instrucoes de instalacao;
(Para dúvidas na instalacao do arquivo, consulte: http://www.postgresql.org/);

2- Extraia o conteudo do arquivo "PgOleDb.zip" para a pasta System32 (usualmente: c:\windows\system32);

3- Abra o programa "pg-AdminIII", selecione "PostgreSQL Database Server 8.0" e digite a senha para conectar ao banco.

4- Em "Databases", clique com o botao direito e selecione "New Database", preencha os campos: Name: "scos", Owner: "postgres", Encoding: "Latin2". Confirme em OK.

5- Será criado o banco "scos". Selecione-o e clique no ícone representado por um lápis com os escritos "sql", selecione o menu "File", clique em "Open", selecione o arquivo "scos.sql", clique em "Abrir" e pressione "F5".

6- O banco está instalado. Feche o "pg-AdminIII" e execute o aplicativo "SCOS.exe".

7- Entre com o usuário e senha.

17 IMPLEMENTAÇÕES FUTURAS

Para permitir que novas implementações possam ser adicionadas com facilidade, o sistema SCOS (Sistema de Cadastro de Ordem de Serviço) foi desenvovido utilizando interfaces limpas, baseando em poucas telas e código fonte comentado.

Dentre as funções que podem ser implementadas, destacam-se:

- Módulo de backup personalizado, permitindo a configuração de políticas de retenção e métodos de Backup Full, Incremental e Diferencial.

- Módulo de geração de notas fiscais para faturamento, compatibilizando de acordo com as normas federais e estaduais, permitindo o uso de impressoras para Emissão de Cupom Fiscal (ECF).

- Módulo WEB, para consultas e alterações por parte do usuário, permitindo acompanhar o andamento em tempo real, melhorando a qualidade da relação empresa-cliente, através de um processo transparente.

- Módulo de Email, permitindo, em conjunto com o Módulo WEB, efetuar o envio de mensagens eletrônicas que permitam ao cliente, verificar o andamento do processo de execução da Ordem de Serviço.

18 AMBIENTE DE PRODUÇÃO

18.1 SERVIDOR

Sistema operacional: Linux / Windows(2000,2003,xp)

512MB Memória (recomendado)

5gb de Espaço em Disco

Mínimo 850Mhzs

18.2 CLIENTE

Sistema Operacional: 98, windows 2000, xp, vista

128MB Memória

30MB de Espaço em Disco

Mínimo 233mhzs

19 CONCLUSÃO

Desenvolvido com base em necessidades de um Cliente Real, o sistema SCOS (Sistema de Cadastro de Ordem de Serviço), procurou atender os objetivos propostos da maneira mais objetiva, utilizando-se de tecnologias atuais e tendo o cuidado de observar as dificuldades mais corriqueiras enfrentadas pelo usuário, tais como o uso de uma interface limpa e adequada e o posicionamento de itens de menu, botões e campos em locais visíveis e de fácil memorização.

Dentro das características propostas, pode-se afirmar, que a solução atende com grande competência aos requisitos, possuindo um banco de dados robusto e escalável, tendo, inclusive, a possibilidade de utilização remota, tanto para consultas, como para uso efetivo.

Como definição de estudo, possibilitou aos integrantes agregar uma gama de conhecimento de uma amplitude permissível a execução de projetos que não tão somente visem o âmbito funcional, mas que também tenham abrangência na questão da usabilidade, facilidade de aprendizado e inclusão de novas implementações, atestando a importância e o entendimento das questões propostas nas disciplinas cursadas ao longo do curso de Tecnologia Em Informática.

20 REFERÊNCIAS BIBLIOGRÁFICAS

PRESSMAN, Roger S. **Engenharia de Software**. 3. ed. São Paulo: Makron Books, 1995. 1056 p.

RUMBAUGH, James e BLAHA Michael. **Modelagem e Projetos Baseados em Objetos**. 11° ed. Rio de Janeiro: Campus , 1994.

DEBONI, José E. Z. **Modelagem Orientada a Objetos com a UML**. São Paulo: Editora Futura, 2003.

FURLAN, José D. **Modelagem de Objetos Através da UML**. São Paulo: Makron Books, 1998.

BOOCH, Grady e RUMBAUGH, James. **UML - Guia do Usuário**. 11° ed. São Paulo: Campus, 2000.

TEIXEIRA, Steve e PACHECO, Xavier. **Borland Delphi 7 Developer's Guide**. 4° ed. Indianapolis, Indiana, United States of America : SAMS Publishing, 2002.

UNIVERSIDADE FEDERAL DO PARANÁ. Biblioteca Central. **Normas para apresentaąão.** Curitiba: Ed da UFPR, 1992. pt. 2: Teses, dissertações e trabalhos acadêmicos.

1 ANEXOS

2 CÓDIGO FONTE

2.1 CLASSES

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.1 unit TUsuario;

```
interface

uses
   Classes, PSQLDbTables, TGrupo, Controls, SysUtils ;

type
  TCUsuario = class(TComponent)
    private
      //FIELDS
      F_id_usuario        : Integer;
      F_id_grupo          : Integer;
      F_nome              : String;
      F_nomecompleto      : String;
      F_senha             : String;
      F_status            : Integer;
      F_ramal             : String;
      F_email             : String;
      F_cargo             : String;
      F_datacadastro      : TDate;
      F_dataultimoacesso  : TDate;

    public

      Grupo : TCGrupo;

      constructor Create(Dono: TComponent);

      //Property's
      property P_id_usuario        : Integer read F_id_usuario       write F_id_usuario;
      property P_id_grupo          : Integer read F_id_grupo         write F_id_grupo;
      property P_login             : String  read F_nome             write F_nome;
      property P_nomecompleto      : String  read F_nomecompleto     write
F_nomecompleto;
      property P_senha             : String  read F_senha            write F_senha;
      property P_status            : Integer read F_status           write F_status;
      property P_ramal             : String  read F_ramal            write F_ramal;
      property P_email             : String  read F_email            write F_email;
      property P_cargo             : String  read F_cargo            write F_cargo;
```

```
      property P_datacadastro      : TDate   read F_datacadastro     write
F_datacadastro;
      property P_dataultimoacesso  : TDate   read F_dataultimoacesso write
F_dataultimoacesso;


      //Operações
      procedure Carregar(Login: String);
      procedure Carregar_id(Id_DoUsuario: Integer);
      procedure Salvar(Estado: Integer);
      procedure Ativar(wp_id_usuario : Integer);
      procedure Desativar(wp_id_usuario : Integer);


      function  Comeco: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Anterior;
      procedure Ultimo;
      function  Final: Boolean;
      procedure SelecionaAtivos;
      procedure SelecionaItivos;
      procedure SelecionaTodos;
      procedure Coloca;


  end;

 var
 Busca,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;
{ TCUsuario }

procedure TCUsuario.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCUsuario.Ativar(wp_id_usuario: Integer);
begin
   try
      Acao := TPSQLQuery.Create(Self);
      Acao.Database := dm_base.my_conn;
      Acao.Close;
      ws_Sql := ('Update usuario set ');
      ws_Sql := (ws_Sql + 'Status = :p_Status ');
      ws_Sql := (ws_Sql + 'where Id_Usuario = :p_Id_Usuario');
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      Acao.ParamByName('p_id_usuario').Value       := wp_id_usuario;
      Acao.ParamByName('p_Status').Value           := 1;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCUsuario.Carregar(Login: String);
begin
   try
      Busca := TPSQLQuery.Create(Self);
      Busca.Database:= dm_base.my_conn;
      ws_sql:= ('select * from usuario where (nome = :p_nome) order by id_usuario');
      ws_sql := lowercase(ws_sql);
      Busca.SQL.Clear;
      Busca.SQL.Add(ws_sql);
      Busca.ParamByName('p_nome').AsString := Login;
      Busca.Open;
      if busca.IsEmpty then
      begin
         F_id_usuario       := 0;
         F_id_grupo         := 0;
         F_nome             := '';
         F_nomecompleto     := '';
```

```
            F_senha             := '';
            F_status            := 0;
            F_ramal             := '';
            F_email             := '';
            F_cargo             := '';
            F_datacadastro      := 0;
            F_dataultimoacesso  := 0;
         end
         else
         begin
            F_id_usuario        := Busca.FieldByName('id_usuario').AsInteger;
            F_id_grupo          := Busca.FieldByName('id_grupo').AsInteger;
            F_nome              := lowercase(Busca.FieldByName('nome').AsString);
            F_nomecompleto      := lowercase(Busca.FieldByName('nomecompleto').AsString);
            F_senha             := lowercase(Busca.FieldByName('senha').AsString);
            F_status            := Busca.FieldByName('status').AsInteger;
            F_ramal             := lowercase(Busca.FieldByName('ramal').AsString);
            F_email             := lowercase(Busca.FieldByName('email').AsString);
            F_cargo             := lowercase(Busca.FieldByName('cargo').AsString);
            F_datacadastro      := Busca.FieldByName('datacadastro').AsDateTime;
            F_dataultimoacesso  := Busca.FieldByName('dataultacesso').AsDateTime;
         end;
         Busca.Close;
      Finally
         FreeAndNil(Busca);
      end;
   end;

   procedure TCUsuario.Carregar_id(Id_DoUsuario: Integer);
   begin
      try
         Busca := TPSQLQuery.Create(Self);
         Busca.Database:= dm_base.my_conn;
         ws_sql:= ('select * from usuario where (id_usuario = :p_id_usuario) order by
   id_usuario');
         ws_sql := lowercase(ws_sql);
         Busca.SQL.Clear;
         Busca.SQL.Add(ws_sql);
         Busca.ParamByName('p_id_usuario').AsInteger := Id_DoUsuario;
         Busca.Open;
         if busca.IsEmpty then
         begin
            F_id_usuario        := 0;
            F_id_grupo          := 0;
            F_nome              := '';
            F_nomecompleto      := '';
            F_senha             := '';
            F_status            := 0;
            F_ramal             := '';
            F_email             := '';
            F_cargo             := '';
            F_datacadastro      := 0;
            F_dataultimoacesso  := 0;
         end
         else
         begin
            F_id_usuario        := Busca.FieldByName('id_usuario').AsInteger;
            F_id_grupo          := Busca.FieldByName('id_grupo').AsInteger;
            F_nome              := lowercase(Busca.FieldByName('nome').AsString);
            F_nomecompleto      := lowercase(Busca.FieldByName('nomecompleto').AsString);
            F_senha             := lowercase(Busca.FieldByName('senha').AsString);
            F_status            := Busca.FieldByName('status').AsInteger;
            F_ramal             := lowercase(Busca.FieldByName('ramal').AsString);
            F_email             := lowercase(Busca.FieldByName('email').AsString);
            F_cargo             := lowercase(Busca.FieldByName('cargo').AsString);
            F_datacadastro      := Busca.FieldByName('datacadastro').AsDateTime;
            F_dataultimoacesso  := Busca.FieldByName('dataultacesso').AsDateTime;
         end;
         Busca.Close;
      Finally
         FreeAndNil(Busca);
      end;
   end;

   procedure TCUsuario.Coloca;
   begin
      F_id_usuario        := OColecao.FieldByName('id_usuario').AsInteger;
```

```
   F_id_grupo         := OColecao.FieldByName('id_grupo').AsInteger;
   F_nome             := lowercase(OColecao.FieldByName('nome').AsString);
   F_nomecompleto     := lowercase(OColecao.FieldByName('nomecompleto').AsString);
   F_senha            := lowercase(OColecao.FieldByName('senha').AsString);
   F_status           := OColecao.FieldByName('status').AsInteger;
   F_ramal            := lowercase(OColecao.FieldByName('ramal').AsString);
   F_email            := lowercase(OColecao.FieldByName('email').AsString);
   F_cargo            := lowercase(OColecao.FieldByName('cargo').AsString);
   F_datacadastro     := OColecao.FieldByName('datacadastro').AsDateTime;
   F_datault imoacesso := OColecao.FieldByName('datault acesso').AsDateTime;
end;

function TCUsuario.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TCUsuario.Create(Dono: TComponent);
begin
    inherited Create(Dono);
    Grupo := TCGrupo.Create(nil);
    Ocolecao := TPSQLQuery.Create(Self);
    Ocolecao.Database := dm_base.my_conn;
end;

procedure TCUsuario.Desativar(wp_id_usuario: Integer);
begin
   try
      Acao := TPSQLQuery.Create(Self);
      Acao.Database := dm_base.my_conn;
      Acao.Close;
      ws_Sql := ('Update usuario set ');
      ws_Sql := (ws_Sql + 'Status = :p_Status ');
      ws_Sql := (ws_Sql + 'where Id_Usuario = :p_Id_Usuario');
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      Acao.ParamByName('p_id_usuario').Value       := wp_id_usuario;
      Acao.ParamByName('p_Status').Value           := 0;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

function TCUsuario.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCUsuario.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCUsuario.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCUsuario.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('insert into usuario ( Nome, NomeCompleto, Status, Id_Grupo, Ramal,
EMail, Cargo,');
         ws_Sql := (ws_Sql + ' DataCadastro,    DataUltAcesso, Senha)');
         ws_Sql := (ws_Sql + ' Values (:p_Nome, :p_NomeCompl, :p_Status, :p_Id_Grupo,
:p_Ramal, :p_EMail, :p_Cargo,');
         ws_Sql := (ws_Sql + ' :p_DataCadastro, :p_DataUltAcesso, :p_senha)');
      end
   else
```

```
    if (estado = 1) then
       begin
          ws_Sql := ('Update usuario set ');
          ws_Sql := (ws_Sql + 'Nome = :p_Nome, ');
          ws_Sql := (ws_Sql + 'NomeCompleto = :p_NomeCompl, ');
          ws_Sql := (ws_Sql + 'Status = :p_Status, ');
          ws_Sql := (ws_Sql + 'Id_Grupo = :p_Id_Grupo, ');
          ws_Sql := (ws_Sql + 'Ramal = :p_Ramal, ');
          ws_Sql := (ws_Sql + 'EMail =:p_EMail, ');
          ws_Sql := (ws_Sql + 'Cargo = :p_Cargo, ');
          ws_Sql := (ws_Sql + 'DataUltAcesso = :p_DataUltAcesso, ');
          ws_Sql := (ws_Sql + 'Senha = :p_senha ');
          ws_Sql := (ws_Sql + 'where Id_Usuario = :p_Id_Usuario');
       end;
       ws_sql := lowercase(ws_sql);
       Acao.SQL.Add(ws_sql);
       //ATRIBUI PARÂMETROS A CONSULTA
       if (estado = 1) then
       begin
          Acao.ParamByName('p_id_usuario').Value    := F_id_usuario;
       end;
       Acao.ParamByName('p_Nome').Value             := F_nome;
       Acao.ParamByName('p_NomeCompl').Value        := F_nomecompleto;
       Acao.ParamByName('p_Status').Value           := F_status;
       Acao.ParamByName('p_Id_Grupo').Value         := F_id_grupo;
       Acao.ParamByName('p_Ramal').Value            := F_ramal;
       Acao.ParamByName('p_EMail').Value            := F_email;
       Acao.ParamByName('p_Cargo').Value            := F_cargo;
       if (estado = 0) then
       begin
          Acao.ParamByName('p_DataCadastro').AsDate := F_datacadastro;
       end;
       Acao.ParamByName('p_DataUltAcesso').AsDate   := F_dataultimoacesso;
       Acao.ParamByName('p_senha').Value            := F_senha;
       Acao.ExecSQL;
    Finally
      FreeAndNil(Acao);
    End;
end;

procedure TCUsuario.SelecionaAtivos;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from usuario where (status = 1) order by id_grupo, nome';
   Ocolecao.SQL.Clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCUsuario.SelecionaItivos;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from usuario where (status = 0) order by id_grupo, nome';
   Ocolecao.SQL.Clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCUsuario.SelecionaTodos;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from usuario order by id_grupo, nome';
   Ocolecao.SQL.Clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCUsuario.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.2 unit TAcesso;

```
interface

uses
   Classes, PSQLDbTables, tmenu, sysutils;

type
  TCAcesso = class(TComponent)

    private
      F_id_usuario  : Integer;
      F_id_menu     : String;
      F_nivel       : Integer;

    public

      //Property's
      property P_id_usuario : Integer read F_id_usuario write F_id_usuario;
      property P_id_menu    : String  read F_id_menu    write F_id_menu;
      property P_nivel      : Integer read F_nivel      write F_nivel;

      //Operações
      procedure Inserir(wp_id_usuario,wp_nivel : Integer ; wp_id_menu : string);
      procedure Alterar(wp_id_usuario,wp_nivel : Integer ; wp_id_menu : string);
      function SelecionarAcesso(wp_id_usuario : Integer ; wp_id_menu : string): boolean;
      function PesquisaAcesso(wp_id_usuario : Integer ; wp_id_menu : string): boolean;

  end;

 var
 Acao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TCAcesso.Alterar(wp_id_usuario, wp_nivel: Integer ; wp_id_menu : string);
begin
      Acao := TPSQLQuery.Create(Self);
      Acao.Database:= dm_base.my_conn;
      ws_sql := ('Update acesso set Nivel = :p_Nivel where ((Id_Usuario = :p_Usu) And
(Id_Menu = :p_Menu))');
      Acao.SQL.Clear;
      Acao.SQL.Add(WS_SQL);
      Acao.ParamByName('p_Usu').Value   := wp_id_usuario;
      Acao.ParamByName('p_Menu').Value  := wp_id_menu;
      Acao.ParamByName('p_Nivel').Value := wp_nivel;
      Acao.ExecSQL;
      Acao.FreeOnRelease;
end;

procedure TCAcesso.Inserir(wp_id_usuario,wp_nivel : Integer ; wp_id_menu : string);
begin
      Acao := TPSQLQuery.Create(Self);
      Acao.Database:= dm_base.my_conn;
```

```
      ws_sql := ('insert into acesso (id_usuario, id_menu, nivel) values (:p_Usu,
:p_Menu, :p_Nivel)');
      Acao.SQL.Clear;
      Acao.SQL.Add(WS_SQL);
      Acao.ParamByName('p_Usu').Value     := wp_id_usuario;
      Acao.ParamByName('p_Menu').Value    := wp_id_menu;
      Acao.ParamByName('p_Nivel').Value   := wp_nivel;
      Acao.ExecSQL;
      Acao.FreeOnRelease;
end;

function TCAcesso.PesquisaAcesso(wp_id_usuario: Integer; wp_id_menu: string): boolean;
begin
   try
      Acao := TPSQLQuery.Create(Self);
      Acao.Database:= dm_base.my_conn;
      ws_sql := ('select * from acesso where ((Id_Usuario = :p_Usu) And (Id_Menu =
:p_Menu))');
      ws_sql := LowerCase(ws_sql);
      Acao.SQL.Clear;
      Acao.SQL.Add(ws_sql);
      Acao.ParamByName('p_Usu').Value   := wp_id_usuario;
      Acao.ParamByName('p_Menu').Value  := wp_id_menu;
      Acao.Open;
      if Acao.IsEmpty then
      begin
         Result := False;
      end
      else
      begin
         Result := True;
      end;
   Finally
      Acao.FreeOnRelease;
   End;
end;

function TCAcesso.SelecionarAcesso(wp_id_usuario : Integer ; wp_id_menu : string):
boolean;
begin
   try
      Acao := TPSQLQuery.Create(Self);
      Acao.Database:= dm_base.my_conn;
      ws_sql := ('select * from acesso where ((Id_Usuario = :p_Usu) And (Id_Menu =
:p_Menu))');
      ws_sql := LowerCase(ws_sql);
      Acao.SQL.Clear;
      Acao.SQL.Add(ws_sql);
      Acao.ParamByName('p_Usu').Value   := wp_id_usuario;
      Acao.ParamByName('p_Menu').Value  := wp_id_menu;
      Acao.Open;
      Result := False;

      if not Acao.IsEmpty then
      begin
         if acao.FieldByName('Nivel').AsInteger = 1 then
         begin
            Result := False;
         end
         else
         begin
            Result := True;
         end;
      end;

   Finally
      Acao.FreeOnRelease;
   End;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.3 unit TCaixa;

```
interface

uses
   Classes, PSQLDbTables, Controls, SysUtils ;

type
  TCCaixa = class(TComponent)
    private
      F_id_lcto         : Integer;
      F_id_pessoa       : Integer;
      F_id_tipomov      : Integer;
      F_num_doc         : Integer;
      F_num_parcela     : Integer;
      F_data_emissao    : TDate;
      F_data_vcto       : TDate;
      F_data_prorrog    : TDate;
      F_valor_apagar    : Double;
      F_situacao        : String;
      F_data_pgto       : TDate;
      F_tipo_pgto       : Integer;
      F_valor_juro      : Double;
      F_valor_pago      : Double;
      F_obs             : String;
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_lcto         : Integer read F_id_lcto      write F_id_lcto;
      property P_id_pessoa       : Integer read F_id_pessoa    write F_id_pessoa;
      property P_id_tipomov      : Integer read F_id_tipomov   write F_id_tipomov;
      property P_num_doc         : Integer read F_num_doc      write F_num_doc;
      property P_num_parcela     : Integer read F_num_parcela  write F_num_parcela;
      property P_data_emissao    : TDate   read F_data_emissao write F_data_emissao;
      property P_data_vcto       : TDate   read F_data_vcto    write F_data_vcto;
      property P_data_prorrog    : TDate   read F_data_prorrog write F_data_prorrog;
      property P_valor_apagar    : Double  read F_valor_apagar write F_valor_apagar;
      property P_situacao        : String  read F_situacao     write F_situacao;
      property P_data_pgto       : TDate   read F_data_pgto    write F_data_pgto;
      property P_tipo_pgto       : Integer read F_tipo_pgto    write F_tipo_pgto;
      property P_valor_juro      : Double  read F_valor_juro   write F_valor_juro;
      property P_valor_pago      : Double  read F_valor_pago   write F_valor_pago;
      property P_historico       : String  read F_obs          write F_obs;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      function  Calcular(wp_data_fim: tdate;tipo: integer): double;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_dataini,wp_datafim : tdate; wp_tipomovimento : integer);
      procedure Selecionar(wp_id_lcto : Integer);
      procedure SelecionarPeloDocumento(wp_num_doc : Integer);
```

```
      procedure Ultimo;
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TCCaixa.Anterior;
begin
   Ocolecao.Prior;
end;

function TCCaixa.Calcular(wp_data_fim: tdate; tipo: integer): double;
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'SELECT sum(public.caixa.valor_apagar) AS valortotal FROM public.caixa';
   ws_Sql := (ws_Sql + ' INNER JOIN public.tipomov ON (public.caixa.id_tipomov =
public.tipomov.id_tipomov) ');
   ws_Sql := (ws_Sql + ' WHERE (public.caixa.data_vcto <= :p_data) AND
(public.tipomov.tipomov = :p_tip) AND (public.caixa.situacao = :p_SIT)');
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_data').AsDateTime   := wp_data_fim;
   Acao.ParamByName('p_tip').AsInteger     := tipo;
   Acao.ParamByName('p_SIT').AsString      := 'A';
   Acao.Open;
   if  Acao.fieldbyname('valortotal').IsNull then
      result:= 0
   else
      Result:= Acao.fieldbyname('valortotal').Value;
   Acao.Free;
end;

procedure TCCaixa.Coloca;
begin
   F_id_lcto         := OColecao.FieldByName('id_lcto').AsInteger;
   F_id_pessoa       := OColecao.FieldByName('id_pessoa').AsInteger;
   F_id_tipomov      := OColecao.FieldByName('id_tipomov').AsInteger;
   F_num_doc         := OColecao.FieldByName('num_doc').AsInteger;
   F_num_parcela     := OColecao.FieldByName('num_parcela').AsInteger;
   F_data_emissao    := OColecao.FieldByName('data_emissao').AsDateTime;
   F_data_vcto       := OColecao.FieldByName('data_vcto').AsDateTime;
   F_data_prorrog    := OColecao.FieldByName('data_prorrog').AsDateTime;
   F_valor_apagar    := OColecao.FieldByName('valor_apagar').AsFloat;
   F_situacao        := OColecao.FieldByName('situacao').AsString;
   F_data_pgto       := OColecao.FieldByName('data_pgto').AsDateTime;
   F_tipo_pgto       := OColecao.FieldByName('tipo_pgto').AsInteger;
   F_valor_juro      := OColecao.FieldByName('valor_juro').AsFloat;
   F_valor_pago      := OColecao.FieldByName('valor_pago').AsFloat;
   F_obs             := OColecao.FieldByName('obs').AsString;
end;

function TCCaixa.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TCCaixa.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCCaixa.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCCaixa.Primeiro;
begin
   Ocolecao.First;
```

```
end;

procedure TCCaixa.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCCaixa.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into caixa ');
         ws_Sql := (ws_Sql + ' (num_doc, id_tipomov, id_pessoa, num_parcela,
data_emissao, data_vcto, data_pgto, ');
         if F_data_prorrog <> 0 then
            ws_Sql := (ws_Sql + ' data_prorrog, ');
         ws_Sql := (ws_Sql + ' tipo_pgto, valor_apagar, valor_juro, valor_pago,
situacao, obs)');
         ws_Sql := (ws_Sql + ' Values ');
         ws_Sql := (ws_Sql + ' (:P_num_doc, :P_id_tipomov, :P_id_pessoa, :P_num_parcela,
:P_data_emissao, :P_data_vcto, :P_data_pgto, ');
         if F_data_prorrog <> 0 then
            ws_Sql := (ws_Sql + ' :P_data_prorrog, ');
         ws_Sql := (ws_Sql + ' :P_tipo_pgto, :P_valor_apagar, :P_valor_juro,
:P_valor_pago, :P_situacao, :P_obs)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update caixa set ');
         ws_Sql := (ws_Sql + ' num_doc      = :P_num_doc,       ');
         ws_Sql := (ws_Sql + ' id_tipomov   = :P_id_tipomov,    ');
         ws_Sql := (ws_Sql + ' id_pessoa    = :P_id_pessoa,     ');
         ws_Sql := (ws_Sql + ' num_parcela  = :P_num_parcela,   ');
         ws_Sql := (ws_Sql + ' data_emissao = :P_data_emissao,  ');
         ws_Sql := (ws_Sql + ' data_vcto    = :P_data_vcto,     ');
         ws_Sql := (ws_Sql + ' data_pgto    = :P_data_pgto,     ');
         ws_Sql := (ws_Sql + ' data_prorrog = :P_data_prorrog,  ');
         ws_Sql := (ws_Sql + ' tipo_pgto    = :P_tipo_pgto,     ');
         ws_Sql := (ws_Sql + ' valor_apagar = :P_valor_apagar,  ');
         ws_Sql := (ws_Sql + ' valor_juro   = :P_valor_juro,    ');
         ws_Sql := (ws_Sql + ' valor_pago   = :P_valor_pago,    ');
         ws_Sql := (ws_Sql + ' situacao     = :P_situacao,      ');
         ws_Sql := (ws_Sql + ' obs          = :P_obs            ');
         ws_Sql := (ws_Sql + ' where ( id_lcto   = :p_id_lcto ) ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
         Acao.ParamByName('p_id_lcto').asinteger    := F_id_lcto;
      Acao.ParamByName('P_num_doc').AsInteger       := F_num_doc;
      Acao.ParamByName('P_id_tipomov').AsInteger    := F_id_tipomov;
      Acao.ParamByName('P_id_pessoa').AsInteger     := F_id_pessoa;
      Acao.ParamByName('P_num_parcela').AsInteger   := F_num_parcela;
      Acao.ParamByName('P_data_emissao').AsDateTime := F_data_emissao;
      Acao.ParamByName('P_data_vcto').AsDateTime    := F_data_vcto;
      if F_data_pgto <> 0 then
         Acao.ParamByName('P_data_pgto').AsDateTime    := F_data_pgto;
      if F_data_prorrog <> 0 then
         Acao.ParamByName('P_data_prorrog').AsDateTime := F_data_prorrog;
      Acao.ParamByName('P_tipo_pgto').AsInteger     := F_tipo_pgto;
      Acao.ParamByName('P_valor_apagar').AsFloat    := F_valor_apagar;
      Acao.ParamByName('P_valor_juro').AsFloat      := F_valor_juro;
      Acao.ParamByName('P_valor_pago').AsFloat      := F_valor_pago;
      Acao.ParamByName('P_situacao').AsString       := F_situacao;
      Acao.ParamByName('P_obs').AsString            := F_obs;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
```

```
end;

procedure TCCaixa.Seleciona(wp_dataini,wp_datafim : tdate; wp_tipomovimento : integer);
begin
// tipo = 0 >> Contas a receber
// tipo = 1 >> Contas a pagar
// tipo = 2 >> Todas as Contas

   Ocolecao.Close;
   WS_SQL := 'select caixa.* from tipomov INNER JOIN caixa ON (tipomov.id_tipomov =
caixa.id_tipomov) ';
   WS_SQL := (WS_SQL + ' where (data_vcto >= :p_diaini) and (data_vcto <= :p_diafim) ');
   if wp_tipomovimento <> 2 then
      WS_SQL := (WS_SQL + ' and (tipomov.tipomov = :p_tipomov) and (situacao =
:P_situacao) ')
   else
      WS_SQL := (WS_SQL + ' and (situacao <> :P_situacao)');
   WS_SQL := (WS_SQL + ' order by data_vcto, id_lcto ,id_tipomov');

   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_diaini').AsDateTime     := wp_dataini;
   Ocolecao.ParamByName('p_diafim').AsDateTime     := wp_datafim;
   if wp_tipomovimento <> 2 then
   begin
      Ocolecao.ParamByName('p_tipomov').Value      := wp_tipomovimento;
      Ocolecao.ParamByName('P_situacao').AsString  := 'A';
   end
   else
      Ocolecao.ParamByName('P_situacao').AsString  := 'C';

   Ocolecao.Open;
end;

procedure TCCaixa.Selecionar(wp_id_lcto : Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from caixa where (id_lcto = :p_id_lcto) ';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_lcto').Value   := wp_id_lcto;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_lcto         := 0;
      F_id_pessoa       := 0;
      F_id_tipomov      := 0;
      F_num_doc         := 0;
      F_num_parcela     := 0;
      F_data_emissao    := 0;
      F_data_vcto       := 0;
      F_data_prorrog    := 0;
      F_valor_apagar    := 0;
      F_situacao        := '';
      F_data_pgto       := 0;
      F_tipo_pgto       := 0;
      F_valor_juro      := 0;
      F_valor_pago      := 0;
      F_obs             := '';
   end
   else
   begin
      F_id_lcto         := Acao.FieldByName('id_lcto').AsInteger;
      F_id_pessoa       := Acao.FieldByName('id_pessoa').AsInteger;
      F_id_tipomov      := Acao.FieldByName('id_tipomov').AsInteger;
      F_num_doc         := Acao.FieldByName('num_doc').AsInteger;
      F_num_parcela     := Acao.FieldByName('num_parcela').AsInteger;
      F_data_emissao    := Acao.FieldByName('data_emissao').AsDateTime;
      F_data_vcto       := Acao.FieldByName('data_vcto').AsDateTime;
      F_data_prorrog    := Acao.FieldByName('data_prorrog').AsDateTime;
      F_valor_apagar    := Acao.FieldByName('valor_apagar').AsFloat;
      F_situacao        := Acao.FieldByName('situacao').AsString;
      F_data_pgto       := Acao.FieldByName('data_pgto').AsDateTime;
      F_tipo_pgto       := Acao.FieldByName('tipo_pgto').AsInteger;
      F_valor_juro      := Acao.FieldByName('valor_juro').AsFloat;
```

```
      F_valor_pago      := Acao.FieldByName('valor_pago').AsFloat;
      F_obs             := Acao.FieldByName('obs').AsString;
   end;
   Acao.Free;
end;

procedure TCCaixa.SelecionarPeloDocumento(wp_num_doc: Integer);
begin
   Ocolecao.Close;
   WS_SQL := ' UPDATE caixa SET situacao = :p_0';
   WS_SQL := (WS_SQL + ' where (num_doc = :P_1) AND (obs = :P_2)');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('P_0').AsString := 'C';
   Ocolecao.ParamByName('P_1').AsInteger:= wp_num_doc;
   Ocolecao.ParamByName('P_2').AsString := 'REFERENTE OS N° '+inttostr(wp_num_doc);
   Ocolecao.Open;
end;

procedure TCCaixa.Ultimo;
begin
   Ocolecao.Last;
end;


end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.4 unit TCidade;

```
interface

uses
   Classes, PSQLDbTables, TUf, Controls, SysUtils ;

type
  TcCidade = class(TComponent)
    private
      F_id_cidade : Integer;
      F_id_uf     : Integer;
      F_descricao : String;
    public
      UF : TCUf;

      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_cidade : Integer read F_id_cidade write F_id_cidade;
      property P_id_uf     : Integer read F_id_uf     write F_id_uf;
      property P_descricao : String  read F_descricao write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure SelecionacidUf(wp_id_Uf: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_Cidade,wp_id_Uf: Integer);
      procedure Ultimo;
  end;

 var
 Busca,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TcCidade.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcCidade.Coloca;
begin
   F_id_cidade := OColecao.FieldByName('id_cidade').AsInteger;
   F_id_uf     := OColecao.FieldByName('id_uf').AsInteger;
   F_descricao := OColecao.FieldByName('descricao').AsString;
end;

function TcCidade.Comeco: Boolean;
```

```
begin
   Result := OColecao.Bof;
end;

constructor TcCidade.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  UF := TCUf.Create(nil);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcCidade.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcCidade.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcCidade.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcCidade.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;

   if (estado = 0) then
      begin
         ws_Sql := ('Insert into cidade (id_uf, descricao)');
         ws_Sql := (ws_Sql +' Values (:P_id_uf, :p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update cidade set ');
         ws_Sql := (ws_Sql + ' descricao         = :p_descricao, ');
         ws_Sql := (ws_Sql + ' id_uf             = :P_id_uf    ');
         ws_Sql := (ws_Sql + ' where (id_cidade  = :p_id_cidade)');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_cidade').AsInteger := F_id_cidade;
      end;
      Acao.ParamByName('P_id_uf').AsInteger     := F_id_uf;
      Acao.ParamByName('p_descricao').AsString  := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcCidade.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from cidade';
   WS_SQL := (WS_SQL + ' order by id_cidade,id_uf, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcCidade.SelecionacidUf(wp_id_Uf: Integer);
begin
   Ocolecao.Close;
```

```
   WS_SQL := 'select * from cidade where (id_uf = :p_id_uf)';
   WS_SQL := (WS_SQL + ' order by id_cidade,id_uf, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_uf').Value     := wp_id_Uf;
   Ocolecao.Open;
end;

procedure TcCidade.Selecionar(wp_id_Cidade,wp_id_Uf: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from cidade where ((id_cidade = :p_id_cidade) and (id_uf =
:p_id_uf))';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_cidade').Value := wp_id_Cidade;
   Acao.ParamByName('p_id_uf').Value     := wp_id_Uf;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_cidade := 0;
      F_id_uf     := 0;
      F_descricao := '';
   end
   else
   begin
      F_id_cidade := Acao.FieldByName('id_cidade').AsInteger;
      F_id_uf     := Acao.FieldByName('id_uf').AsInteger;
      F_descricao := Acao.FieldByName('descricao').AsString;
   end;
   Acao.Free;
end;

procedure TcCidade.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.5 unit TContatos;

```
interface

uses
   Classes, PSQLDbTables, SysUtils,TFones;

type
  TcContatos = class(TComponent)
    private
      F_id_pessoa : Integer;
      F_nome      : String;
      F_dia_nasc  : String;
      F_mes_nasc  : String;
      F_cargo     : String;
      F_setor     : String;
      F_email     : String;
    public
      Telefones : TcFones;
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_pessoa : Integer read F_id_pessoa write F_id_pessoa;
      property P_nome      : String  read F_nome      write F_nome;
      property P_dia_nasc  : String  read F_dia_nasc  write F_dia_nasc;
      property P_mes_nasc  : String  read F_mes_nasc  write F_mes_nasc;
      property P_cargo     : String  read F_cargo     write F_cargo;
      property P_setor     : String  read F_setor     write F_setor;
      property P_email     : String  read F_email     write F_email;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_id_pessoa: Integer);
      procedure SelecionarUm(wp_id_pessoa: Integer; wp_nome : String);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TTipoDePeca }

procedure TcContatos.Anterior;
begin
   Ocolecao.Prior;
end;
```

```
procedure TcContatos.Coloca;
begin
      F_id_pessoa := OColecao.FieldByName('id_pessoa').AsInteger;
      F_nome      := OColecao.FieldByName('nome').AsString;
      F_dia_nasc  := OColecao.FieldByName('dia_nasc').AsString;
      F_mes_nasc  := OColecao.FieldByName('mes_nasc').AsString;
      F_cargo     := OColecao.FieldByName('cargo').AsString;
      F_setor     := OColecao.FieldByName('setor').AsString;
      F_email     := OColecao.FieldByName('email').AsString;
end;

function TcContatos.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcContatos.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Telefones := TcFones.create(nil);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcContatos.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcContatos.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcContatos.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcContatos.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into contatos (id_pessoa, nome, dia_nasc, mes_nasc, cargo,
setor, email)');
         ws_Sql := (ws_Sql +' Values (:p_id_pessoa, :p_nome, :p_dia_nasc, :p_mes_nasc,
:p_cargo, :p_setor, :p_email)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update contatos set ');
         ws_Sql := (ws_Sql + ' dia_nasc = :p_dia_nasc, ');
         ws_Sql := (ws_Sql + ' mes_nasc = :p_mes_nasc, ');
         ws_Sql := (ws_Sql + ' cargo    = :p_cargo, ');
         ws_Sql := (ws_Sql + ' setor    = :p_setor, ');
         ws_Sql := (ws_Sql + ' email    = :p_email ');
         ws_Sql := (ws_Sql + ' where (id_pessoa = :p_id_pessoa) and (nome = :p_nome)
');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      Acao.ParamByName('p_id_pessoa').AsInteger := F_id_pessoa;
      Acao.ParamByName('p_nome').AsString       := F_nome;
      Acao.ParamByName('p_dia_nasc').AsString   := F_dia_nasc;
      Acao.ParamByName('p_mes_nasc').AsString   := F_mes_nasc;
      Acao.ParamByName('p_cargo').AsString      := F_cargo;
      Acao.ParamByName('p_setor').AsString      := F_setor;
      Acao.ParamByName('p_email').AsString      := F_email;
```

```
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcContatos.Seleciona(wp_id_pessoa: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from contatos where (id_pessoa = :p_id_pessoa)';
   WS_SQL := (WS_SQL + ' order by id_pessoa, nome');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_pessoa').AsInteger := wp_id_pessoa;
   Ocolecao.Open;
end;

procedure TcContatos.SelecionarUm(wp_id_pessoa: Integer; wp_nome : String);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from contatos where (id_pessoa = :p_id_pessoa) and (nome =
:p_nome)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_pessoa').AsInteger := wp_id_pessoa;
   Acao.ParamByName('p_nome').AsString       := wp_nome;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_pessoa := 0;
      F_nome      := '';
      F_dia_nasc  := '';
      F_mes_nasc  := '';
      F_cargo     := '';
      F_setor     := '';
      F_email     := '';
   end
   else
   begin
      F_id_pessoa := Acao.FieldByName('id_pessoa').AsInteger;
      F_nome      := Acao.FieldByName('nome').AsString;
      F_dia_nasc  := Acao.FieldByName('dia_nasc').AsString;
      F_mes_nasc  := Acao.FieldByName('mes_nasc').AsString;
      F_cargo     := Acao.FieldByName('cargo').AsString;
      F_setor     := Acao.FieldByName('setor').AsString;
      F_email     := Acao.FieldByName('email').AsString;
   end;
   Acao.Free;
end;

procedure TcContatos.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.6 unit TContrato;

```
interface

uses
   Classes, PSQLDbTables, Controls, SysUtils ;

type
  TCContrato = class(TComponent)
    private
      F_id_contrato           : Integer;
      F_id_pessoa             : Integer;
      F_data_inicio_contrato  : TDate;
      F_data_fim_vigencia     : TDate;
      F_data_inicio_vigencia  : TDate;
      F_data_bloqueio         : TDate;
      F_valor_mensal          : Double;
      F_dia_vcto              : Integer;
      F_dias_prorog           : Integer;
      F_bloqueado             : Integer;
    public

      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_contrato           : Integer read F_id_contrato          write
F_id_contrato;
      property P_id_pessoa             : Integer read F_id_pessoa            write
F_id_pessoa;
      property P_data_inicio_contrato  : TDate   read F_data_inicio_contrato write
F_data_inicio_contrato;
      property P_data_fim_vigencia     : TDate   read F_data_fim_vigencia    write
F_data_fim_vigencia;
      property P_data_inicio_vigencia  : TDate   read F_data_inicio_vigencia write
F_data_inicio_vigencia;
      property P_data_bloqueio         : TDate   read F_data_bloqueio        write
F_data_bloqueio;
      property P_valor_mensal          : Double  read F_valor_mensal         write
F_valor_mensal;
      property P_dia_vcto              : Integer read F_dia_vcto             write
F_dia_vcto;
      property P_dias_prorog           : Integer read F_dias_prorog          write
F_dias_prorog;
      property P_bloqueado             : Integer read F_bloqueado            write
F_bloqueado;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_Pessoa,wp_id_Contrato: Integer);
      procedure Pesquisar(wp_id_Pessoa : Integer);
```

```
      function  TemRegistros(wp_id_Pessoa : Integer) : Boolean;
      procedure Ultimo;
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TCContrato.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCContrato.Coloca;
begin
   F_id_contrato          := OColecao.FieldByName('id_contrato').AsInteger;
   F_id_pessoa            := OColecao.FieldByName('id_pessoa').AsInteger;
   F_data_inicio_contrato := OColecao.FieldByName('data_inicio_contrato').AsDateTime;
   F_data_fim_vigencia    := OColecao.FieldByName('data_fim_vigencia').AsDateTime;
   F_data_inicio_vigencia := OColecao.FieldByName('data_inicio_vigencia').AsDateTime;
   F_bloqueado            := OColecao.FieldByName('bloqueado').AsInteger;
   if F_bloqueado = 1 then
      F_data_bloqueio     := OColecao.FieldByName('data_bloqueio').AsDateTime
   else
      F_data_bloqueio     := 0;
   F_valor_mensal         := OColecao.FieldByName('valor_mensal').AsFloat;
   F_dia_vcto             := OColecao.FieldByName('dia_vcto').AsInteger;
   F_dias_prorog          := OColecao.FieldByName('dias_prorog').AsInteger;
end;

function TCContrato.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TCContrato.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCContrato.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCContrato.Pesquisar(wp_id_Pessoa: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from contrato where ((id_pessoa = :p_id_pessoa) and (bloqueado <>
1) and (data_fim_vigencia > :p_data))';
   WS_SQL := (WS_SQL + ' order by id_contrato desc');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_pessoa').AsInteger := wp_id_Pessoa;
   Ocolecao.ParamByName('p_data').AsDate         := date;
   Ocolecao.Open;
end;

procedure TCContrato.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCContrato.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCContrato.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
```

```
    try
    Acao := TPSQLQuery.Create(Self);
    Acao.Database := dm_base.my_conn;
    Acao.Close;
    if (estado = 0) then
       begin
          ws_Sql := ('Insert into contrato ');
          ws_Sql := (ws_Sql + ' (id_pessoa, data_inicio_contrato, data_fim_vigencia,
data_inicio_vigencia, ');
          if F_bloqueado = 1 then
          begin
             ws_Sql := (ws_Sql + ' data_bloqueio, ');
          end;
          ws_Sql := (ws_Sql + '   valor_mensal, dia_vcto, dias_prorog, bloqueado)');
          ws_Sql := (ws_Sql + ' Values ');
          ws_Sql := (ws_Sql + ' (:P_id_pessoa, :P_data_inicio_contrato,
:P_data_fim_vigencia, :P_data_inicio_vigencia, ');
          if F_bloqueado = 1 then
          begin
             ws_Sql := (ws_Sql + ' :P_data_bloqueio, ');
          end;
          ws_Sql := (ws_Sql + ' :P_valor_mensal, :P_dia_vcto, :P_dias_prorog,
:P_bloqueado)');
       end
    else
    if (estado = 1) then
       begin
          ws_Sql := ('Update contrato set ');
          ws_Sql := (ws_Sql + ' id_pessoa            = :P_id_pessoa,             ');
          ws_Sql := (ws_Sql + ' data_inicio_contrato = :P_data_inicio_contrato,  ');
          ws_Sql := (ws_Sql + ' data_fim_vigencia    = :P_data_fim_vigencia,     ');
          ws_Sql := (ws_Sql + ' data_inicio_vigencia = :P_data_inicio_vigencia,  ');
          if F_bloqueado = 1 then
          begin
             ws_Sql := (ws_Sql + ' data_bloqueio        = :P_data_bloqueio,         ');
          end;
          ws_Sql := (ws_Sql + ' valor_mensal         = :P_valor_mensal,          ');
          ws_Sql := (ws_Sql + ' dia_vcto             = :P_dia_vcto,              ');
          ws_Sql := (ws_Sql + ' dias_prorog          = :P_dias_prorog,           ');
          ws_Sql := (ws_Sql + ' bloqueado            = :P_bloqueado              ');
          ws_Sql := (ws_Sql + ' where (id_contrato  = :p_id_contrato)             ');
       end;
       ws_sql := lowercase(ws_sql);
       Acao.SQL.Add(ws_sql);
       //ATRIBUI PARÂMETROS A CONSULTA
       if (estado = 1) then
       begin
          Acao.ParamByName('p_id_contrato').AsInteger        := F_id_contrato;
       end;
       Acao.ParamByName('P_id_pessoa').AsInteger             := F_id_pessoa;
       Acao.ParamByName('P_data_inicio_contrato').AsDateTime := F_data_inicio_contrato;
       Acao.ParamByName('P_data_fim_vigencia').AsDateTime    := F_data_fim_vigencia;
       Acao.ParamByName('P_data_inicio_vigencia').AsDateTime := F_data_inicio_vigencia;
       if F_bloqueado = 1 then
       begin
          Acao.ParamByName('P_data_bloqueio').AsDateTime     := F_data_bloqueio;
       end;
       Acao.ParamByName('P_valor_mensal').AsFloat            := F_valor_mensal;
       Acao.ParamByName('P_dia_vcto').AsInteger              := F_dia_vcto;
       Acao.ParamByName('P_dias_prorog').AsInteger           := F_dias_prorog;
       Acao.ParamByName('P_bloqueado').AsInteger             := F_bloqueado;
       Acao.ExecSQL;
    Finally
      FreeAndNil(Acao);
    End;
end;

procedure TCContrato.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from contrato';
   WS_SQL := (WS_SQL + ' order by id_pessoa,id_contrato');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;
```

```
procedure TCContrato.Selecionar(wp_id_Pessoa,wp_id_Contrato: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from contrato where ((id_pessoa = :p_id_pessoa) and (id_contrato
= :p_id_contrato))';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_pessoa').Value   := wp_id_Pessoa;
   Acao.ParamByName('p_id_contrato').Value := wp_id_Contrato;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_contrato          := 0;
      F_id_pessoa            := 0;
      F_data_inicio_contrato := date;
      F_data_fim_vigencia    := 0;
      F_data_inicio_vigencia := 0;
      F_data_bloqueio        := 0;
      F_valor_mensal         := 0;
      F_dia_vcto             := 0;
      F_dias_prorog          := 0;
      F_bloqueado            := 0;
   end
   else
   begin
      F_id_contrato          := Acao.FieldByName('id_contrato').AsInteger;
      F_id_pessoa            := Acao.FieldByName('id_pessoa').AsInteger;
      F_data_inicio_contrato := Acao.FieldByName('data_inicio_contrato').AsDateTime;
      F_data_fim_vigencia    := Acao.FieldByName('data_fim_vigencia').AsDateTime;
      F_data_inicio_vigencia := Acao.FieldByName('data_inicio_vigencia').AsDateTime;
      F_bloqueado            := Acao.FieldByName('bloqueado').AsInteger;
      if F_bloqueado = 1 then
         F_data_bloqueio     := Acao.FieldByName('data_bloqueio').AsDateTime
      else
         F_data_bloqueio     := 0;
      F_valor_mensal         := Acao.FieldByName('valor_mensal').AsFloat;
      F_dia_vcto             := Acao.FieldByName('dia_vcto').AsInteger;
      F_dias_prorog          := Acao.FieldByName('dias_prorog').AsInteger;
   end;
   Acao.Free;
end;

function TCContrato.TemRegistros(wp_id_Pessoa : Integer) : Boolean;
begin
   Ocolecao.Close;
   WS_SQL := 'select id_contrato from contrato where ((id_pessoa = :p_id_pessoa) and
(bloqueado <> 1) and (data_fim_vigencia > :p_data))';
   WS_SQL := (WS_SQL + ' order by id_contrato');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_pessoa').AsInteger := wp_id_Pessoa;
   Ocolecao.ParamByName('p_data').AsDate         := date;
   Ocolecao.Open;
   Result:= not Ocolecao.IsEmpty;
end;

procedure TCContrato.Ultimo;
begin
   Ocolecao.Last;
end;


end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.7 unit TEquipamento;

```
interface

uses
   Classes, PSQLDbTables, SysUtils, TTipoDeEquipamento;

type
  TcEquipamento = class(TComponent)
    private
      F_id_equipamento        : Integer;
      F_id_tipo_equipamento   : Integer;
      F_id_contrato           : Integer;
      F_descricao             : wideString;
      F_numero_serie          : String;
    public
      Tipo_de_Equipamento     : TcTipoDeEquipamento;

      //Property's
      property P_id_equipamento      : Integer    read F_id_equipamento      write
F_id_equipamento;
      property P_id_tipo_equipamento : Integer    read F_id_tipo_equipamento write
F_id_tipo_equipamento;
      property P_id_contrato         : Integer    read F_id_contrato         write
F_id_contrato;
      property P_descricao           : wideString read F_descricao           write
F_descricao;
      property P_numero_serie        : String     read F_numero_serie        write
F_numero_serie;

      constructor Create(Dono: TComponent);

      //Operações
      procedure Salvar(Estado: Integer);
      procedure Selecionar(wp_id_equipamento: Integer);
      // operacoes na coleção
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Seleciona(wp_id_contrato : Integer);
      procedure Pesquisar(wp_id_contrato : Integer);
      procedure Ultimo;
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;


{ TcPeca }
```

```
constructor TcEquipamento.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Tipo_de_Equipamento   := TcTipoDeEquipamento.Create(nil);
  Ocolecao              := TPSQLQuery.Create(Self);
  Ocolecao.Database     := dm_base.my_conn;
end;

procedure TcEquipamento.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar

   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         //INSERINDO DADOS NA TABELA
         ws_Sql := ('Insert into equipamento (id_tipo_equipamento, id_contrato,
descricao, numero_serie)');
         ws_Sql := (ws_Sql +' Values (:p_id_tipo_equipamento, :p_id_contrato,
:p_descricao,');
         ws_Sql := (ws_Sql +' :p_numero_serie)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update equipamento set ');
         ws_Sql := (ws_Sql + ' id_tipo_equipamento   = :p_id_tipo_equipamento , ');
         ws_Sql := (ws_Sql + ' id_contrato           = :p_id_contrato ,         ');
         ws_Sql := (ws_Sql + ' descricao             = :p_descricao,            ');
         ws_Sql := (ws_Sql + ' numero_serie          = :p_numero_serie          ');
         ws_Sql := (ws_Sql + ' where (id_equipamento = :p_id_equipamento)       ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_equipamento').AsInteger    := F_id_equipamento;
      end;
      Acao.ParamByName('p_id_tipo_equipamento').AsInteger  := F_id_tipo_equipamento;
      Acao.ParamByName('p_id_contrato').AsInteger          := F_id_contrato;
      Acao.ParamByName('p_descricao').AsString             := F_descricao;
      Acao.ParamByName('p_numero_serie').AsString          := F_numero_serie;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcEquipamento.Seleciona(wp_id_contrato : Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from equipamento where (id_contrato = :p_id_contrato)';
   WS_SQL := (WS_SQL + ' order by id_contrato,id_equipamento');
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_contrato').AsInteger := wp_id_contrato;
   Ocolecao.Open;
end;

procedure TcEquipamento.Ultimo;
begin
   Ocolecao.Last;
end;

procedure TcEquipamento.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcEquipamento.Coloca;
begin
   F_id_equipamento      := OColecao.FieldByName('id_equipamento').AsInteger;
   F_id_tipo_equipamento := OColecao.FieldByName('id_tipo_equipamento').AsInteger;
```

```
   F_id_contrato         := OColecao.FieldByName('id_contrato').AsInteger;
   F_descricao           := OColecao.FieldByName('descricao').AsString;
   F_numero_serie        := OColecao.FieldByName('numero_serie').AsString;
end;

function TcEquipamento.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

procedure TcEquipamento.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcEquipamento.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcEquipamento.Selecionar(wp_id_equipamento: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from equipamento where (id_equipamento = :p_id_equipamento)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_equipamento').Value  := wp_id_equipamento;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_equipamento      := 0;
      F_id_tipo_equipamento := 0;
      F_id_contrato         := 0;
      F_descricao           := '';
      F_numero_serie        := '';
   end
   else
   begin
      F_id_equipamento      := Acao.FieldByName('id_equipamento').AsInteger;
      F_id_tipo_equipamento := Acao.FieldByName('id_tipo_equipamento').AsInteger;
      F_id_contrato         := Acao.FieldByName('id_contrato').AsInteger;
      F_descricao           := Acao.FieldByName('descricao').AsString;
      F_numero_serie        := Acao.FieldByName('numero_serie').AsString;
   end;
   Acao.Free;
end;

function TcEquipamento.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcEquipamento.Pesquisar(wp_id_contrato: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from equipamento where (id_contrato = :p_id_contrato)';
   WS_SQL := (WS_SQL + ' order by id_contrato');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_contrato').AsInteger := wp_id_contrato;
   Ocolecao.Open;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.8 unit TFones;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type

  TcFones = class(TComponent)
    private
      F_id_fone   : Integer;
      F_id_pessoa : Integer;
      F_nome      : String;
      F_telefone  : String;
      F_tipo_fone : String;
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_fone   : Integer read F_id_fone   write F_id_fone;
      property P_id_pessoa : Integer read F_id_pessoa write F_id_pessoa;
      property P_nome      : String  read F_nome      write F_nome;
      property P_telefone  : String  read F_telefone  write F_telefone;
      property P_tipo_fone : String  read F_tipo_fone write F_tipo_fone;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure SelecionaTodos(wp_id_pessoa: Integer; wp_nome: String);
      procedure SelecionarUm(wp_id_pessoa,wp_id_fone: Integer; wp_nome: String);
      procedure SelecionarPdes(wp_id_pessoa: Integer; wp_telefone: String);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TFones }

procedure TcFones.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcFones.Coloca;
begin
```

```
   F_id_fone   := OColecao.FieldByName('id_fone').AsInteger;
   F_id_pessoa := OColecao.FieldByName('id_pessoa').AsInteger;
   F_nome      := OColecao.FieldByName('nome').AsString;
   F_telefone  := OColecao.FieldByName('telefone').AsString;
   F_tipo_fone := OColecao.FieldByName('tipo_fone').AsString;
end;

function TcFones.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcFones.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcFones.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcFones.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcFones.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcFones.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
//Estado = 2 >> Excluir
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into telefones (id_pessoa, nome, telefone, tipo_fone)');
         ws_Sql := (ws_Sql +' Values (:p_id_pessoa, :p_nome, :p_telefone,
:p_tipo_fone)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update telefones set ');
         ws_Sql := (ws_Sql + ' telefone  = :p_telefone, ');
         ws_Sql := (ws_Sql + ' tipo_fone = :p_tipo_fone ');
         ws_Sql := (ws_Sql + ' where ((id_fone = :p_id_fone) and (id_pessoa =
:p_id_pessoa) and (nome = :p_nome)) ');
      end
      else
      if (estado = 2) then
         begin
         ws_Sql := ('delete from telefones where (id_fone = :p_id_fone) and (id_pessoa =
:p_id_pessoa) and (Nome = :p_nome)');
         end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) or (estado = 2) then
      begin
         Acao.ParamByName('p_id_fone').AsInteger   := F_id_fone;

      end;
      Acao.ParamByName('p_id_pessoa').AsInteger := F_id_pessoa;
      Acao.ParamByName('p_nome').AsString       := F_nome;

      if (estado <> 2) then
      begin
```

```
         Acao.ParamByName('p_telefone').AsString  := F_telefone;
         Acao.ParamByName('p_tipo_fone').AsString := F_tipo_fone;
      end;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcFones.SelecionaTodos(wp_id_pessoa: Integer ; wp_nome: String);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from telefones where (id_pessoa = :p_id_pessoa) and (nome =
:p_nome)';
   WS_SQL := (WS_SQL + ' order by id_pessoa, nome');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_pessoa').AsInteger := wp_id_pessoa;
   Ocolecao.ParamByName('p_nome').AsString       := wp_nome;
   Ocolecao.Open;
end;

procedure TcFones.SelecionarUm(wp_id_pessoa,wp_id_fone: Integer ; wp_nome: String);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from telefones where (id_fone = :p_id_fone) and (id_pessoa =
:p_id_pessoa) and (nome = :p_nome)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_fone').Value   := wp_id_fone;
   Acao.ParamByName('p_id_pessoa').Value := wp_id_pessoa;
   Acao.ParamByName('p_nome').Value      := wp_nome;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_fone   := 0;
      F_id_pessoa := 0;
      F_nome      := '';
      F_telefone  := '';
      F_tipo_fone := '';
   end
   else
   begin
      F_id_fone   := Acao.FieldByName('id_fone').AsInteger;
      F_id_pessoa := Acao.FieldByName('id_pessoa').AsInteger;
      F_nome      := Acao.FieldByName('nome').AsString;
      F_telefone  := Acao.FieldByName('telefone').AsString;
      F_tipo_fone := Acao.FieldByName('tipo_fone').AsString;
   end;
   Acao.Free;
end;

procedure TcFones.Ultimo;
begin
   Ocolecao.Last;
end;

procedure TcFones.SelecionarPdes(wp_id_pessoa: Integer; wp_telefone: String);
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from telefones where (id_pessoa = :p_id_pessoa) and (telefone =
:p_telefone)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_pessoa').Value := wp_id_pessoa;
   Acao.ParamByName('p_telefone').Value  := wp_telefone;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_fone   := 0;
      F_id_pessoa := 0;
      F_nome      := '';
      F_telefone  := '';
      F_tipo_fone := '';
   end
```

```
    else
    begin
       F_id_fone   := Acao.FieldByName('id_fone').AsInteger;
       F_id_pessoa := Acao.FieldByName('id_pessoa').AsInteger;
       F_nome      := Acao.FieldByName('nome').AsString;
       F_telefone  := Acao.FieldByName('telefone').AsString;
       F_tipo_fone := Acao.FieldByName('tipo_fone').AsString;
    end;
    Acao.Free;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.9 unit TGrupo;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TCGrupo = class(TComponent)
    private
      F_id_grupo  : Integer;
      F_descricao : String;
    public
      constructor Create(Dono: TComponent);

      //Property's
      property P_id_grupo  : Integer read F_id_grupo  write F_id_grupo;
      property P_descricao : String  read F_descricao write F_descricao;

      procedure Salvar(Estado : Integer);
      procedure Selecionar(wp_id_grupo : Integer);
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Seleciona;
      procedure Ultimo;
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;
{ TCGrupo }

procedure TCGrupo.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCGrupo.Coloca;
begin
   F_id_grupo  := OColecao.FieldByName('id_grupo').AsInteger;
   F_descricao := OColecao.FieldByName('nomegrupo').AsString;
end;

function TCGrupo.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TCGrupo.Create(Dono: TComponent);
begin
```

```
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCGrupo.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCGrupo.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCGrupo.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCGrupo.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into grupo (nomegrupo)');
         ws_Sql := (ws_Sql +' Values (:p_nomegrupo)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update grupo set ');
         ws_Sql := (ws_Sql + ' nomegrupo       = :p_nomegrupo ');
         ws_Sql := (ws_Sql + ' where (Id_Grupo = :p_Id_grupo)  ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_Id_grupo').AsInteger := F_id_grupo;
      end;
      Acao.ParamByName('p_nomegrupo').AsString := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCGrupo.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from grupo';
   WS_SQL := (WS_SQL + ' order by id_grupo, nomegrupo');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCGrupo.Selecionar(wp_id_grupo: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from grupo where (id_grupo = :p_grupo)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_Grupo').Value  := wp_id_grupo;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_grupo  := 0;
```

```
      F_descricao := '';
   end
   else
   begin
      F_id_grupo  := Acao.FieldByName('id_grupo').AsInteger;
      F_descricao := Acao.FieldByName('nomegrupo').AsString;
   end;
   Acao.Free;
end;

procedure TCGrupo.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.10 unit TItensOrdemDeServico;

```
interface

uses
   Classes, PSQLDbTables, SysUtils, TSubItensOrdemDeServico;

type
  TcItensOrdemDeServico = class(TComponent)
    private
      F_id_item_os         : Integer;
      F_id_os              : Integer;
      F_id_contrato        : Integer;
      F_id_equipamento     : Integer;
      F_desc_equip         : String;
      F_desc_prob          : String;
      F_tipo_atendimento   : Integer;
    public
      SubItensOS : TcSubItensOrdemDeServico;

      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_item_os       : Integer read F_id_item_os       write F_id_item_os;
      property P_id_os            : Integer read F_id_os            write F_id_os;
      property P_id_contrato      : Integer read F_id_contrato      write F_id_contrato;
      property P_id_equipamento   : Integer read F_id_equipamento   write
F_id_equipamento;
      property P_tipo_atendimento : Integer read F_tipo_atendimento write
F_tipo_atendimento;
      property P_desc_equip       : String  read F_desc_equip       write F_desc_equip;
      property P_desc_prob        : String  read F_desc_prob        write F_desc_prob;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_id_os: Integer);
      procedure Selecionar(wp_id_item_os: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TTipoDePeca }

procedure TcItensOrdemDeServico.Anterior;
```

```
begin
   Ocolecao.Prior;
end;

procedure TcItensOrdemDeServico.Coloca;
begin
   F_id_item_os       := OColecao.FieldByName('id_item_os').AsInteger;
   F_id_os            := OColecao.FieldByName('id_os').AsInteger;
   F_id_contrato      := OColecao.FieldByName('id_contrato').AsInteger;
   F_id_equipamento   := OColecao.FieldByName('id_equipamento').AsInteger;
   F_tipo_atendimento := OColecao.FieldByName('tipo_atendimento').AsInteger;
   F_desc_equip       := OColecao.FieldByName('desc_equip').AsString;
   F_desc_prob        := OColecao.FieldByName('desc_prob').AsString;
end;

function TcItensOrdemDeServico.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcItensOrdemDeServico.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  SubItensOS := TcSubItensOrdemDeServico.Create(nil);
  Ocolecao   := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcItensOrdemDeServico.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcItensOrdemDeServico.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcItensOrdemDeServico.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcItensOrdemDeServico.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
//Estado = 2 >> Excluir
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into item_os (id_os,  desc_equip, desc_prob,
tipo_atendimento');
         if (F_id_contrato <> 0) and (F_id_equipamento <> 0) then
         begin
            ws_Sql := (ws_Sql +', id_contrato, id_equipamento ');
         end;
         ws_Sql := (ws_Sql +' )Values (:p_id_os, :p_desc_equip, :p_desc_prob,
:P_tipo_atendimento ');
         if (F_id_contrato <> 0) and (F_id_equipamento <> 0) then
         begin
            ws_Sql := (ws_Sql +', :p_id_contrato, :p_id_equipamento ');
         end;
         ws_Sql := (ws_Sql +')');
      end
   else
   if (estado = 1) then
   begin
      ws_Sql := ('Update item_os set ');
      ws_Sql := (ws_Sql + ' id_os            = :p_id_os,           ');
      if (F_id_contrato <> 0) and (F_id_equipamento <> 0) then
      begin
         ws_Sql := (ws_Sql + ' id_contrato    = :p_id_contrato,     ');
         ws_Sql := (ws_Sql + ' id_equipamento = :p_id_equipamento,  ');
```

```
      end;
      ws_Sql := (ws_Sql + ' tipo_atendimento  = :P_tipo_atendimento, ');
      ws_Sql := (ws_Sql + ' desc_equip        = :p_desc_equip,       ');
      ws_Sql := (ws_Sql + ' desc_prob         = :p_desc_prob         ');
      ws_Sql := (ws_Sql + ' where (id_item_os = :p_id_item_os)       ');
   end
   else
   if (estado = 2) then
   begin
      ws_Sql := ('delete from item_os where ((id_os = :p_id_os ) and (id_item_os =
:p_id_item_os))');
   end;

      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) or (estado = 2) then
      begin
         Acao.ParamByName('p_id_item_os').AsInteger     := F_id_item_os;
      end;
      Acao.ParamByName('p_id_os').AsInteger             := F_id_os;
      if (estado <> 2) then
      begin
         if (F_id_contrato <> 0) and (F_id_equipamento <> 0) then
         begin
            Acao.ParamByName('p_id_contrato').AsInteger    := F_id_contrato;
            Acao.ParamByName('p_id_equipamento').AsInteger := F_id_equipamento;
         end;
         Acao.ParamByName('p_tipo_atendimento').AsInteger  := F_tipo_atendimento;
         Acao.ParamByName('p_desc_equip').AsString         := F_desc_equip;
         Acao.ParamByName('p_desc_prob').AsString          := F_desc_prob;
      END;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcItensOrdemDeServico.Seleciona(wp_id_os: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from item_os where (id_os = :p_id_os)';
   WS_SQL := (WS_SQL + ' order by id_item_os, id_os');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_os').Value  := wp_id_os;
   Ocolecao.Open;
end;

procedure TcItensOrdemDeServico.Selecionar(wp_id_item_os: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from item_os where (id_item_os = :p_id_item_os)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_item_os').Value  := wp_id_item_os;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_item_os       := 0;
      F_id_os            := 0;
      F_id_contrato      := 0;
      F_id_equipamento   := 0;
      F_tipo_atendimento := -1;
      F_desc_equip       := '';
      F_desc_prob        := '';
   end
   else
   begin
      F_id_item_os       := Acao.FieldByName('id_item_os').AsInteger;
      F_id_os            := Acao.FieldByName('id_os').AsInteger;
      F_id_contrato      := Acao.FieldByName('id_contrato').AsInteger;
      F_id_equipamento   := Acao.FieldByName('id_equipamento').AsInteger;
      F_tipo_atendimento := Acao.FieldByName('tipo_atendimento').AsInteger;
      F_desc_equip       := Acao.FieldByName('desc_equip').AsString;
```

```
      F_desc_prob         := Acao.FieldByName('desc_prob').AsString;
   end;
   Acao.Free;
end;

procedure TcItensOrdemDeServico.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//      - José Leonardo Sivek                                                 //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.11 unit TLogOS;

```
interface

uses
   Classes, PSQLDbTables, Controls, SysUtils ;

type
  TcLogOS = class(TComponent)
    private
      F_id_log       : Integer;
      F_id_os        : Integer;
      F_id_usuario   : Integer;
      F_acao         : String;
      F_data         : TDate;
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_log     : Integer read F_id_log      write F_id_log;
      property P_id_os      : Integer read F_id_os       write F_id_os;
      property P_id_usuario : Integer read F_id_usuario  write F_id_usuario;
      property P_acao       : String  read F_acao        write F_acao;
      property P_data       : TDate   read F_data        write F_data;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Ultimo;
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TcLogOS.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcLogOS.Coloca;
begin
   F_id_log       := OColecao.FieldByName('id_log').AsInteger;
   F_id_os        := OColecao.FieldByName('id_os').AsInteger;
   F_id_usuario   := OColecao.FieldByName('id_usuario').AsInteger;
   F_acao         := OColecao.FieldByName('acao').asstring;
   F_data         := OColecao.FieldByName('data').AsDateTime;
end;
```

```
function TcLogOS.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcLogOS.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcLogOS.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcLogOS.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcLogOS.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcLogOS.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into logs_os ');
         ws_Sql := (ws_Sql + ' (id_os, id_usuario, acao, data ) ');
         ws_Sql := (ws_Sql + ' Values ');
         ws_Sql := (ws_Sql + ' (:P_id_os, :P_id_usuario, :P_acao, :P_data) ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      Acao.ParamByName('P_id_os').AsInteger      := F_id_os;
      Acao.ParamByName('P_id_usuario').AsInteger := F_id_usuario;
      Acao.ParamByName('P_acao').asstring        := F_acao;
      Acao.ParamByName('P_data').AsDateTime      := F_data;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcLogOS.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from logs_os';
   WS_SQL := (WS_SQL + ' order by id_os');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcLogOS.Ultimo;
begin
   Ocolecao.Last;
end;


end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.12 unit TMenu;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TCMenu = class(TComponent)

    private
      F_id_menu   : String;
      F_tipo_menu : Integer;
      F_descricao : String;
      F_id_modulo : Integer;
      F_sequencia : Integer;
      F_menupai   : Integer;

    public

      //Property's
      property P_id_menu   : String  read F_id_menu   write F_id_menu;
      property P_tipo_menu : Integer read F_tipo_menu write F_tipo_menu;
      property P_descricao : String  read F_descricao write F_descricao;
      property P_id_modulo : Integer read F_id_modulo write F_id_modulo;
      property P_sequencia : Integer read F_sequencia write F_sequencia;
      property P_menupai   : Integer read F_menupai   write F_menupai;

      constructor Create(Dono: TComponent);
      destructor Destroy; override;

      //Operações
      function  Pesq_id_Menu(wfs_Menu: String): Boolean;
      function  Pesq_desc_Menu(wfs_Menu: String): Boolean;
      procedure Salvar(Estado: Integer);

      // operacoes na coleção
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Seleciona;
      procedure Ultimo;


  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;
{ TCMenu }
```

```
constructor TCMenu.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

destructor TCMenu.Destroy;
begin
  inherited;
  OColecao.FreeOnRelease;
end;

function TCMenu.Pesq_id_Menu(wfs_Menu: String): Boolean;
begin
  try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   Acao.Close;
   ws_Sql := ('Select * from menu where (Id_Menu = :p_Menu)');
   ws_sql:= LowerCase(ws_sql);
   Acao.SQL.add(ws_sql);
   Acao.ParamByName('p_Menu').AsString := wfs_Menu;
   Acao.Open;
   Result := False;

   if not Acao.IsEmpty then
   begin
      F_tipo_menu := Acao.FieldByName('Tipo_Menu').AsInteger;
      F_sequencia := Acao.FieldByName('Sequencia').AsInteger;
      F_menupai   := Acao.FieldByName('MenuPai').AsInteger;
      F_descricao := Acao.FieldByName('Descricao').AsString;
      F_id_modulo := Acao.FieldByName('Id_Modulo').AsInteger;
      F_id_menu   := Acao.FieldByName('Id_Menu').AsString;
      Result := True;
   end;

   Finally
      FreeAndNil(Acao);
   End;
end;

procedure TCMenu.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar

   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         //INSERINDO DADOS NA TABELA
         ws_Sql := ('Insert into menu (Id_Menu, Tipo_Menu, Descricao,');
         ws_Sql := (ws_Sql +' Sequencia, MenuPai, Id_Modulo)');
         ws_Sql := (ws_Sql +' Values (:p_Id_Menu, :p_Tipo_Menu, :p_Descricao,');
         ws_Sql := (ws_Sql +' :p_Sequencia, :p_MenuPai, :p_Id_Modulo)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update menu set ');
         ws_Sql := (ws_Sql + ' Tipo_Menu      = :p_Tipo_Menu , ');
         ws_Sql := (ws_Sql + ' Sequencia      = :p_Sequencia , ');
         ws_Sql := (ws_Sql + ' MenuPai        = :p_MenuPai   , ');
         ws_Sql := (ws_Sql + ' Descricao      = :p_Descricao , ');
         ws_Sql := (ws_Sql + ' Id_Modulo      = :p_Id_Modulo   ');
         ws_Sql := (ws_Sql + ' where (Id_Menu = :p_Id_Menu)    ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      Acao.ParamByName('p_Tipo_Menu').AsInteger := F_tipo_menu;
      Acao.ParamByName('p_Sequencia').AsInteger := F_sequencia;
      Acao.ParamByName('p_MenuPai').AsInteger   := F_menupai;
      Acao.ParamByName('p_Descricao').AsString  := F_descricao;
```

```
      Acao.ParamByName('p_Id_Modulo').AsInteger := F_id_modulo;
      Acao.ParamByName('p_Id_Menu').AsString    := F_id_menu;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCMenu.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from menu';
   WS_SQL := (WS_SQL + ' order by id_modulo, tipo_menu, sequencia');
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCMenu.Ultimo;
begin
   Ocolecao.Last;
end;

procedure TCMenu.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCMenu.Coloca;
begin
   F_tipo_menu := OColecao.FieldByName('Tipo_Menu').AsInteger;
   F_sequencia := OColecao.FieldByName('Sequencia').AsInteger;
   F_menupai   := OColecao.FieldByName('MenuPai').AsInteger;
   F_descricao := OColecao.FieldByName('Descricao').AsString;
   F_id_modulo := OColecao.FieldByName('Id_Modulo').AsInteger;
   F_id_menu   := OColecao.FieldByName('Id_Menu').AsString;
end;

function TCMenu.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

procedure TCMenu.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCMenu.Proximo;
begin
   Ocolecao.Next;
end;

function TCMenu.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

function TCMenu.Pesq_desc_Menu(wfs_Menu: String): Boolean;
begin
  try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   Acao.Close;
   ws_Sql := ('Select * from menu where (Descricao = :p_Menu)');
   ws_sql:= LowerCase(ws_sql);
   Acao.SQL.add(ws_sql);
   Acao.ParamByName('p_Menu').AsString := wfs_Menu;
   Acao.Open;
   Result := False;
   if not Acao.IsEmpty then
   begin
      F_tipo_menu := Acao.FieldByName('Tipo_Menu').AsInteger;
      F_sequencia := Acao.FieldByName('Sequencia').AsInteger;
      F_menupai   := Acao.FieldByName('MenuPai').AsInteger;
      F_descricao := Acao.FieldByName('Descricao').AsString;
      F_id_modulo := Acao.FieldByName('Id_Modulo').AsInteger;
      F_id_menu   := Acao.FieldByName('Id_Menu').AsString;
```

```
      Result := True;
   end;
   Finally
      FreeAndNil(Acao);
   End;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.13 unit TOrdemDeServico;

```
interface

uses
   Classes, PSQLDbTables, SysUtils, Controls, TItensOrdemDeServico;

type
  TcOrdemDeServico = class(TComponent)
    private
      F_id_os                : Integer;
      F_id_pessoa            : Integer;
      F_status               : wideString;
      F_data_cad             : TDate;
      F_data_alt             : TDate;
      F_data_fechou          : TDate;
      F_data_canc            : TDate;
      F_valor_orcado         : Double;
      F_valor_cobrado        : Double;
      F_valor_mao_obra       : Double;
      F_valor_pecas          : Double;
      F_valor_pecas_ncobrado : Double;
      F_valor_desconto_mo    : Double;
      F_valor_desconto_pecas : Double;


    public
      Items_OS : TcItensOrdemDeServico;
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_os                : Integer    read F_id_os                write
F_id_os;
      property P_id_pessoa            : Integer    read F_id_pessoa            write
F_id_pessoa;
      property P_status               : wideString read F_status               write
F_status;
      property P_data_cad             : TDate      read F_data_cad             write
F_data_cad;
      property P_data_alt             : TDate      read F_data_alt             write
F_data_alt;
      property P_data_fechou          : TDate      read F_data_fechou          write
F_data_fechou;
      property P_data_canc            : TDate      read F_data_canc            write
F_data_canc;
      property P_valor_orcado         : Double     read F_valor_orcado         write
F_valor_orcado;
      property P_valor_cobrado        : Double     read F_valor_cobrado        write
F_valor_cobrado;
      property P_valor_mao_obra       : Double     read F_valor_mao_obra       write
F_valor_mao_obra;
      property P_valor_pecas          : Double     read F_valor_pecas          write
F_valor_pecas;
      property P_valor_pecas_ncobrado : Double     read F_valor_pecas_ncobrado write
F_valor_pecas_ncobrado;
      property P_valor_desconto_mo    : Double     read F_valor_desconto_mo    write
F_valor_desconto_mo;
```

```
      property P_valor_desconto_pecas  : Double     read F_valor_desconto_pecas  write
F_valor_desconto_pecas;


      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_id_os: Integer; wp_status : String);
      procedure Pesquisar(Estado,Tipo: Integer);
      procedure Selecionar(wp_id_os : Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TcOrdemDeServico.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcOrdemDeServico.Coloca;
begin
   F_id_os                := OColecao.FieldByName('id_os').AsInteger;
   F_id_pessoa            := OColecao.FieldByName('id_pessoa').AsInteger;
   F_status               := OColecao.FieldByName('status').AsString;
   F_data_cad             := OColecao.FieldByName('data_cad').AsDateTime;
   F_data_alt             := OColecao.FieldByName('data_alt').AsDateTime;
   F_data_fechou          := OColecao.FieldByName('data_fechou').AsDateTime;
   F_data_canc            := OColecao.FieldByName('data_canc').AsDateTime;
   F_valor_orcado         := OColecao.FieldByName('valor_orcado').AsFloat;
   F_valor_cobrado        := OColecao.FieldByName('valor_cobrado').AsFloat;
   F_valor_mao_obra       := OColecao.FieldByName('valor_mao_obra').AsFloat;
   F_valor_pecas          := OColecao.FieldByName('valor_pecas').AsFloat;
   F_valor_pecas_ncobrado := OColecao.FieldByName('valor_pecas_ncobrado').AsFloat;
   F_valor_desconto_mo    := OColecao.FieldByName('valor_desconto_mo').AsFloat;
   F_valor_desconto_pecas := OColecao.FieldByName('valor_desconto_pecas').AsFloat;
end;

function TcOrdemDeServico.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcOrdemDeServico.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Items_OS           := TcItensOrdemDeServico.Create(nil);
  Ocolecao           := TPSQLQuery.Create(Self);
  Ocolecao.Database  := dm_base.my_conn;
end;

function TcOrdemDeServico.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcOrdemDeServico.Pesquisar(Estado,Tipo: Integer);
begin
//Tipo = 0 >> Aprovadas
//Tipo = 1 >> Fechadas
//Tipo = 2 >> Canceladas
//Tipo = 3 >> Fechadas e Liberadas
//Tipo = 5 >> Orçamento
//Tipo = 6 >> Orçando
//Tipo = 7 >> Orçado
//Tipo = 8 >> Recusada
```

```
//Tipo = 9 >> Fechadas e Liberadas e Devolução

//Tipo = 4 >> Lista todas as Ordens de Serviço
//Estado = 6 >> MOSTRA TODAS

   Ocolecao.Close;
   WS_SQL := ('select * from os ');
   case Estado of
      0    :   begin
                  if Tipo = 9 then
                  begin
                     ws_Sql := (ws_sql + ' where (status like :p_status) or (status =
:p_status1) or (status = :p_status2)');
                  end
                  else
                  if Tipo = 3 then
                  begin
                     ws_Sql := (ws_sql + ' where (status like :p_status) or (status =
:p_status1) ');
                  end
                  else
                     ws_Sql := (ws_sql + ' where (status like :p_status)');
               end;
      1..5 : ws_Sql := (ws_sql + ' where (status = :p_status)');
   end;
   ws_Sql := (ws_sql + ' order by id_os desc ');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   case estado of
      0 : begin
            if Tipo = 0 then
               Ocolecao.ParamByName('p_status').AsString  := 'APROVADA%';
            if Tipo = 1 then
               Ocolecao.ParamByName('p_status').AsString  := 'FECHADA';
            if Tipo = 2 then
               Ocolecao.ParamByName('p_status').AsString  := 'CANCELADA';
            if Tipo = 3 then
            begin
               Ocolecao.ParamByName('p_status').AsString  := 'FECHADA';
               Ocolecao.ParamByName('p_status1').AsString := 'APROVADA - LIBERADA';
            end;
            if Tipo = 5 then
               Ocolecao.ParamByName('p_status').AsString  := 'ORÇAMENTO';
            if Tipo = 6 then
               Ocolecao.ParamByName('p_status').AsString  := 'ORÇAR';
            if Tipo = 7 then
               Ocolecao.ParamByName('p_status').AsString  := 'ORÇADO';
            if Tipo = 8 then
               Ocolecao.ParamByName('p_status').AsString  := 'RECUSADA';
            if Tipo = 9 then
            begin
               Ocolecao.ParamByName('p_status').AsString  := 'FECHADA';
               Ocolecao.ParamByName('p_status1').AsString := 'APROVADA - LIBERADA';
               Ocolecao.ParamByName('p_status2').AsString := 'DEVOLVER';
            end;
         end;
      1 :   begin
               if Tipo = 3 then
               begin
                  Ocolecao.ParamByName('p_status').AsString  := 'APROVADA - LIBERADA';
               end
               else
                  Ocolecao.ParamByName('p_status').AsString  := 'APROVADA';
            end;
      2 :   begin
               if Tipo = 3 then
               begin
                  Ocolecao.ParamByName('p_status').AsString  := 'FECHADA';
               end
               else
                  Ocolecao.ParamByName('p_status').AsString  := 'APROVADA - EM
ANDAMENTO';
            end;
      3 :   begin
               if Tipo = 3 then
               begin
                  Ocolecao.ParamByName('p_status').AsString  := 'DEVOLVER';
```

```
                end
                else
                   Ocolecao.ParamByName('p_status').AsString  := 'APROVADA - AGUARDANDO
PEÇA';
             END;
       4 : Ocolecao.ParamByName('p_status').AsString  := 'APROVADA - LIBERADA';
    end;
    Ocolecao.Open;
end;

procedure TcOrdemDeServico.Primeiro;
begin
    Ocolecao.First;
end;

procedure TcOrdemDeServico.Proximo;
begin
    Ocolecao.Next;
end;

procedure TcOrdemDeServico.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
//Estado = 2 >> Excluir
    try
    Acao := TPSQLQuery.Create(Self);
    Acao.Database := dm_base.my_conn;
    Acao.Close;
    if Estado = 2 Then
    begin
      Acao.Close;
      Acao.SQL.Clear;
      ws_Sql := ('delete from os where (id_os = :p_id_os) ');
      Acao.SQL.Add(ws_sql);
      Acao.ParamByName('p_id_os').AsInteger := F_id_os;
      Acao.ExecSQL;
    end
    else
    begin
       if (estado = 0) then
          begin
             ws_Sql := ('Insert into os (id_pessoa, status, data_cad, valor_orcado, ');
             ws_Sql := (ws_Sql +' valor_cobrado, valor_mao_obra,valor_pecas, ');
             ws_Sql := (ws_Sql +' valor_pecas_ncobrado, valor_desconto_mo,
valor_desconto_pecas) ');
             ws_Sql := (ws_Sql +' Values (:p_id_pessoa, :p_status, :p_data_cad, ');
             ws_Sql := (ws_Sql +' :p_valor_orcado, :p_valor_cobrado, :p_valor_mao_obra,
');
             ws_Sql := (ws_Sql +' :p_valor_pecas, :p_valor_pecas_ncobrado,
:p_valor_desconto_mo, :p_valor_desconto_pecas) ');
          end
       else
       if (estado = 1) then
          begin
             ws_Sql := ('Update os set ');
             ws_Sql := (ws_Sql + ' id_pessoa            = :p_id_pessoa,           ');
             ws_Sql := (ws_Sql + ' status               = :p_status,              ');
             ws_Sql := (ws_Sql + ' data_cad             = :p_data_cad,            ');
             if F_data_alt <> 0 then
                ws_Sql := (ws_Sql + ' data_alt          = :p_data_alt,            ');
             if F_data_fechou <> 0 then
                ws_Sql := (ws_Sql + ' data_fechou       = :p_data_fechou,         ');
             if F_data_canc <> 0 then
                ws_Sql := (ws_Sql + ' data_canc         = :p_data_canc,           ');
             ws_Sql := (ws_Sql + ' valor_orcado         = :p_valor_orcado,        ');
             ws_Sql := (ws_Sql + ' valor_cobrado        = :p_valor_cobrado,       ');
             ws_Sql := (ws_Sql + ' valor_mao_obra       = :p_valor_mao_obra,      ');
             ws_Sql := (ws_Sql + ' valor_pecas          = :p_valor_pecas,         ');
             ws_Sql := (ws_Sql + ' valor_pecas_ncobrado = :p_valor_pecas_ncobrado,');
             ws_Sql := (ws_Sql + ' valor_desconto_mo    = :p_valor_desconto_mo,   ');
             ws_Sql := (ws_Sql + ' valor_desconto_pecas = :p_valor_desconto_pecas ');
             ws_Sql := (ws_Sql + ' where (id_os         = :p_id_os)               ');
          end;
          ws_sql := lowercase(ws_sql);
          Acao.SQL.CLEAR;
          Acao.SQL.Add(ws_sql);
```

```
        //ATRIBUI PARÂMETROS A CONSULTA
        if (estado = 1) then
        begin
           Acao.ParamByName('p_id_os').AsInteger           := F_id_os;
           if F_data_alt <> 0 then
              Acao.ParamByName('p_data_alt').AsDateTime    := F_data_alt;
           if F_data_fechou <> 0 then
              Acao.ParamByName('p_data_fechou').AsDateTime := F_data_fechou;
           if F_data_canc <> 0 then
              Acao.ParamByName('p_data_canc').AsDateTime   := F_data_canc;
        end;
        Acao.ParamByName('p_id_pessoa').AsInteger          := F_id_pessoa;
        Acao.ParamByName('p_status').AsString              := F_status;
        Acao.ParamByName('p_data_cad').AsDateTime          := F_data_cad;
        Acao.ParamByName('p_valor_orcado').AsFloat         := F_valor_orcado;
        Acao.ParamByName('p_valor_cobrado').AsFloat        := F_valor_cobrado;
        Acao.ParamByName('p_valor_mao_obra').AsFloat       := F_valor_mao_obra;
        Acao.ParamByName('p_valor_pecas').AsFloat          := F_valor_pecas;
        Acao.ParamByName('p_valor_pecas_ncobrado').AsFloat := F_valor_pecas_ncobrado;
        Acao.ParamByName('p_valor_desconto_mo').AsFloat    := F_valor_desconto_mo;
        Acao.ParamByName('p_valor_desconto_pecas').AsFloat := F_valor_desconto_pecas;
        Acao.ExecSQL;
        if Estado = 0 Then
        begin
          Acao.Close;
          Acao.SQL.Clear;
          ws_Sql := ('select max(id_os) as n_os from os');
          Acao.SQL.Add(ws_sql);
          Acao.Open;
          F_id_os := Acao.FieldByName('n_os').AsInteger;
        end;
   end;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcOrdemDeServico.Seleciona(wp_id_os : Integer; wp_status : String);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from os where (id_os = :p_id_os) and (status = :p_status) order
by id_os desc ';
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_os').AsInteger  := wp_id_os;
   Acao.ParamByName('p_status').AsString  := wp_status;
   Ocolecao.Open;
end;

procedure TcOrdemDeServico.Selecionar(wp_id_os : Integer);
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   Acao.close;
   WS_SQL := 'select * from os where (id_os = :p_id_os)';
   Acao.SQL.Clear;
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_os').AsInteger  := wp_id_os;
   Acao.Open;
   if Acao.IsEmpty then
   begin
      F_id_os                := 0;
      F_id_pessoa            := 0;
      F_status               := '';
      F_data_cad             := 0;
      F_data_alt             := 0;
      F_data_fechou          := 0;
      F_data_canc            := 0;
      F_valor_orcado         := 0;
      F_valor_cobrado        := 0;
      F_valor_mao_obra       := 0;
      F_valor_pecas          := 0;
      F_valor_pecas_ncobrado := 0;
      F_valor_desconto_mo    := 0;
      F_valor_desconto_pecas := 0;
   end
   else
```

```
   begin
      F_id_os                := Acao.FieldByName('id_os').AsInteger;
      F_id_pessoa            := Acao.FieldByName('id_pessoa').AsInteger;
      F_status               := Acao.FieldByName('status').AsString;
      F_data_cad             := Acao.FieldByName('data_cad').AsDateTime;
      F_data_alt             := Acao.FieldByName('data_alt').AsDateTime;
      F_data_fechou          := Acao.FieldByName('data_fechou').AsDateTime;
      F_data_canc            := Acao.FieldByName('data_canc').AsDateTime;
      F_valor_orcado         := Acao.FieldByName('valor_orcado').AsFloat;
      F_valor_cobrado        := Acao.FieldByName('valor_cobrado').AsFloat;
      F_valor_mao_obra       := Acao.FieldByName('valor_mao_obra').AsFloat;
      F_valor_pecas          := Acao.FieldByName('valor_pecas').AsFloat;
      F_valor_pecas_ncobrado := Acao.FieldByName('valor_pecas_ncobrado').AsFloat;
      F_valor_desconto_mo    := Acao.FieldByName('valor_desconto_mo').AsFloat;
      F_valor_desconto_pecas := Acao.FieldByName('valor_desconto_pecas').AsFloat;
   end;
   Acao.free;
end;

procedure TcOrdemDeServico.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.14 unit TPeca;

```
interface

uses
   Classes, PSQLDbTables, SysUtils, TTipoDePeca;

type
  TcPeca = class(TComponent)

    private
      F_id_peca         : Integer;
      F_id_tipo_peca    : Integer;
      F_descricao       : String;
      F_quantidade      : Integer;
      F_valor_unitario  : Double;
    public
      Tipo_da_Peca : TcTipoDePeca;

      //Property's
      property P_id_peca       : Integer read F_id_peca        write F_id_peca;
      property P_id_tipo_peca  : Integer read F_id_tipo_peca   write F_id_tipo_peca;
      property P_descricao     : String  read F_descricao      write F_descricao;
      property P_quantidade    : Integer read F_quantidade     write F_quantidade;
      property P_valor_unitario: Double  read F_valor_unitario write F_valor_unitario;

      constructor Create(Dono: TComponent);

      //Operações
      procedure Salvar(Estado: Integer);
      procedure Selecionar(wp_id_peca: Integer);
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Seleciona;
      procedure Listar(Tipo: Integer);
      procedure Ultimo;
      procedure Pesquisar(wp_desc_peca: String);
  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;


{ TcPeca }

constructor TcPeca.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Tipo_da_Peca := TcTipoDePeca.Create(nil);
```

```
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

procedure TcPeca.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar

   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         //INSERINDO DADOS NA TABELA
         ws_Sql := ('Insert into peca (id_tipo_peca, descricao, quantidade,
valor_unitario)');
         ws_Sql := (ws_Sql +' Values (:p_id_tipo_peca, :p_descricao, :p_quantidade,');
         ws_Sql := (ws_Sql +' :p_valor_unitario)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update peca set ');
         ws_Sql := (ws_Sql + ' id_tipo_peca   = :p_id_tipo_peca , ');
         ws_Sql := (ws_Sql + ' descricao      = :p_descricao ,    ');
         ws_Sql := (ws_Sql + ' quantidade     = :p_quantidade,    ');
         ws_Sql := (ws_Sql + ' valor_unitario = :p_valor_unitario ');
         ws_Sql := (ws_Sql + ' where (id_Peca = :P_id_Peca)       ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('P_id_Peca').AsInteger      := F_id_Peca;
      end;
      Acao.ParamByName('p_id_tipo_peca').AsInteger    := F_id_tipo_peca;
      Acao.ParamByName('p_descricao').AsString        := F_descricao;
      Acao.ParamByName('p_quantidade').AsInteger      := F_quantidade;
      Acao.ParamByName('p_valor_unitario').AsFloat    := F_valor_unitario;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcPeca.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from peca';
   WS_SQL := (WS_SQL + ' order by id_Peca, id_tipo_peca, descricao');
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcPeca.Ultimo;
begin
   Ocolecao.Last;
end;

procedure TcPeca.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcPeca.Coloca;
begin
   F_id_tipo_peca    := OColecao.FieldByName('id_tipo_peca').AsInteger;
   F_descricao       := OColecao.FieldByName('descricao').AsString;
   F_quantidade      := OColecao.FieldByName('quantidade').AsInteger;
   F_valor_unitario  := OColecao.FieldByName('valor_unitario').AsFloat;
   F_id_Peca         := OColecao.FieldByName('id_Peca').AsInteger;
end;

function TcPeca.Comeco: Boolean;
```

```
begin
   Result := OColecao.Bof;
end;

procedure TcPeca.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcPeca.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcPeca.Selecionar(wp_id_peca: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from peca where (id_Peca = :p_id_Peca)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_Peca').Value  := wp_id_peca;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_tipo_peca    := 0;
      F_descricao       := '';
      F_quantidade      := 0;
      F_valor_unitario  := 0;
      F_id_Peca         := 0;
   end
   else
   begin
      F_id_tipo_peca    := Acao.FieldByName('id_tipo_peca').AsInteger;
      F_descricao       := Acao.FieldByName('descricao').AsString;
      F_quantidade      := Acao.FieldByName('quantidade').AsInteger;
      F_valor_unitario  := Acao.FieldByName('valor_unitario').AsFloat;
      F_id_Peca         := Acao.FieldByName('id_Peca').AsInteger;
   end;
   Acao.Free;
end;

function TcPeca.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcPeca.Pesquisar(wp_desc_peca: String);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from peca where ';
   WS_SQL := (WS_SQL + ' (descricao like ' + QuotedStr('%' + wp_desc_peca + '%')+')');
   WS_SQL := (WS_SQL + ' order by descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcPeca.Listar(Tipo: Integer);
begin
   Ocolecao.Close;
   WS_SQL := ('select * from peca ');
   if (tipo <> 0) then
      WS_SQL := (WS_SQL + ' where (id_tipo_peca = :p_tipo) ');

   WS_SQL := (WS_SQL + ' order by id_Peca, id_tipo_peca, descricao');
   Ocolecao.SQL.Add(WS_SQL);
   if (tipo <> 0) then
      Ocolecao.ParamByName('p_tipo').AsInteger := tipo;
   Ocolecao.Open;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.15 unit TPessoa;

```
interface

uses
   Classes, PSQLDbTables, SysUtils,TContatos,TCidade, TContrato, TEquipamento;

type
  TcPessoa = class(TComponent)
    private
      F_id_pessoa     : Integer;
      F_id_cidade     : Integer;
      F_id_UF         : Integer;
      F_ie_rg         : String;
      F_cgc_cnpj      : String;
      F_tipo_pessoa   : Integer;
      F_tipo          : Integer;
      F_nome          : String;
      F_endereco      : String;
      F_bairro        : String;
      F_cep           : String;
      F_contato       : String;
      F_email         : String;
      F_homepage      : String;
      F_obs           : String;
    public
      Contatos     : TcContatos;
      Cidade       : TcCidade;
      Contrato     : TcContrato;
      Equipamento  : TCequipamento;
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_pessoa     : Integer read F_id_pessoa     write F_id_pessoa;
      property P_id_cidade     : Integer read F_id_cidade     write F_id_cidade;
      property P_id_UF         : Integer read F_id_UF         write F_id_UF;
      property P_ie_rg         : String  read F_ie_rg         write F_ie_rg;
      property P_cgc_cnpj      : String  read F_cgc_cnpj      write F_cgc_cnpj;
      property P_tipo_pessoa   : Integer read F_tipo_pessoa   write F_tipo_pessoa;
      property P_tipo          : Integer read F_tipo          write F_tipo;
      property P_nome          : String  read F_nome          write F_nome;
      property P_endereco      : String  read F_endereco      write F_endereco;
      property P_bairro        : String  read F_bairro        write F_bairro;
      property P_cep           : String  read F_cep           write F_cep;
      property P_contato       : String  read F_contato       write F_contato;
      property P_email         : String  read F_email         write F_email;
      property P_homepage      : String  read F_homepage      write F_homepage;
      property P_obs           : String  read F_obs           write F_obs;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado,wp_tipo: Integer);
```

```
      procedure Seleciona(wp_tipo: Integer);
      procedure Selecionar(wp_id_pessoa, wp_tipo : Integer);
      procedure Pesquisar(wp_valor : String; wp_cpf0_ou_nome1,wp_tipo : Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TcPessoa.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcPessoa.Coloca;
begin
   F_id_pessoa    := OColecao.FieldByName('id_pessoa').AsInteger;
   F_id_cidade    := OColecao.FieldByName('id_cidade').AsInteger;
   F_id_UF        := OColecao.FieldByName('id_uf').AsInteger;
   F_ie_rg        := OColecao.FieldByName('ie_rg').AsString;
   F_cgc_cnpj     := OColecao.FieldByName('cgc_cnpj').AsString;
   F_tipo_pessoa  := OColecao.FieldByName('tipo_pessoa').AsInteger;
   F_tipo         := OColecao.FieldByName('tipo').AsInteger;
   F_nome         := OColecao.FieldByName('nome').AsString;
   F_endereco     := OColecao.FieldByName('endereco').AsString;
   F_bairro       := OColecao.FieldByName('bairro').AsString;
   F_cep          := OColecao.FieldByName('cep').AsString;
   F_contato      := OColecao.FieldByName('contato').AsString;
   F_email        := OColecao.FieldByName('email').AsString;
   F_homepage     := OColecao.FieldByName('homepage').AsString;
   F_obs          := OColecao.FieldByName('obs').AsString;
end;

function TcPessoa.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcPessoa.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Contatos           := TcContatos.create(nil);
  Cidade             := TcCidade.create(nil);
  Contrato           := TcContrato.create(nil);
  Equipamento        := TCequipamento.Create(nil);
  Ocolecao           := TPSQLQuery.Create(Self);
  Ocolecao.Database  := dm_base.my_conn;
end;

function TcPessoa.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcPessoa.Pesquisar(wp_valor : String; wp_cpf0_ou_nome1,wp_tipo : Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from pessoa where (tipo = :p_tipo) and';

   if wp_cpf0_ou_nome1 = 0 then
      WS_SQL := (WS_SQL + ' (cgc_cnpj like ' + QuotedStr('%' + wp_valor + '%')+')')
   else
      if wp_cpf0_ou_nome1 = 1 then
         WS_SQL := (WS_SQL + ' (nome     like ' + QuotedStr('%' + wp_valor + '%')+')');

   WS_SQL := (WS_SQL + ' order by id_pessoa, nome');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_tipo').AsInteger  := wp_tipo;
   Ocolecao.Open;
end;
```

```
procedure TcPessoa.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcPessoa.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcPessoa.Salvar(Estado,wp_tipo: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into pessoa (id_cidade, id_uf, nome, endereco, bairro, cep,
ie_rg, cgc_cnpj, ');
         ws_Sql := (ws_Sql +' tipo_pessoa, tipo, contato, email, homepage, obs)');
         ws_Sql := (ws_Sql +' Values (:p_id_cidade, :p_id_uf, :p_nome, :p_endereco,
:p_bairro, :p_cep, :p_ie_rg, :p_cgc_cnpj, ');
         ws_Sql := (ws_Sql +' :p_tipo_pessoa, :p_tipo, :p_contato, :p_email,
:p_homepage, :p_obs)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update pessoa set ');
         ws_Sql := (ws_Sql + ' id_cidade   = :p_id_cidade, ');
         ws_Sql := (ws_Sql + ' id_uf       = :p_id_uf, ');
         ws_Sql := (ws_Sql + ' nome        = :p_nome, ');
         ws_Sql := (ws_Sql + ' endereco    = :p_endereco, ');
         ws_Sql := (ws_Sql + ' bairro      = :p_bairro, ');
         ws_Sql := (ws_Sql + ' cep         = :p_cep, ');
         ws_Sql := (ws_Sql + ' ie_rg       = :p_ie_rg, ');
         ws_Sql := (ws_Sql + ' cgc_cnpj    = :p_cgc_cnpj, ');
         ws_Sql := (ws_Sql + ' tipo_pessoa = :p_tipo_pessoa, ');
         ws_Sql := (ws_Sql + ' tipo        = :p_tipo, ');
         ws_Sql := (ws_Sql + ' contato     = :p_contato, ');
         ws_Sql := (ws_Sql + ' email       = :p_email, ');
         ws_Sql := (ws_Sql + ' homepage    = :p_homepage, ');
         ws_Sql := (ws_Sql + ' obs         = :p_obs ');
         ws_Sql := (ws_Sql + ' where (id_pessoa = :p_id_pessoa)');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.CLEAR;
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_pessoa').AsInteger := F_id_pessoa;
      end;
      Acao.ParamByName('p_id_cidade').AsInteger    := F_id_cidade;
      Acao.ParamByName('P_id_uf').AsInteger        := F_id_UF;
      Acao.ParamByName('p_nome').AsString          := F_nome;
      Acao.ParamByName('p_ie_rg').AsString         := F_ie_rg;
      Acao.ParamByName('p_cgc_cnpj').AsString      := F_cgc_cnpj;
      Acao.ParamByName('p_tipo_pessoa').AsInteger  := F_tipo_pessoa;
      Acao.ParamByName('p_tipo').AsInteger         := F_tipo;
      Acao.ParamByName('p_endereco').AsString      := F_endereco;
      Acao.ParamByName('p_bairro').AsString        := F_bairro;
      Acao.ParamByName('p_cep').AsString           := F_cep;
      Acao.ParamByName('p_contato').AsString       := F_contato;
      Acao.ParamByName('p_email').AsString         := F_email;
      Acao.ParamByName('p_homepage').AsString      := F_homepage;
      Acao.ParamByName('p_obs').AsString           := F_obs;

      Acao.ExecSQL;

      if (estado = 0) then
      begin
         Acao.Close;
         Acao.SQL.Clear;
```

```
        ws_Sql := ('select max(id_pessoa) as n_id_pess from pessoa');
        Acao.SQL.Add(ws_sql);
        Acao.Open;
        F_id_pessoa := Acao.FieldByName('n_id_pess').AsInteger;
      end;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcPessoa.Seleciona(wp_tipo: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from pessoa where (tipo = :p_tipo)';
   WS_SQL := (WS_SQL + ' order by id_pessoa, nome');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_tipo').AsInteger  := wp_tipo;
   Ocolecao.Open;
end;
procedure TcPessoa.Selecionar(wp_id_pessoa, wp_tipo: Integer);
begin
//tipo 1 = cliente
//tipo 2 = fornecedor
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   Acao.close;
   WS_SQL := 'select * from pessoa where (id_pessoa = :p_id_pessoa) and (tipo =
:p_tipo)';
   Acao.SQL.Clear;
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_pessoa').AsInteger := wp_id_pessoa;
   Acao.ParamByName('p_tipo').AsInteger      := wp_tipo;
   Acao.Open;
   if Acao.IsEmpty then
   begin
      F_id_pessoa    := 0;
      F_id_cidade    := 0;
      F_id_uf        := 0;
      F_ie_rg        := '';
      F_cgc_cnpj     := '';
      F_tipo_pessoa  := 0;
      F_tipo         := 0;
      F_nome         := '';
      F_endereco     := '';
      F_bairro       := '';
      F_cep          := '';
      F_contato      := '';
      F_email        := '';
      F_homepage     := '';
      F_obs          := '';
   end
   else
   begin
      F_id_pessoa    := Acao.FieldByName('id_pessoa').AsInteger;
      F_id_cidade    := Acao.FieldByName('id_cidade').AsInteger;
      F_id_uf        := Acao.FieldByName('id_uf').AsInteger;
      F_ie_rg        := Acao.FieldByName('ie_rg').AsString;
      F_cgc_cnpj     := Acao.FieldByName('cgc_cnpj').AsString;
      F_tipo_pessoa  := Acao.FieldByName('tipo_pessoa').AsInteger;
      F_tipo         := Acao.FieldByName('tipo').AsInteger;
      F_nome         := Acao.FieldByName('nome').AsString;
      F_endereco     := Acao.FieldByName('endereco').AsString;
      F_bairro       := Acao.FieldByName('bairro').AsString;
      F_cep          := Acao.FieldByName('cep').AsString;
      F_contato      := Acao.FieldByName('contato').AsString;
      F_email        := Acao.FieldByName('email').AsString;
      F_homepage     := Acao.FieldByName('homepage').AsString;
      F_obs          := Acao.FieldByName('obs').AsString;
   end;
   Acao.free;
end;
procedure TcPessoa.Ultimo;
begin
   Ocolecao.Last;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.16 unit TSubItensOrdemDeServico;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TcSubItensOrdemDeServico = class(TComponent)
    private
      F_id_Sitem       : Integer;
      F_id_item_os     : Integer;
      F_id_os          : Integer;
      F_id_peca        : Integer;
      F_qtde_peca      : Integer;
      F_cobrar         : Integer;
      F_descricao      : String;
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_Sitem    : Integer read F_id_Sitem    write F_id_Sitem;
      property P_id_item_os  : Integer read F_id_item_os  write F_id_item_os;
      property P_id_os       : Integer read F_id_os       write F_id_os;
      property P_id_peca     : Integer read F_id_peca     write F_id_peca;
      property P_qtde_peca   : Integer read F_qtde_peca   write F_qtde_peca;
      property P_cobrar      : Integer read F_cobrar      write F_cobrar;
      property P_descricao   : String  read F_descricao   write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_id_os,wp_id_Item: Integer);
      procedure Selecionar(wp_id_os,wp_id_Item,wp_id_SubItem: Integer);
      procedure SelecionarPecas(wp_id_os: Integer);
      function  Valor_total_SItens_itens(wp_id_os,wp_id_Item: Integer): Double;
      function  Valor_total_pecas_os(wp_id_os, wp_cobra: Integer): Double;
      procedure Altera_Cobranca_pecas(wp_id_os,wp_id_Item,wp_id_SubItem,
wp_cobrar{1=cobra / 0=Ñ Cobrar} : integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TTipoDePeca }
```

```
procedure TcSubItensOrdemDeServico.Altera_Cobranca_pecas(wp_id_os, wp_id_Item,
wp_id_SubItem, wp_cobrar: integer);
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   ws_Sql := ('          Update item_os_mov set ');
   ws_Sql := (ws_Sql + ' cobrar = :p_cobrar ');
   ws_Sql := (ws_Sql + ' where ((id_Sitem = :p_id_Sitem) and (id_os = :p_id_os ) and
(id_item_os = :p_id_item_os))   ');
   ws_sql := lowercase(ws_sql);
   Acao.SQL.Clear;
   Acao.SQL.Add(ws_sql);
   //ATRIBUI PARÂMETROS A CONSULTA
   Acao.ParamByName('p_id_Sitem').AsInteger     := wp_id_SubItem;
   Acao.ParamByName('p_id_item_os').AsInteger   := wp_id_Item;
   Acao.ParamByName('p_id_os').AsInteger        := wp_id_os;
   Acao.ParamByName('p_cobrar').AsInteger       := wp_cobrar;
   Acao.ExecSQL;
   Acao.FreeOnRelease;
end;

procedure TcSubItensOrdemDeServico.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcSubItensOrdemDeServico.Coloca;
begin
   F_id_Sitem   := OColecao.FieldByName('id_Sitem').AsInteger;
   F_id_item_os := OColecao.FieldByName('id_item_os').AsInteger;
   F_id_peca    := OColecao.FieldByName('id_peca').AsInteger;
   F_qtde_peca  := OColecao.FieldByName('qtde_peca').AsInteger;
   F_descricao  := OColecao.FieldByName('descricao').AsString;
   F_cobrar     := OColecao.FieldByName('cobrar').AsInteger;
   F_id_os      := OColecao.FieldByName('id_os').AsInteger;
end;

function TcSubItensOrdemDeServico.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcSubItensOrdemDeServico.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcSubItensOrdemDeServico.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcSubItensOrdemDeServico.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcSubItensOrdemDeServico.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcSubItensOrdemDeServico.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
//Estado = 2 >> Excluir o Subitens especifico.
//Estado = 3 >> Excluir Todos os Subitens do item especifico.
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
```

```
      begin
         ws_Sql := ('Insert into item_os_mov (id_os, id_item_os, descricao, cobrar ');
         if F_id_peca <> 0 then
         begin
            ws_Sql := (ws_Sql +', id_peca, qtde_peca ');
         end;

         ws_Sql := (ws_Sql +') Values (:p_id_os, :p_id_item_os, :p_descricao, 1');

         if F_id_peca <> 0 then
         begin
            ws_Sql := (ws_Sql +', :p_id_peca, :p_qtde_peca');
         end;

         ws_Sql := (ws_Sql +' )');

      end
   else
   if (estado = 1) then
   begin
      ws_Sql := ('Update item_os_mov set ');
      if F_id_peca <> 0 then
      begin
         ws_Sql := (ws_Sql + ' id_peca      = :p_id_peca,    ');
         ws_Sql := (ws_Sql + ' qtde_peca    = :p_qtde_peca,  ');
      end;
      ws_Sql := (ws_Sql + ' descricao       = :p_descricao   ');
      ws_Sql := (ws_Sql + ' where ((id_Sitem = :p_id_Sitem) and (id_os = :p_id_os ) and
(id_item_os = :p_id_item_os))   ');
   end
   else
   if (estado = 2) then
   begin
      ws_Sql := ('delete from item_os_mov where ((id_Sitem = :p_id_Sitem) and (id_os =
:p_id_os ) and (id_item_os = :p_id_item_os))');
   end
   else
   if (estado = 3) then
   begin
      ws_Sql := ('delete from item_os_mov where (id_item_os = :p_id_item_os) and (id_os
= :p_id_os )');
   end;

      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) or (estado = 2) then
      begin
         Acao.ParamByName('p_id_Sitem').AsInteger     := F_id_Sitem;
      end;

      if (estado <> 2) then
      begin
         if (estado <> 3) then
         begin
            if F_id_peca <> 0 then
            begin
               Acao.ParamByName('p_id_peca').AsInteger   := F_id_peca;
               Acao.ParamByName('p_qtde_peca').AsInteger := F_qtde_peca;
            end;
            Acao.ParamByName('p_descricao').AsString     := F_descricao;
         end;
      end;
      Acao.ParamByName('p_id_os').AsInteger        := F_id_os;
      Acao.ParamByName('p_id_item_os').AsInteger   := F_id_item_os;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcSubItensOrdemDeServico.Seleciona(wp_id_os,wp_id_Item: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from item_os_mov where (id_item_os = :p_id_item_os) and (id_os =
:p_id_os )';
   WS_SQL := (WS_SQL + ' order by id_item_os, id_sitem');
```

```
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_item_os').Value    := wp_id_Item;
   Ocolecao.ParamByName('p_id_os').Value         := wp_id_os;
   Ocolecao.Open;
end;

procedure TcSubItensOrdemDeServico.SelecionarPecas(wp_id_os: Integer);
begin
   Ocolecao.Close;
   WS_SQL := 'select * from item_os_mov where (id_os = :p_id_os) and (id_peca <> 0)';
   WS_SQL := (WS_SQL + ' order by id_os,id_item_os, id_sitem');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.ParamByName('p_id_os').Value         := wp_id_os;
   Ocolecao.Open;
end;

procedure TcSubItensOrdemDeServico.Selecionar(wp_id_os,wp_id_Item,wp_id_SubItem:
Integer);
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from item_os_mov where (id_Sitem = :p_id_Sitem) and (id_item_os =
:p_id_item_os) and (id_os = :p_id_os )';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_Sitem').Value  := wp_id_SubItem;
   Acao.ParamByName('p_id_item_os').Value:= wp_id_Item;
   Acao.ParamByName('p_id_os').Value     := wp_id_os;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_Sitem   := 0;
      F_id_item_os := 0;
      F_id_os      := 0;
      F_id_peca    := 0;
      F_qtde_peca  := 0;
      F_cobrar     := 0;
      F_descricao  := '';
   end
   else
   begin
      F_id_Sitem   := Acao.FieldByName('id_Sitem').AsInteger;
      F_id_item_os := Acao.FieldByName('id_item_os').AsInteger;
      F_id_os      := Acao.FieldByName('id_os').AsInteger;
      F_id_peca    := Acao.FieldByName('id_peca').AsInteger;
      F_qtde_peca  := Acao.FieldByName('qtde_peca').AsInteger;
      F_cobrar     := Acao.FieldByName('cobrar').AsInteger;
      F_descricao  := Acao.FieldByName('descricao').AsString;
   end;
   Acao.Free;
end;

procedure TcSubItensOrdemDeServico.Ultimo;
begin
   Ocolecao.Last;
end;

function TcSubItensOrdemDeServico.Valor_total_SItens_itens(wp_id_os,wp_id_Item:
Integer): Double;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := ('select item_os_mov.id_os, item_os_mov.id_item_os, sum(peca.valor_unitario
* item_os_mov.qtde_peca) as valor_total');
   WS_SQL := (ws_sql + ' from  item_os_mov,peca');
   WS_SQL := (ws_sql + ' where (item_os_mov.id_peca = peca.id_peca) and
(item_os_mov.id_os = :p_id_os) and (cobrar = :p_cobrar)');
   WS_SQL := (ws_sql + ' and (item_os_mov.id_item_os = :p_id_item_os)');
   WS_SQL := (ws_sql + ' group by item_os_mov.id_os, item_os_mov.id_item_os order by
item_os_mov.id_os');
   Acao.Close;
   Acao.SQL.clear;
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_os').Value         := wp_id_os;
   Acao.ParamByName('p_id_item_os').Value    := wp_id_Item;
   Acao.ParamByName('p_cobrar').Value        := 1;
```

```
   Acao.Open;
   Result:= Acao.FieldByName('valor_total').AsFloat;
   Acao.Free;
end;

function TcSubItensOrdemDeServico.Valor_total_pecas_os(wp_id_os,wp_cobra: Integer):
Double;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := ('          select item_os_mov.id_os, sum(peca.valor_unitario *
item_os_mov.qtde_peca) as valor_total');
   WS_SQL := (ws_sql + ' from  item_os_mov,peca');
   WS_SQL := (ws_sql + ' where (item_os_mov.id_peca = peca.id_peca) and
(item_os_mov.id_os = :p_id_os) and (cobrar = :p_cobrar)');
   WS_SQL := (ws_sql + ' group by item_os_mov.id_os order by item_os_mov.id_os');
   Acao.Close;
   Acao.SQL.clear;
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_os').Value        := wp_id_os;
   Acao.ParamByName('p_cobrar').Value       := wp_cobra;
   Acao.Open;
   Result:= Acao.FieldByName('valor_total').AsFloat;
   Acao.Free;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.17 unit TTipoDeEquipamento;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TcTipoDeEquipamento = class(TComponent)
    private
      F_id_tipo_equipamento : Integer;
      F_descricao           : String;

    protected
    published
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_tipo_equipamento : Integer read F_id_tipo_equipamento write
F_id_tipo_equipamento;
      property P_descricao           : String  read F_descricao           write
F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_tipo_equipamento: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TcTipoDeEquipamento.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcTipoDeEquipamento.Coloca;
begin
   F_id_tipo_equipamento := OColecao.FieldByName('id_tipo_equipamento').AsInteger;
   F_descricao           := OColecao.FieldByName('descricao').AsString;
end;

function TcTipoDeEquipamento.Comeco: Boolean;
```

```
begin
   Result := OColecao.Bof;
end;

constructor TcTipoDeEquipamento.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcTipoDeEquipamento.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcTipoDeEquipamento.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcTipoDeEquipamento.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcTipoDeEquipamento.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into tipo_equipamento (descricao)');
         ws_Sql := (ws_Sql +' Values (:p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update tipo_equipamento set ');
         ws_Sql := (ws_Sql + ' descricao       = :p_descricao ');
         ws_Sql := (ws_Sql + ' where (id_tipo_equipamento = :p_id_tipo_equipamento)  ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_tipo_equipamento').AsInteger := F_id_tipo_equipamento;
      end;
      Acao.ParamByName('p_descricao').AsString := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcTipoDeEquipamento.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from tipo_equipamento';
   WS_SQL := (WS_SQL + ' order by id_tipo_equipamento, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcTipoDeEquipamento.Selecionar(wp_id_tipo_equipamento: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
```

```
   WS_SQL := 'select * from tipo_equipamento where (id_tipo_equipamento =
:p_id_tipo_equipamento)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_tipo_equipamento').Value  := wp_id_tipo_equipamento;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_tipo_equipamento := 0;
      F_descricao           := '';
   end
   else
   begin
      F_id_tipo_equipamento := Acao.FieldByName('id_tipo_equipamento').AsInteger;
      F_descricao           := Acao.FieldByName('descricao').AsString;
   end;
   Acao.Free;
end;

procedure TcTipoDeEquipamento.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.18 unit TTipoDePeca;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TCTipoDePeca = class(TComponent)
    private
      F_id_tipo_peca : Integer;
      F_descricao    : String;

    protected
    published
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_tipo_peca : Integer read F_id_tipo_peca write F_id_tipo_peca;
      property P_descricao    : String  read F_descricao    write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_tipo_peca: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TTipoDePeca }

procedure TCTipoDePeca.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCTipoDePeca.Coloca;
begin
   F_id_tipo_peca := OColecao.FieldByName('id_tipo_peca').AsInteger;
   F_descricao    := OColecao.FieldByName('descricao').AsString;
end;

function TCTipoDePeca.Comeco: Boolean;
```

```
begin
   Result := OColecao.Bof;
end;

constructor TCTipoDePeca.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCTipoDePeca.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCTipoDePeca.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCTipoDePeca.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCTipoDePeca.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into tipo_peca (descricao)');
         ws_Sql := (ws_Sql +' Values (:p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update tipo_peca set ');
         ws_Sql := (ws_Sql + ' descricao       = :p_descricao ');
         ws_Sql := (ws_Sql + ' where (id_tipo_peca = :p_id_tipo_peca)  ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_tipo_peca').AsInteger := F_id_tipo_peca;
      end;
      Acao.ParamByName('p_descricao').AsString := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCTipoDePeca.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from tipo_peca';
   WS_SQL := (WS_SQL + ' order by id_tipo_peca, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCTipoDePeca.Selecionar(wp_id_tipo_peca: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from tipo_peca where (id_tipo_peca = :p_id_tipo_peca)';
```

```
    Acao.SQL.Add(WS_SQL);
    Acao.ParamByName('p_id_tipo_peca').Value  := wp_id_tipo_peca;
    Acao.Open;
    if acao.IsEmpty then
    begin
       F_id_tipo_peca := 0;
       F_descricao    := '';
    end
    else
    begin
       F_id_tipo_peca := Acao.FieldByName('id_tipo_peca').AsInteger;
       F_descricao    := Acao.FieldByName('descricao').AsString;
    end;
    Acao.Free;
end;

procedure TCTipoDePeca.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.19 unit TTipoDocumento;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TCTipoDocumento = class(TComponent)
    private
      F_id_tipo_doc : Integer;
      F_descricao : String;

    protected
    published
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_tipo_doc : Integer read F_id_tipo_doc write F_id_tipo_doc;
      property P_descricao   : String  read F_descricao   write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_tipo_documento: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

{ TCTipoDocumento }

procedure TCTipoDocumento.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCTipoDocumento.Coloca;
begin
   F_id_tipo_doc := OColecao.FieldByName('id_tipo_doc').AsInteger;
   F_descricao   := OColecao.FieldByName('descricao').AsString;
end;

function TCTipoDocumento.Comeco: Boolean;
```

```
begin
   Result := OColecao.Bof;
end;

constructor TCTipoDocumento.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCTipoDocumento.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCTipoDocumento.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCTipoDocumento.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCTipoDocumento.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into tipo_documento (descricao)');
         ws_Sql := (ws_Sql +' Values (:p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update tipo_documento set ');
         ws_Sql := (ws_Sql + ' descricao       = :p_descricao ');
         ws_Sql := (ws_Sql + ' where (id_tipo_doc = :p_id_tipo_doc)  ');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_tipo_doc').AsInteger := F_id_tipo_doc;
      end;
      Acao.ParamByName('p_descricao').AsString := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCTipoDocumento.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from tipo_documento';
   WS_SQL := (WS_SQL + ' order by id_tipo_doc, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TCTipoDocumento.Selecionar(wp_id_tipo_documento: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from tipo_documento where (id_tipo_doc = :p_id_tipo_doc)';
```

```
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_tipo_doc').Value  := wp_id_tipo_documento;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_tipo_doc := 0;
      F_descricao   := '';
   end
   else
   begin
      F_id_tipo_doc := Acao.FieldByName('id_tipo_doc').AsInteger;
      F_descricao   := Acao.FieldByName('descricao').AsString;
   end;
   Acao.Free;
end;

procedure TCTipoDocumento.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.20 unit TTipoMov;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TCTipoMov = class(TComponent)
    private
      F_id_tipomov : Integer;
      F_tipomov    : Integer;
      F_descricao  : String;

    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_tipomov : Integer read F_id_tipomov write F_id_tipomov;
      property P_tipomov    : Integer read F_tipomov    write F_tipomov;
      property P_descricao  : String  read F_descricao  write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona(wp_tipomov: Integer);
      procedure Selecionar(wp_id_tipo_Movimento: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;

procedure TCTipoMov.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TCTipoMov.Coloca;
begin
   F_id_tipomov := OColecao.FieldByName('id_tipomov').AsInteger;
   F_tipomov    := OColecao.FieldByName('tipomov').AsInteger;
   F_descricao  := OColecao.FieldByName('descricao').AsString;
end;

function TCTipoMov.Comeco: Boolean;
begin
```

```
   Result := OColecao.Bof;
end;

constructor TCTipoMov.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TCTipoMov.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TCTipoMov.Primeiro;
begin
   Ocolecao.First;
end;

procedure TCTipoMov.Proximo;
begin
   Ocolecao.Next;
end;

procedure TCTipoMov.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into tipomov (tipomov, descricao)');
         ws_Sql := (ws_Sql +' Values (:P_tipomov, :p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update tipomov set ');
         ws_Sql := (ws_Sql + ' descricao         = :p_descricao, ');
         ws_Sql := (ws_Sql + ' tipomov           = :P_tipomov    ');
         ws_Sql := (ws_Sql + ' where (id_tipomov = :p_id_tipomov)');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_tipomov').AsInteger := F_id_tipomov;
      end;
      Acao.ParamByName('P_tipomov').AsInteger    := F_tipomov;
      Acao.ParamByName('p_descricao').AsString  := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TCTipoMov.Seleciona(wp_tipomov: Integer);
begin
// 0 = Contas a receber
// 1 = Contas a pagar
// 2 = Todas as Contas
   Ocolecao.Close;
   WS_SQL := 'select * from tipomov ';
   if wp_tipomov <> 2 then
      WS_SQL := (WS_SQL + ' where (tipomov = :p_tipomov) ');
   WS_SQL := (WS_SQL + ' order by id_tipomov, tipomov, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   if wp_tipomov <> 2 then
      Ocolecao.ParamByName('p_tipomov').Value  := wp_tipomov;
   Ocolecao.Open;
end;
```

```
procedure TCTipoMov.Selecionar(wp_id_tipo_Movimento: Integer);
var
   Acao: TPSQLQuery;
begin
   Acao := TPSQLQuery.Create(Self);
   Acao.Database:= dm_base.my_conn;
   WS_SQL := 'select * from tipomov where (id_tipomov = :p_id_tipomov)';
   Acao.SQL.Add(WS_SQL);
   Acao.ParamByName('p_id_tipomov').Value  := wp_id_tipo_Movimento;
   Acao.Open;
   if acao.IsEmpty then
   begin
      F_id_tipomov := 0;
      F_tipomov    := 0;
      F_descricao  := '';
   end
   else
   begin
      F_id_tipomov := Acao.FieldByName('id_tipomov').AsInteger;
      F_tipomov    := Acao.FieldByName('tipomov').AsInteger;
      F_descricao  := Acao.FieldByName('descricao').AsString;
   end;
   Acao.Free;
end;

procedure TCTipoMov.Ultimo;
begin
   Ocolecao.Last;
end;

 end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.1.21 unit TUf;

```
interface

uses
   Classes, PSQLDbTables, SysUtils;

type
  TcUf = class(TComponent)
    private
      F_id_uf : Integer;
      F_abreviada : String;
      F_descricao : String;
    public
      constructor Create(Dono: TComponent);

      //Propriedades
      property P_id_uf     : Integer read F_id_uf     write F_id_uf;
      property P_abreviada : String  read F_abreviada write F_abreviada;
      property P_descricao : String  read F_descricao write F_descricao;

      // operações
      procedure Anterior;
      procedure Coloca;
      function  Comeco: Boolean;
      function  Final: Boolean;
      procedure Primeiro;
      procedure Proximo;
      procedure Salvar(Estado: Integer);
      procedure Seleciona;
      procedure Selecionar(wp_id_uf: Integer);
      procedure Ultimo;

  end;

 var
 Acao,Ocolecao: TPSQLQuery;

implementation

uses uDm_Base;


{ TcUf }

procedure TcUf.Anterior;
begin
   Ocolecao.Prior;
end;

procedure TcUf.Coloca;
begin
   F_id_uf     := OColecao.FieldByName('id_uf').AsInteger;
   F_abreviada := OColecao.FieldByName('abreviada').AsString;
   F_descricao := OColecao.FieldByName('descricao').AsString;
end;
```

```
function TcUf.Comeco: Boolean;
begin
   Result := OColecao.Bof;
end;

constructor TcUf.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TPSQLQuery.Create(Self);
  Ocolecao.Database := dm_base.my_conn;
end;

function TcUf.Final: Boolean;
begin
   Result := OColecao.Eof;
end;

procedure TcUf.Primeiro;
begin
   Ocolecao.First;
end;

procedure TcUf.Proximo;
begin
   Ocolecao.Next;
end;

procedure TcUf.Salvar(Estado: Integer);
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
   try
   Acao := TPSQLQuery.Create(Self);
   Acao.Database := dm_base.my_conn;
   Acao.Close;
   if (estado = 0) then
      begin
         ws_Sql := ('Insert into uf (abreviada, descricao)');
         ws_Sql := (ws_Sql +' Values (:P_abreviada, :p_descricao)');
      end
   else
   if (estado = 1) then
      begin
         ws_Sql := ('Update uf set ');
         ws_Sql := (ws_Sql + ' abreviada    = :p_abreviada, ');
         ws_Sql := (ws_Sql + ' descricao    = :p_descricao  ');
         ws_Sql := (ws_Sql + ' where (id_uf = :p_id_uf)');
      end;
      ws_sql := lowercase(ws_sql);
      Acao.SQL.Add(ws_sql);
      //ATRIBUI PARÂMETROS A CONSULTA
      if (estado = 1) then
      begin
         Acao.ParamByName('p_id_uf').AsInteger := F_id_uf;
      end;
      Acao.ParamByName('P_abreviada').AsString := F_abreviada;
      Acao.ParamByName('p_descricao').AsString  := F_descricao;
      Acao.ExecSQL;
   Finally
     FreeAndNil(Acao);
   End;
end;

procedure TcUf.Seleciona;
begin
   Ocolecao.Close;
   WS_SQL := 'select * from uf';
   WS_SQL := (WS_SQL + ' order by id_uf,abreviada, descricao');
   Ocolecao.SQL.clear;
   Ocolecao.SQL.Add(WS_SQL);
   Ocolecao.Open;
end;

procedure TcUf.Selecionar(wp_id_uf: Integer);
var
   Acao: TPSQLQuery;
begin
```

```
    Acao := TPSQLQuery.Create(Self);
    Acao.Database:= dm_base.my_conn;
    WS_SQL := 'select * from uf where (id_uf = :p_id_uf)';
    Acao.SQL.Add(WS_SQL);
    Acao.ParamByName('p_id_uf').Value  := wp_id_uf;
    Acao.Open;
    if acao.IsEmpty then
    begin
       F_id_uf     := 0;
       F_abreviada := '';
       F_descricao := '';
    end
    else
    begin
       F_id_uf     := Acao.FieldByName('id_uf').AsInteger;
       F_abreviada := Acao.FieldByName('abreviada').AsString;
       F_descricao := Acao.FieldByName('descricao').AsString;
    end;
    Acao.Free;
end;

procedure TcUf.Ultimo;
begin
   Ocolecao.Last;
end;

end.
```

2.2 FONTES DO SISTEMA

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.1 unit uV_OS;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, Mask, ToolEdit, CurrEdit,
  fcButton, fcImgBtn, fcShapeBtn, ExtCtrls, ComCtrls, TOrdemDeServico, TPeca, Tpessoa,
  JvExStdCtrls, JvMemo, JvExExtCtrls, JvComponent, JvCaptionPanel;

type
  Tfrm_V_OS = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    e_descpessoa: TEdit;
    Label2: TLabel;
    e_idpessoa: TCurrencyEdit;
    Panel6: TPanel;
    p_dados: TPanel;
    p_desc: TPanel;
    GroupBox3: TGroupBox;
    Edit_desc_equip: TMemo;
    GroupBox4: TGroupBox;
    Edit_desc_prob: TMemo;
    Panel5: TPanel;
    GroupBox1: TGroupBox;
    lv_Pessoa: TListView;
    p_csitens: TJvCaptionPanel;
    Panel4: TPanel;
    Label6: TLabel;
    Label8: TLabel;
    desc_peca: TEdit;
    GroupBox2: TGroupBox;
    edit_desc_atu: TJvMemo;
    edit_qtdepeca: TCurrencyEdit;
    btn_fechaatu: TfcShapeBtn;
    id_peca: TCurrencyEdit;
    P_Valores: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    GroupBox5: TGroupBox;
    e_numOS: TLabel;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    GroupBox6: TGroupBox;
    Panel7: TPanel;
    lv: TListView;
    Panel8: TPanel;
    btn_alt_sit: TfcShapeBtn;
    Label9: TLabel;
    e_totpecas: TCurrencyEdit;
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure p_dadosEnter(Sender: TObject);
    procedure Panel3Enter(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btn_alt_sitClick(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure p_csitensButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
  private
    procedure monta_lvequip(wp_iddoItem: integer);

    procedure limpar_p_csitens;
    procedure Valor_Orcamento;
    { Private declarations }
  public
    procedure monta_item_os;
    { Public declarations }
  end;

var
  frm_V_OS  : Tfrm_V_OS;
  wi_IDSubitem   : Integer;
  wi_IDitem      : Integer;

implementation

{$R *.dfm}

procedure Tfrm_V_OS.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   frm_V_OS.Release;
   frm_V_OS:= nil;
end;

procedure Tfrm_V_OS.monta_lvequip(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
   wd_v_peca   : double;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   wd_v_peca:= 0;
   pescontr.Items_OS.SubItensOS.Seleciona(strtoint(e_numos.Caption),wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
            wd_v_peca:= peca_sitem.P_valor_unitario *
pescontr.Items_OS.SubItensOS.P_qtde_peca;
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wd_v_peca:= 0;
         end;
         wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);
         if wd_v_peca <> 0 then
            wl_ItemList.SubItems.Add(floattostr(wd_v_peca))
```

```
        else
           wl_ItemList.SubItems.Add('');


        pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_V_OS.p_dadosEnter(Sender: TObject);
begin
  inherited;
    btn_alt_sit.Enabled  := False;
end;

procedure Tfrm_V_OS.Panel3Enter(Sender: TObject);
begin
  inherited;
  p_dados.Enabled:= false;
end;

procedure Tfrm_V_OS.lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;  Shift:
TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado       := lv_Pessoa.GetItemAt(x, y);
  wi_IDitem             := 0;
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
       wi_IDitem            := strtoint(lv_Pessoa.Selected.Caption);
       edit_desc_equip.Text := lv_Pessoa.Selected.SubItems[2];
       edit_desc_prob.Text  := lv_Pessoa.Selected.SubItems[3];
    end
    else
    begin
       edit_desc_equip.Clear;
       edit_desc_prob.Clear;
    end;
  end
  else
  begin
    edit_desc_equip.Clear;
    edit_desc_prob.Clear;
  end;
  monta_lvequip(wi_IDitem);
end;

procedure Tfrm_V_OS.monta_item_os;
var
   os_abrindo  : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   inherited;
   os_abrindo := TcOrdemDeServico.Create(nil);
   os_abrindo.Selecionar(strtoint(e_numos.Caption));
   edit_valorOrc.Value  := os_abrindo.P_valor_orcado;
   e_totpecas.Value     := os_abrindo.P_valor_pecas;
   edit_valorMObra.Value:= os_abrindo.P_valor_mao_obra;
   os_abrindo.Items_OS.Seleciona(strtoint(e_numos.Caption));
   os_abrindo.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      os_abrindo.Items_OS.Coloca;
      if os_abrindo.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(os_abrindo.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_prob);
```

```
        wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_tipo_atendimento));

        os_abrindo.Items_OS.proximo;
      end;
   until os_abrindo.Items_OS.final;
   os_abrindo.FreeOnRelease;
end;

procedure Tfrm_V_OS.lvMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDSubitem         := strtoint(lv.Selected.Caption);
      btn_alt_sit.Enabled  := True;
    end
    else
    begin
       btn_alt_sit.Enabled  := False;
       wi_IDSubitem         := 0;
    end;
  end
  else
  begin
    btn_alt_sit.Enabled  := False;
  end;

end;

procedure Tfrm_V_OS.btn_alt_sitClick(Sender: TObject);
var
   os_SItem     : TcOrdemDeServico;
   peca_sitem   : TcPeca;
begin
   limpar_p_csitens;
   p_csitens.Left:= (frm_V_OS.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top := (frm_V_OS.Height div 2) - (p_csitens.Height div 2);
   Panel2.Enabled:= False;
   os_SItem      := TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);

os_SItem.Items_OS.SubItensOS.Selecionar(strtoint(e_numos.Caption),wi_IDitem,wi_IDSubitem
);
   id_peca.AsInteger          := os_SItem.Items_OS.SubItensOS.P_id_peca;
   edit_qtdepeca.AsInteger    := os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   edit_desc_atu.Text         := os_SItem.Items_OS.SubItensOS.P_descricao;
   peca_sitem.Selecionar(id_peca.AsInteger);
   desc_peca.Text:= peca_sitem.P_descricao;
   peca_sitem.FreeOnRelease;
   os_SItem.FreeOnRelease;
   edit_qtdepeca.enabled := True;
   p_csitens.Visible := True;
end;

procedure Tfrm_V_OS.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;

procedure Tfrm_V_OS.p_csitensButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      p_csitens.Visible:= False;
      Panel2.Enabled   := True;
   end;
end;
```

```
procedure Tfrm_V_OS.limpar_p_csitens;
begin
   desc_peca.Clear;
   edit_qtdepeca.Clear;
   id_peca.Clear;
   edit_desc_atu.Clear;
   edit_qtdepeca.Enabled:= false;
end;

procedure Tfrm_V_OS.Valor_Orcamento;
var
  pescontr        : TcOrdemDeServico;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   edit_valororc.Value  :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1) +
edit_valorMObra.Value;
   e_totpecas.Value     :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1);
   pescontr.freeonrelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.2 unit u_L_V_OS;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa, JvPanel, JvExExtCtrls,
  JvComponent, JvCaptionPanel, tPeca;

type
  Tfrm_L_V_OS = class(Tfrm_ListaPadrao)
    procedure rgp_tipoClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_AlteraClick(Sender: TObject);
  private
    procedure listar(Estado,Tipo: integer);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_L_V_OS: Tfrm_L_V_OS;

implementation

uses Funcoes, uV_OS;

{$R *.dfm}

procedure Tfrm_L_V_OS.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   case rgp_tipo.ItemIndex of
      0 : listar(0,2);  // canceladas
      1 : listar(6,4);  // mostra todas
      2 : listar(0,5);  // orçamentos
      3 : listar(0,0);  // aprovadas
      4 : listar(2,4);  // em andamento
      5 : listar(3,4);  // aguardando peças
      6 : listar(4,4);  // liberadas
      7 : listar(2,3);  // fechadas
      8 : listar(0,6);  // Orçar
      9 : listar(0,7);  // Orçado
      10: listar(0,8);  // Recusada
      11: listar(3,3);  // Devolver
   end;
end;

procedure Tfrm_L_V_OS.listar(Estado,Tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
```

```
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(Estado,Tipo);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(Replacestr(os_atu.P_status,'APROVADA - ',''));
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         if os_atu.P_data_fechou <> 0 then
            wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_fechou))
         else
            wl_ItemList.SubItems.Add('');
         if os_atu.P_data_canc <> 0 then
            wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_canc))
         else
            wl_ItemList.SubItems.Add('');

      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_L_V_OS.FormClose(Sender: TObject;  var Action: TCloseAction);
begin
  inherited;
   frm_L_V_OS.Release;
   frm_L_V_OS:= nil;
end;


procedure Tfrm_L_V_OS.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      LockWindowUpDate(Handle);
      try
         if not assigned(frm_V_OS) then
            frm_V_OS  := Tfrm_V_OS.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;
         frm_V_OS.e_numOS.Caption  := lv.Selected.Caption;
         frm_V_OS.monta_item_os;
         frm_V_OS.e_descpessoa.Text:= lv.Selected.SubItems[0];
         frm_V_OS.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.3 unit uApres;

```
interface

uses
  Windows, Controls, StdCtrls, Classes, ExtCtrls, Forms, Buttons;

type
  Tfrm_Apres = class(TForm)
    Panel1: TPanel;
    Label4: TLabel;
    Label2: TLabel;
    SBPacientes: TSpeedButton;
  private
  public
  end;

var
  frm_Apres: Tfrm_Apres;

implementation

{$R *.dfm}

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.4 unit uCAltera_Orcamento;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, Mask, ToolEdit, CurrEdit,
  fcButton, fcImgBtn, fcShapeBtn, ExtCtrls, ComCtrls, TOrdemDeServico, TPeca, Tpessoa,
  JvExStdCtrls, JvMemo, JvExExtCtrls, JvComponent, JvCaptionPanel, TLogos;

type
  Tfrm_AltOrcamento = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    e_descpessoa: TEdit;
    Label2: TLabel;
    e_idpessoa: TCurrencyEdit;
    Panel6: TPanel;
    p_dados: TPanel;
    p_desc: TPanel;
    GroupBox3: TGroupBox;
    Edit_desc_equip: TMemo;
    GroupBox4: TGroupBox;
    Edit_desc_prob: TMemo;
    E_nos: TCurrencyEdit;
    RG_TAtend: TRadioGroup;
    Panel5: TPanel;
    GroupBox1: TGroupBox;
    Panel9: TPanel;
    btn_Altera: TfcShapeBtn;
    btn_excluir: TfcShapeBtn;
    btn_inc_equip: TfcShapeBtn;
    gb_contrato: TGroupBox;
    label1: TLabel;
    Label3: TLabel;
    e_ncontrato: TCurrencyEdit;
    e_nequipamento: TCurrencyEdit;
    btn_pesqContrato: TfcShapeBtn;
    btn_pesqEquip: TfcShapeBtn;
    Label7: TLabel;
    btn_incluir_item: TfcShapeBtn;
    lv_Pessoa: TListView;
    p_csitens: TJvCaptionPanel;
    Panel4: TPanel;
    Label6: TLabel;
    Label8: TLabel;
    btn_pesq: TfcShapeBtn;
    desc_peca: TEdit;
    GroupBox2: TGroupBox;
    edit_desc_atu: TJvMemo;
    edit_qtdepeca: TCurrencyEdit;
    fcShapeBtn2: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    id_peca: TCurrencyEdit;
    P_Valores: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    GroupBox5: TGroupBox;
```

```
    e_numOS: TLabel;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    GroupBox6: TGroupBox;
    Panel7: TPanel;
    lv: TListView;
    Panel8: TPanel;
    btn_alt_sit: TfcShapeBtn;
    btn_exc_sit: TfcShapeBtn;
    btn_inc_sit: TfcShapeBtn;
    Label9: TLabel;
    e_totpecas: TCurrencyEdit;
    fcShapeBtn1: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_pesqContratoClick(Sender: TObject);
    procedure RG_TAtendClick(Sender: TObject);
    procedure btn_pesqEquipClick(Sender: TObject);
    procedure btn_incluir_itemClick(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure btn_excluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure p_dadosEnter(Sender: TObject);
    procedure Panel3Enter(Sender: TObject);
    procedure btn_inc_equipClick(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btn_alt_sitClick(Sender: TObject);
    procedure btn_exc_sitClick(Sender: TObject);
    procedure btn_inc_sitClick(Sender: TObject);
    procedure fcShapeBtn2Click(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
    procedure p_csitensButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
    procedure E_nosChange(Sender: TObject);
    procedure edit_valorMObraExit(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure btn_FechaClick(Sender: TObject);
  private
    procedure monta_lvequip(wp_iddoItem: integer);
    procedure salva_dados_os;
    procedure limpar_p_csitens;
    procedure Salvar_SItem_OS;
    procedure Valor_Orcamento;
    { Private declarations }
  public
    procedure monta_item_os;
    { Public declarations }
  end;

var
  frm_AltOrcamento      : Tfrm_AltOrcamento;
  wi_IDSubitem : Integer;
  wi_IDitem    : Integer;

implementation

uses uPesqContratos, uPesqEquipamento, uPesqPecas,
  uLLiberaOrcar, uMain;

{$R *.dfm}

procedure Tfrm_AltOrcamento.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   frm_AltOrcamento.Release;
   frm_AltOrcamento:= nil;
end;

procedure Tfrm_AltOrcamento.btn_pesqContratoClick(Sender: TObject);
begin
  inherited;
```

```
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PContrato) then
         frm_PContrato := Tfrm_PContrato.Create(Self);

      Screen.Cursor := crDefault;
      frm_PContrato.show;

   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;

   end;
end;

procedure Tfrm_AltOrcamento.RG_TAtendClick(Sender: TObject);
var
   pescontr    : TCpessoa;
begin
  inherited;
  btn_inc_equip.Enabled:= False;
   if E_nos.Value = 0 then
   begin
      if RG_TAtend.ItemIndex = 1 then
      begin
         pescontr:= TCpessoa.Create(frm_PContrato);
         if pescontr.Contrato.TemRegistros(e_idpessoa.AsInteger) then
         begin
            panel3.Enabled    := false;
            RG_TAtend.enabled := False;
         end
         else
         begin
            Showmessage('Este Cliente não possui contratos Ativos');
            RG_TAtend.ItemIndex  := -1;
            Activecontrol        := RG_TAtend;
            if e_numos.Caption <> '' then
               btn_inc_equip.Visible:= True;
            Exit;
         end;
      end
      else
      begin
         panel3.Enabled    := false;
         RG_TAtend.enabled := False;
      end;
   end
   else
   begin
      if RG_TAtend.ItemIndex = 1 then
      begin
         pescontr:= TCpessoa.Create(frm_PContrato);
         if pescontr.Contrato.TemRegistros(e_idpessoa.AsInteger) then
         begin
            panel3.Enabled    := false;
            RG_TAtend.enabled := False;
         end
         else
         begin
            Showmessage('Este Cliente não possui contratos Ativos');
            RG_TAtend.ItemIndex  := -1;
            Activecontrol        := RG_TAtend;
            btn_inc_equip.Enabled:= tRUE;
            Exit;
         end;
      end;
   end;

   p_dados.Enabled   := True;

   case RG_TAtend.ItemIndex of
      0 :   begin
               gb_contrato.Enabled  := False;
               p_desc.Enabled       := True;
               RG_TAtend.enabled    := False;
```

```
                Edit_desc_equip.SetFocus;
             end;
       1 :   begin
                gb_contrato.Enabled  := True;
                p_desc.Enabled       := False;
                RG_TAtend.enabled    := False;
                btn_pesqContratoclick(self);
             end;
       2 :   begin
                gb_contrato.Enabled  := False;
                p_desc.Enabled       := True;
                RG_TAtend.enabled    := False;
                Edit_desc_equip.SetFocus;
             end;
    end;

end;

procedure Tfrm_AltOrcamento.btn_pesqEquipClick(Sender: TObject);
begin
  inherited;
    Screen.Cursor := crHourGlass;

    LockWindowUpDate(Handle);
    try
       if not assigned(frm_PEquipamentos) then
          frm_PEquipamentos := Tfrm_PEquipamentos.Create(Self);

       Screen.Cursor := crDefault;
       frm_PEquipamentos.show;

    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;

    end;
end;

procedure Tfrm_AltOrcamento.btn_incluir_itemClick(Sender: TObject);
var
    os_abrindo   : TcOrdemDeServico;
    wi_cont      : Integer;
begin
  inherited;
    if Edit_desc_equip.Text = '' then
    begin
       showmessage('A descrição do Equipamento deve ser preenchida');
       Edit_desc_equip.SetFocus;
    end
    else
    begin
       if Edit_desc_prob.Text = '' then
       begin
          showmessage('A descrição do Problema deve ser preenchida');
          Edit_desc_prob.SetFocus;
       end
       else
       begin
          If e_numos.caption = '' then
             btn_salvaclick(self);
          os_abrindo                              := TcOrdemDeServico.Create(nil);
          os_abrindo.Items_OS.P_id_os             := e_nos.asinteger;
          os_abrindo.Items_OS.P_id_contrato       := e_ncontrato.AsInteger;
          os_abrindo.Items_OS.P_id_equipamento    := e_nequipamento.AsInteger;
          os_abrindo.Items_OS.P_tipo_atendimento := RG_TAtend.ItemIndex;
          os_abrindo.Items_OS.P_desc_equip        := Edit_desc_equip.Text;
          os_abrindo.Items_OS.P_desc_prob         := Edit_desc_prob.Text;
          os_abrindo.Items_OS.P_id_item_os        := wi_IDitem;

          if wi_IDitem = 0 then
             os_abrindo.Items_OS.Salvar(0)
          else
             os_abrindo.Items_OS.Salvar(1);
          e_ncontrato.Clear;
          Edit_desc_equip.Clear;
          Edit_desc_prob.Clear;
          e_nequipamento.Clear;
```

```
        RG_TAtend.ItemIndex     := -1;
        os_abrindo.freeonrelease;
        lv.Clear;
        monta_item_os;
        p_desc.Enabled          := False;
        RG_TAtend.Enabled       := False;
        panel5.Enabled          := True;
        btn_inc_equip.Enabled   := TRUE;
     end;
  end;
end;

procedure Tfrm_AltOrcamento.btn_SalvaClick(Sender: TObject);
var
  os_abrindo   : TcOrdemDeServico;
begin
  inherited;
   if e_idpessoa.AsInteger <> 0 then
   begin
      os_abrindo                        := TcOrdemDeServico.Create(nil);
      os_abrindo.P_id_pessoa            := e_idpessoa.AsInteger;
      os_abrindo.P_data_alt             := now;
      os_abrindo.P_valor_orcado         := edit_valororc.AsInteger;
      os_abrindo.P_valor_mao_obra       := edit_valorMobra.AsInteger;
      os_abrindo.p_valor_pecas          := e_totpecas.AsInteger;
      os_abrindo.Salvar(0);
   end;
end;

procedure Tfrm_AltOrcamento.btn_excluirClick(Sender: TObject);
var
  os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este Item!','Exclusão',mb_YesNo)
= IdYes then
   begin
      os_SItem                                  := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := e_nos.AsInteger;
      os_SItem.Items_OS.P_id_item_os            := wi_IDitem;
      os_SItem.Items_OS.P_id_os                 := e_nos.AsInteger;
      //---> exclui o registro
      if wi_IDitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(3);
         os_SItem.Items_OS.Salvar(2);
         lv.Clear;
         monta_item_os;
         monta_lvequip(wi_IDitem);
         btn_inc_equip.Enabled := True;
         btn_inc_sit.Enabled  := False;
         btn_exc_sit.Enabled  := False;
         btn_alt_sit.Enabled  := False;
         btn_Altera.Enabled   := False;
         btn_excluir.Enabled  := False;

      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_AltOrcamento.monta_lvequip(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
   wd_v_peca   : double;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   wd_v_peca:= 0;
   pescontr.Items_OS.SubItensOS.Seleciona(e_nos.AsInteger,wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
```

```
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
            wd_v_peca:= peca_sitem.P_valor_unitario *
pescontr.Items_OS.SubItensOS.P_qtde_peca;
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wd_v_peca:= 0;
         end;
         wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);
         if wd_v_peca <> 0 then
            wl_ItemList.SubItems.Add(floattostr(wd_v_peca))
         else
            wl_ItemList.SubItems.Add('');


         pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_AltOrcamento.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   lv.Clear;
   wi_IDSubitem            := strtoint(lv_Pessoa.ItemFocused.Caption);
   e_ncontrato.Text        := lv_Pessoa.ItemFocused.SubItems[0];
   e_nequipamento.Text     := lv_Pessoa.ItemFocused.SubItems[1];
   Edit_desc_equip.Text    := lv_Pessoa.ItemFocused.SubItems[2];
   Edit_desc_prob.Text     := lv_Pessoa.ItemFocused.SubItems[3];
   RG_TAtend.ItemIndex     := strtoint(lv_Pessoa.ItemFocused.SubItems[4]);
   btn_inc_sit.Enabled     := False;
   btn_exc_sit.Enabled     := False;
   btn_alt_sit.Enabled     := False;
   btn_Altera.Enabled      := False;
   btn_excluir.Enabled     := False;
   p_desc.Enabled          := True;
   lv_Pessoa.ItemFocused.Delete;
end;

procedure Tfrm_AltOrcamento.p_dadosEnter(Sender: TObject);
begin
  inherited;
    btn_inc_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
    btn_alt_sit.Enabled  := False;
end;

procedure Tfrm_AltOrcamento.Panel3Enter(Sender: TObject);
begin
  inherited;
  p_dados.Enabled:= false;
end;

procedure Tfrm_AltOrcamento.btn_inc_equipClick(Sender: TObject);
begin
  inherited;
   lv.Clear;
   RG_TAtend.Enabled    := True;
   btn_inc_sit.Enabled  := False;
   btn_exc_sit.Enabled  := False;
   btn_alt_sit.Enabled  := False;
   btn_Altera.Enabled   := False;
   btn_excluir.Enabled  := False;
```

```
   btn_inc_equip.Enabled := False;
   showmessage('Selecione o Tipo de Atendimento para o novo Equipamento');
   Activecontrol        := RG_TAtend;
end;

procedure Tfrm_AltOrcamento.lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv_Pessoa.GetItemAt(x, y);
  wi_IDitem       := 0;
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
       wi_IDitem           := strtoint(lv_Pessoa.Selected.Caption);
       btn_Altera.Enabled  := True;
       btn_excluir.Enabled := True;
       btn_inc_sit.Enabled := True;
    end
    else
    begin
       btn_Altera.Enabled  := False;
       btn_excluir.Enabled := False;
       btn_inc_sit.Enabled := False;
    end;
  end
  else
  begin
    btn_Altera.Enabled  := False;
    btn_excluir.Enabled := False;
    btn_inc_sit.Enabled := False;
  end;
  monta_lvequip(wi_IDitem);
end;

procedure Tfrm_AltOrcamento.monta_item_os;
var
   os_abrindo  : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   inherited;
   os_abrindo := TcOrdemDeServico.Create(nil);
   os_abrindo.Selecionar(e_nos.asinteger);
   edit_valorOrc.Value  := os_abrindo.P_valor_orcado;
   e_totpecas.Value     := os_abrindo.P_valor_pecas;
   edit_valorMObra.Value:= os_abrindo.P_valor_mao_obra;
   os_abrindo.Items_OS.Seleciona(e_nos.AsInteger);
   os_abrindo.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      os_abrindo.Items_OS.Coloca;
      if os_abrindo.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(os_abrindo.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_prob);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_tipo_atendimento));

         os_abrindo.Items_OS.proximo;
      end;
   until os_abrindo.Items_OS.final;
   os_abrindo.FreeOnRelease;
end;

procedure Tfrm_AltOrcamento.lvMouseUp(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv.GetItemAt(x, y);
  if (itemselecionado <> nil) then
```

```
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDSubitem          := strtoint(lv.Selected.Caption);
      btn_alt_sit.Enabled   := True;
      btn_exc_sit.Enabled   := True;
    end
    else
    begin
       btn_alt_sit.Enabled  := False;
       btn_exc_sit.Enabled  := False;
       wi_IDSubitem         := 0;
    end;
  end
  else
  begin
    btn_alt_sit.Enabled   := False;
    btn_exc_sit.Enabled   := False;
  end;

end;

procedure Tfrm_AltOrcamento.btn_alt_sitClick(Sender: TObject);
var
   os_SItem     : TcOrdemDeServico;
   peca_sitem   : TcPeca;
begin
   limpar_p_csitens;
   p_csitens.Left:= (frm_AltOrcamento.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top := (frm_AltOrcamento.Height div 2) - (p_csitens.Height div 2);
   Panel2.Enabled:= False;
   os_SItem      := TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);

os_SItem.Items_OS.SubItensOS.Selecionar(strtoint(e_numos.Caption),wi_IDitem,wi_IDSubitem
);
   id_peca.AsInteger          := os_SItem.Items_OS.SubItensOS.P_id_peca;
   edit_qtdepeca.AsInteger    := os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   edit_desc_atu.Text         := os_SItem.Items_OS.SubItensOS.P_descricao;
   peca_sitem.Selecionar(id_peca.AsInteger);
   desc_peca.Text:= peca_sitem.P_descricao;
   peca_sitem.FreeOnRelease;
   os_SItem.FreeOnRelease;
   edit_qtdepeca.enabled := True;
   p_csitens.Visible := True;
end;

procedure Tfrm_AltOrcamento.btn_exc_sitClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este
movimento!','Exclusão',mb_YesNo) = IdYes then
   begin
      os_SItem                                := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := E_nos.AsInteger;
      os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
      //---> exclui o registro
      if wi_IDSubitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(2);
         monta_lvequip(wi_IDitem);
         Valor_Orcamento;
         salva_dados_os;
         btn_fechaatuclick(self);
      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_AltOrcamento.btn_inc_sitClick(Sender: TObject);
begin
  inherited;
   limpar_p_csitens;
   p_csitens.Left    := (frm_AltOrcamento.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top     := (frm_AltOrcamento.Height div 2) - (p_csitens.Height div 2);
```

```
   wi_IDSubitem      := 0;
   Panel2.Enabled    := False;
   p_csitens.Visible := TRUE;
end;

procedure Tfrm_AltOrcamento.fcShapeBtn2Click(Sender: TObject);
begin
  inherited;
   Valor_Orcamento;
   if edit_desc_atu.Text <> '' then
   begin
      if id_peca.AsInteger <> 0 then
      begin
         if edit_qtdepeca.AsInteger <> 0 then
         begin
            Salvar_SItem_OS;
            monta_lvequip(wi_IDitem);
            Valor_Orcamento;
            salva_dados_os;
            btn_fechaatuclick(self);
         end
         else
         begin
            Showmessage('Preencha a quantidade de Peças');
         end;
      end
      else
      begin
         Salvar_SItem_OS;
         monta_lvequip(wi_IDitem);
         Valor_Orcamento;
         salva_dados_os;
         btn_fechaatuclick(self);
      end;
   end
   else
      Showmessage('Preencha a Descrição');
   Valor_Orcamento;
end;

procedure Tfrm_AltOrcamento.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;

procedure Tfrm_AltOrcamento.btn_pesqClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqPecas) then
         frm_PesqPecas  := Tfrm_PesqPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;
      frm_PesqPecas.Tag := 4;
      frm_PesqPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_AltOrcamento.p_csitensButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      p_csitens.Visible:= False;
      Panel2.Enabled   := True;
   end;
end;
```

```
procedure Tfrm_AltOrcamento.salva_dados_os;
var
   os_atualiza    : TcOrdemDeServico;
   log            : TcLogos;
begin
   os_atualiza                    := TcOrdemDeServico.Create(nil);
   os_atualiza.Selecionar(strtoint(e_numOS.Caption));
   os_atualiza.P_valor_orcado     := edit_valorOrc.Value;
   os_atualiza.P_valor_pecas      := e_totpecas.Value;
   os_atualiza.P_valor_mao_obra   := edit_valorMObra.Value;
   os_atualiza.Salvar(1);
   os_atualiza.freeonrelease;
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'ALTEROU O ORÇAMENTO';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;

procedure Tfrm_AltOrcamento.limpar_p_csitens;
begin
   desc_peca.Clear;
   edit_qtdepeca.Clear;
   id_peca.Clear;
   edit_desc_atu.Clear;
   edit_qtdepeca.Enabled:= false;
end;

procedure Tfrm_AltOrcamento.Salvar_SItem_OS;
var
   os_SItem   : TcOrdemDeServico;
begin
   os_SItem                                    := TcOrdemDeServico.Create(nil);
   os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
   os_SItem.Items_OS.SubItensOS.P_id_os      := strtoint(e_numOS.Caption);
   os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
   os_SItem.Items_OS.SubItensOS.P_id_peca    := id_peca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_qtde_peca  := edit_qtdepeca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_descricao  := edit_desc_atu.Text;

   //---> salva o registro
   if wi_IDSubitem = 0 then
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(0);
   end
   else
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(1);
   end;
   os_SItem.freeonrelease;
end;

procedure Tfrm_AltOrcamento.E_nosChange(Sender: TObject);
begin
  inherited;
   e_numOS.Caption:= inttostr(E_nos.AsInteger);
end;

procedure Tfrm_AltOrcamento.edit_valorMObraExit(Sender: TObject);
begin
   Valor_Orcamento;
   salva_dados_os;
   Valor_Orcamento;
end;

procedure Tfrm_AltOrcamento.Valor_Orcamento;
var
  pescontr        : TcOrdemDeServico;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   edit_valororc.Value  :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1) +
edit_valorMObra.Value;
   e_totpecas.Value     :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1);
```

```
   pescontr.freeonrelease;
end;

procedure Tfrm_AltOrcamento.fcShapeBtn1Click(Sender: TObject);
begin
  inherited;
   RG_TAtend.Enabled    := True;
   RG_TAtend.ItemIndex  := -1;
end;

procedure Tfrm_AltOrcamento.btn_FechaClick(Sender: TObject);
begin
  inherited;
   frm_LEfetua.rgp_tipo.ItemIndex:= 0;
   frm_LEfetua.rgp_tipoClick(self);
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.5 unit uCCidades;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TCidade, funcoes;

type
  Tfrm_CCidades = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    Label1: TLabel;
    editUF: TComboBox;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
     procedure MontarEstados;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CCidades: Tfrm_CCidades;
  citys       : Tccidade;

implementation

uses uLCidades;

{$R *.dfm}

procedure Tfrm_CCidades.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Estado deve ser preenchido');
   end
   else
      if (Edituf.Text = '') then
      begin
         showmessage('O campo da Sigla da UF deve ser preenchida');
      end
      else
      begin
         citys.P_id_cidade := uLCidades.wi_IDCidade;
         citys.P_id_uf     := StrToInt(copy(editUF.Text,1,4));
         citys.P_descricao := Edit_DGrupo.Text;

         //---> salva o registro
         if uLCidades.wi_IDCidade = 0 then
         begin
```

```
            citys.Salvar(0);
         end
         else
         begin
            citys.Salvar(1);
         end;
         Close;
      end;
end;

procedure Tfrm_CCidades.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   citys.freeonrelease;
   frm_LCidades.monta_lv;
   frm_CCidades.Release;
   frm_CCidades := nil;
end;

procedure Tfrm_CCidades.FormShow(Sender: TObject);
begin
  inherited;
   citys := TcCidade.Create(frm_CCidades);
   citys.Selecionar(uLCidades.wi_IDCidade,uLCidades.wi_IDEstauf);

   if uLCidades.wi_IDCidade = 0 then
      btn_limparclick(self)
   else
   Begin
      Edit_DGrupo.Text := citys.P_descricao;
      citys.UF.Selecionar(citys.P_id_uf);
      Edituf.Text      := ZerosEsq(inttostr(citys.UF.P_id_uf),4) + '-' +
citys.UF.P_abreviada+ '-' + citys.UF.P_descricao;
   end;
   MontarEstados;
   ActiveControl := Edit_DGRUPO;
end;

procedure Tfrm_CCidades.MontarEstados;
var
   UnidFed : TCCidade;
begin
   UnidFed:= TCCidade.Create(frm_CCidades);
   UnidFed.UF.Seleciona;
   UnidFed.UF.Primeiro;
   Edituf.Items.Clear;
   repeat
      UnidFed.UF.Coloca;
      Edituf.Items.Add((ZerosEsq(inttostr(UnidFed.UF.P_id_uf),4) + '-' +
UnidFed.UF.P_abreviada+ '-' + UnidFed.UF.P_descricao));
      UnidFed.UF.proximo;
   until UnidFed.UF.final;
   freeandnil(UnidFed);
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.6 unit uCClientes;

```
interface

uses
  Windows, uTelaCadastroPadrao, StdCtrls, Menus, ToolEdit, CurrEdit, forms,
  Controls, ComCtrls, Mask, fcLabel, fcButton, fcImgBtn, fcShapeBtn,
  Classes, ExtCtrls, TPessoa, SysUtils, Dialogs, funcoes;

type
  Tfrm_CadClientes = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    Pg: TPageControl;
    tab_Contatos_Lst: TTabSheet;
    tab_Obs: TTabSheet;
    Edit_OBS: TMemo;
    menu_Contatos: TPopupMenu;
    ExcluirContato1: TMenuItem;
    Tab_Contatos_Cad: TTabSheet;
    Panel4: TPanel;
    IncluirContato1: TMenuItem;
    AlterarContato1: TMenuItem;
    grid_Contatos: TListView;
    tab_Geral: TTabSheet;
    Panel7: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    L_nome_documento: TLabel;
    L_nomeDoc2: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Edit_Endereco: TEdit;
    Edit_Bairro: TEdit;
    Edit_Cep: TMaskEdit;
    Edit_CNPJ: TMaskEdit;
    Edit_IE: TEdit;
    Edit_UF: TComboBox;
    Edit_Email: TEdit;
    Edit_Homepage: TEdit;
    Panel8: TPanel;
    Edit_idCliente: TCurrencyEdit;
    Label1: TLabel;
    Label27: TLabel;
    Edit_Tipopess: TComboBox;
    Edit_Contato: TEdit;
    Label28: TLabel;
    fcLabel1: TfcLabel;
    Label2: TLabel;
    Edit_RSocial: TEdit;
    gp_telefones: TGroupBox;
    grid_telefones: TListView;
    Label3: TLabel;
    edit_fone: TEdit;
    Label9: TLabel;
```

```
    edit_tfone: TComboBox;
    menu_telefones: TPopupMenu;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    Edit_cidade: TComboBox;
    gp_contatos: TPanel;
    Label16: TLabel;
    Label17: TLabel;
    Label19: TLabel;
    Label23: TLabel;
    Edit_emailContato: TEdit;
    Edit_CargoContato: TEdit;
    Edit_DiaNivContato: TCurrencyEdit;
    Label18: TLabel;
    Edit_MesNivContato: TCurrencyEdit;
    Edit_NomeContato: TEdit;
    Label21: TLabel;
    Edit_SetorContato: TEdit;
    fcShapeBtn8: TfcShapeBtn;
    btn_dados: TfcShapeBtn;
    btn_fon: TfcShapeBtn;
    fcShapeBtn1: TfcShapeBtn;
    fcShapeBtn7: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
    procedure grid_ContatosEnter(Sender: TObject);
    procedure IncluirContato1Click(Sender: TObject);
    procedure AlterarContato1Click(Sender: TObject);
    procedure Edit_DiaNivContatoExit(Sender: TObject);
    procedure Edit_MesNivContatoExit(Sender: TObject);
    procedure tab_GeralEnter(Sender: TObject);
    procedure PgChange(Sender: TObject);
    procedure Edit_TipopessChange(Sender: TObject);
    procedure grid_ContatosMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Edit_CNPJExit(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit_UFChange(Sender: TObject);
    procedure fcShapeBtn7Click(Sender: TObject);
    procedure fcShapeBtn8Click(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure btn_fonClick(Sender: TObject);
    procedure btn_dadosClick(Sender: TObject);
    procedure grid_telefonesMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure MenuItem2Click(Sender: TObject);
    procedure MenuItem3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure SalvaCliente;

    procedure Mostra_Contatos;
    procedure Carrega_Contatos;
    procedure Sair_Contatos;
    procedure mostra_telefones;
    procedure CarregaCampos;
    procedure ChecaCampos;

    procedure MontaCidade(wp_uf: integer);
    procedure MontaUf;

  end;

var
  frm_CadClientes             : Tfrm_CadClientes;
  ws_NomeContato              : String;
  ws_Nfone                    : String;
  wb_entrando_na_tela         : Boolean;
  wb_selecionoulinhaContatos  : boolean;
  wb_selecionoulinhafones     : Boolean;
  wb_flag_cnpj                : boolean;
  Campos_chave                : Boolean;
  Pessoa_Ativa                : TcPessoa;
```

```
implementation

uses uLClientes;

{$R *.dfm}

procedure Tfrm_CadClientes.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   Pessoa_ativa.FreeOnRelease;
   frm_LCliente.FormShow(self);
   frm_CadClientes.Release;
   frm_CadClientes := nil;
end;

procedure Tfrm_CadClientes.btn_SalvaClick(Sender: TObject);
begin
  inherited;

   ChecaCampos;
   if Campos_chave then
   begin
     salvaCliente;
     carregacampos;
     tab_contatos_lst.Visible := True;
     tab_contatos_lst.Enabled := True;
     Pg.ActivePage            := tab_contatos_lst;
   end;


end;

procedure Tfrm_CadClientes.SalvaCliente;
var
   clientecad : TcPessoa;
begin

   clientecad:= TcPessoa.Create(nil);
   clientecad.Selecionar(Edit_idcliente.AsInteger,1);
   clientecad.P_id_pessoa  := Edit_idcliente.AsInteger;
   clientecad.P_nome       := Edit_RSocial.Text;
   clientecad.P_endereco   := Edit_Endereco.Text;
   clientecad.P_bairro     := Edit_Bairro.Text;
   clientecad.P_cep        := Edit_Cep.Text;
   clientecad.P_id_cidade  := StrToInt(copy(Edit_Cidade.Text,1,4));
   clientecad.P_id_UF      := StrToInt(copy(Edit_UF.Text,1,4));
   clientecad.P_tipo_pessoa:= Edit_Tipopess.ItemIndex;
   clientecad.P_cgc_cnpj   := Edit_CNPJ.Text;
   clientecad.P_tipo       := 1;  // Cliente
   clientecad.P_ie_rg      := Edit_IE.Text;
   clientecad.P_contato    := Edit_Contato.Text;
   clientecad.P_email      := Edit_Email.Text;
   clientecad.P_homepage   := Edit_Homepage.Text;
   clientecad.P_obs        := Edit_OBS.Text;


   if Edit_idCliente.AsInteger = 0 then
   begin
      clientecad.Salvar(0,1);
   end
   else
   begin
      clientecad.Salvar(1,1)
   end;
   pessoa_ativa.Selecionar(clientecad.P_id_pessoa,1);
   freeandnil(clientecad);

end;

procedure Tfrm_CadClientes.grid_ContatosEnter(Sender: TObject);
begin
  inherited;
   SalvaCliente;
end;
```

```
procedure Tfrm_CadClientes.IncluirContato1Click(Sender: TObject);
begin
  inherited;
   ws_NomeContato              := '';
   Pessoa_ativa.Contatos.SelecionarUm(Pessoa_ativa.P_id_pessoa,ws_NomeContato);
   Carrega_Contatos;
   mostra_telefones;
   Edit_NomeContato.Enabled    := True;
   Tab_Contatos_Cad.TabVisible := True;
   Pg.ActivePage               := Tab_Contatos_Cad;
   gp_telefones.Enabled        := False;
   gp_contatos.Enabled         := True;
   btn_fon.Enabled             := False;
   btn_dados.Enabled           := False;
   Edit_NomeContato.SetFocus;
end;

procedure Tfrm_CadClientes.Mostra_Contatos;
var
   wl_ItemList : TListItem;
   contpess : TcPessoa;
begin
   contpess:= TcPessoa.Create(nil);
   contpess.Contatos.Seleciona(Pessoa_ativa.P_id_pessoa);
   contpess.Contatos.Primeiro;
   grid_Contatos.Clear;
   repeat
      contpess.Contatos.Coloca;
      if contpess.Contatos.P_id_pessoa <> 0 then
      begin
         wl_ItemList             := grid_Contatos.Items.Add;
         wl_ItemList.Caption     := contpess.Contatos.P_nome;
         wl_ItemList.SubItems.Add(contpess.Contatos.P_dia_nasc + '/'+
contpess.Contatos.P_mes_nasc);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_cargo);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_setor);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_email);
      end;

      contpess.Contatos.proximo;
   until contpess.Contatos.final;
   freeandnil(contpess);
end;

procedure Tfrm_CadClientes.Sair_Contatos;
begin
   Tab_Contatos_Cad.TabVisible := False;
   Mostra_Contatos;
   Pg.ActivePage               := Tab_Contatos_Lst;
   ActiveControl               := grid_Contatos;
end;

procedure Tfrm_CadClientes.AlterarContato1Click(Sender: TObject);
begin
   inherited;
   if wb_selecionoulinhaContatos then
   begin
      ws_NomeContato := trim(grid_Contatos.Selected.Caption);
      Pessoa_ativa.Contatos.SelecionarUm(Pessoa_ativa.P_id_pessoa,ws_NomeContato);

      Tab_Contatos_Cad.TabVisible := True;
      Pg.ActivePage               := Tab_Contatos_Cad;
      gp_telefones.Enabled        := False;
      gp_contatos.Enabled         := True;
      Carrega_Contatos;
      btn_fon.Enabled             := True;
      btn_dados.Enabled           := False;
      Edit_NomeContato.Enabled    := False;
      mostra_telefones;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinhaContatos:= false;
end;

procedure Tfrm_CadClientes.Edit_DiaNivContatoExit(Sender: TObject);
begin
```

```
  inherited;
   if Edit_DiaNivContato.AsInteger > 31 then
   begin
      showmessage('Dia Invalido!!!');
      Edit_DiaNivContato.SetFocus;
      exit;
   end;
end;

procedure Tfrm_CadClientes.Edit_MesNivContatoExit(Sender: TObject);
begin
  inherited;
   if Edit_MesNivContato.AsInteger > 12 then
   begin
      showmessage('Mês Invalido!!!');
      Edit_MesNivContato.SetFocus;
      exit;
   end;
end;

procedure Tfrm_CadClientes.tab_GeralEnter(Sender: TObject);
begin
  inherited;
      Edit_Rsocial.Enabled        := true;
      btn_salva.Visible           := true;
end;

procedure Tfrm_CadClientes.PgChange(Sender: TObject);
begin
   inherited;
      if (pg.ActivePageIndex = 1) or (pg.ActivePageIndex = 2) then
         btn_salva.Visible := false
      else
         btn_salva.Visible := True;
end;

procedure Tfrm_CadClientes.CarregaCampos;
begin
   tab_Contatos_Lst.TabVisible:= False;
   tab_Contatos_cad.TabVisible:= False;
   if Pessoa_ativa.P_id_pessoa <> 0 then
   begin
      Pessoa_ativa.Cidade.Selecionar(Pessoa_ativa.P_id_cidade,Pessoa_ativa.P_id_UF);
      Pessoa_ativa.Cidade.UF.Selecionar(Pessoa_ativa.P_id_UF);
      Edit_idCliente.AsInteger := Pessoa_ativa.P_id_pessoa;
      Edit_RSocial.Text        := Pessoa_ativa.P_nome;
      Edit_Endereco.Text       := Pessoa_ativa.P_endereco;
      Edit_Bairro.Text         := Pessoa_ativa.P_bairro;
      Edit_Cep.Text            := Pessoa_ativa.P_cep;
      Edit_Cidade.Text         := ZerosEsq(inttostr(Pessoa_ativa.Cidade.P_id_cidade),4)
+ '-' + Pessoa_ativa.Cidade.P_descricao;
      Edit_UF.Text             := ZerosEsq(inttostr(Pessoa_ativa.Cidade.UF.P_id_uf),4) +
' - '+ Pessoa_ativa.Cidade.UF.P_abreviada;
      Edit_Email.Text          := Pessoa_ativa.P_email;
      Edit_Homepage.Text       := Pessoa_ativa.P_homepage;
      Edit_OBS.Text            := Pessoa_ativa.P_obs;
      Edit_Tipopess.ItemIndex  := Pessoa_ativa.P_tipo_pessoa;
      Edit_TipopessChange(Self);
      Edit_Contato.Text        := Pessoa_ativa.P_contato;
      Edit_CNPJ.Text           := Pessoa_ativa.P_cgc_cnpj;
      Edit_IE.Text             := Pessoa_ativa.P_ie_rg;
      Mostra_contatos;
      tab_Contatos_Lst.TabVisible:= True;
   end;
   MontaUf;

   Activecontrol            := Edit_RSocial;

end;

procedure Tfrm_CadClientes.Edit_TipopessChange(Sender: TObject);
begin
  if (Edit_Tipopess.ItemIndex = 1) then
  begin
     L_nome_documento.Caption := 'C.N.P.J.';
     Edit_cnpj.Visible        := true;
     Edit_cnpj.EditMask       := '##.###.###/####-##';
```

```
      L_nome_documento.Visible := true;
      L_nomeDoc2.Caption       := 'I.E.';
      L_nomeDoc2.Visible       := true;
      Edit_ie.Visible          := true;
      Edit_cnpj.Text           := '';
      Edit_ie.Text             := '';
      Activecontrol            := Edit_cnpj;
   end
   else
      if (Edit_Tipopess.ItemIndex = 0) then
      begin
         L_nome_documento.Caption := 'C.P.F.';
         Edit_cnpj.Visible        := true;
         Edit_cnpj.EditMask       := '###.###.###-##';
         L_nome_documento.Visible := true;
         L_nomeDoc2.Caption       := 'R.G.';
         L_nomeDoc2.Visible       := true;
         Edit_ie.Visible          := true;
         Edit_cnpj.Text           := '';
         Edit_ie.Text             := '';
         Activecontrol            := Edit_cnpj;
      end
      else
         begin
            L_nome_documento.Caption := '';
            L_nome_documento.Visible := false;
            Edit_cnpj.Visible        := false;
            Edit_cnpj.EditMask       := '';
            L_nomeDoc2.Caption       := '';
            L_nomeDoc2.Visible       := false;
            Edit_ie.Visible          := false;
            Edit_cnpj.Text           := '';
            Edit_ie.Text             := '';
         end;
end;

procedure Tfrm_CadClientes.grid_ContatosMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
wl_Itemcontatos : TListItem;
begin
  inherited;
  wb_selecionoulinhaContatos:=false;
  wl_Itemcontatos := grid_Contatos.GetItemAt(x, y);
  if (wl_Itemcontatos <> nil) then
  begin
    if (wl_Itemcontatos.Selected) then
      wb_selecionoulinhaContatos:= true;
  end
  else
    wb_selecionoulinhaContatos:=false;
end;

procedure Tfrm_CadClientes.Edit_CNPJExit(Sender: TObject);
begin
   inherited;
   wb_flag_cnpj := false;

   if (edit_cnpj.Text = '  .   .   /    -  ') then
            wb_flag_cnpj := false
         else
            if (edit_cnpj.Text = '   .   .   -  ') then
               wb_flag_cnpj := false
            else
               wb_flag_cnpj := true;

   if wb_flag_cnpj then
   begin
      if (Edit_Tipopess.ItemIndex = 1) then
      begin

         if not validacgc(edit_cnpj.Text) then
         begin
            showmessage('CNPJ INVÁLIDO');
            edit_cnpj.SetFocus;
            wb_flag_cnpj := false;
         end
```

```
            else
            begin
               wb_flag_cnpj := true;
            end;
         end
         else
            if (Edit_Tipopess.ItemIndex = 0) then
            begin

            if not validacpf(edit_cnpj.Text) then
            begin
               showmessage('CPF INVÁLIDO');
               edit_cnpj.SetFocus;
               wb_flag_cnpj := false;
            end
            else
            begin
               wb_flag_cnpj := true;
            end;
         end;
      end;

end;

procedure Tfrm_CadClientes.ChecaCampos;
begin

   Pg.ActivePage := tab_Geral;
   Campos_chave:= false;

   if (Edit_Tipopess.ItemIndex = 1) then
   begin
      if wb_flag_cnpj then
      begin
         Campos_chave:= true;
      end
      else
      begin
         showmessage('O campo CNPJ é um campo obrigatório.. Favor preencher');
         Edit_CNPJ.SetFocus;
         Campos_chave:= false;
      end;
   end
   else
      if (Edit_Tipopess.ItemIndex = 0) then
      begin
         if wb_flag_cnpj then
         begin
            Campos_chave:= true;
         end
         else
         begin
            showmessage('O campo CPF é um campo obrigatório.. Favor preencher');
            Edit_CNPJ.SetFocus;
            Campos_chave:= false;
         end;
      end
      else
      begin
         showmessage('O campo Tipo Pessoa é um campo obrigatório.. Selecione um tipo');
         edit_tipopess.SetFocus;
         Campos_chave:= false;
      end;

   if Campos_chave then
   if (edit_RSocial.Text <> '') then
   begin
      Campos_chave:= true;
   end
   else
   begin
      Campos_chave:= false;
      showmessage('O campo Razão Social é um campo obrigatório.. Favor Preencher');
      edit_RSocial.SetFocus;
   end;

   if Campos_chave then
```

```
    if (edit_Uf.ItemIndex >= 0) then
    begin
       Campos_chave:= true;
    end
    else
    begin
       Campos_chave:= false;
       showmessage('O campo Estado (UF) é um campo obrigatório.. Favor Preencher');
       edit_Uf.SetFocus;
    end;

    if Campos_chave then
    if (edit_Cidade.ItemIndex >= 0) then
    begin
       Campos_chave:= true;
    end
    else
    begin
       Campos_chave:= false;
       showmessage('O campo Estado (UF) é um campo obrigatório.. Favor Preencher');
       edit_Cidade.SetFocus;
    end;

end;

procedure Tfrm_CadClientes.FormCreate(Sender: TObject);
begin
  inherited;
   pg.ActivePage := tab_Geral;
   Pessoa_Ativa:= TcPessoa.Create(frm_CadClientes);
   Pessoa_ativa.Selecionar(uLclientes.wi_Cliente,1);
   carregacampos;
end;

procedure Tfrm_CadClientes.MontaCidade(wp_uf: integer);
var
   cidades : TcPessoa;
begin
   cidades:= TcPessoa.Create(nil);
   cidades.Cidade.SelecionacidUf(wp_uf);
   cidades.Cidade.Primeiro;
   Edit_cidade.Items.Clear;
   repeat
      cidades.Cidade.Coloca;
      Edit_cidade.Items.Add((ZerosEsq(inttostr(cidades.Cidade.P_id_cidade),4) + '-' +
cidades.Cidade.P_descricao));
      cidades.Cidade.proximo;
   until cidades.Cidade.final;
   freeandnil(cidades);
end;

procedure Tfrm_CadClientes.MontaUf;
var
   Estadosmont : TcPessoa;
begin
   Estadosmont:= TcPessoa.Create(nil);
   Estadosmont.Cidade.UF.Seleciona;
   Estadosmont.Cidade.UF.Primeiro;
   Edit_uf.Items.Clear;
   repeat
      Estadosmont.Cidade.UF.Coloca;
      Edit_uf.Items.Add((ZerosEsq(inttostr(Estadosmont.Cidade.UF.P_id_uf),4) + ' - '+
Estadosmont.Cidade.UF.P_abreviada));
      Estadosmont.Cidade.UF.proximo;
   until Estadosmont.Cidade.UF.final;
   freeandnil(Estadosmont);
end;

procedure Tfrm_CadClientes.Edit_UFChange(Sender: TObject);
begin
  inherited;
   montaCidade(StrToInt(copy(Edit_UF.Text,1,4)));
   Edit_Cidade.Enabled:= true;
end;

procedure Tfrm_CadClientes.Carrega_Contatos;
begin
```

```
   Edit_NomeContato.Text    := Pessoa_ativa.Contatos.P_nome;
   Edit_DiaNivContato.Text  := Pessoa_ativa.Contatos.P_dia_nasc;
   Edit_MesNivContato.Text  := Pessoa_ativa.Contatos.P_mes_nasc;
   Edit_CargoContato.Text   := Pessoa_ativa.Contatos.P_cargo;
   Edit_SetorContato.Text   := Pessoa_ativa.Contatos.P_setor;
   Edit_emailContato.Text   := Pessoa_ativa.Contatos.P_email;
end;

procedure Tfrm_CadClientes.fcShapeBtn7Click(Sender: TObject);
begin
   if Edit_NomeContato.Text <> '' then
   begin
      Pessoa_ativa.Contatos.P_id_pessoa := edit_idcliente.AsInteger;
      Pessoa_ativa.Contatos.P_nome      := Edit_NomeContato.Text;
      Pessoa_ativa.Contatos.P_dia_nasc  := Edit_DiaNivContato.Text;
      Pessoa_ativa.Contatos.P_mes_nasc  := Edit_MesNivContato.Text;
      Pessoa_ativa.Contatos.P_cargo     := Edit_CargoContato.Text;
      Pessoa_ativa.Contatos.P_setor     := Edit_SetorContato.Text;
      Pessoa_ativa.Contatos.P_email     := Edit_emailContato.Text;
      if ws_NomeContato = '' then
         Pessoa_ativa.Contatos.Salvar(0)
      else
         Pessoa_ativa.Contatos.Salvar(1);
      showmessage('Dados do contato Salvos, Cadastre os telefones');
      gp_telefones.Enabled        := True;
      gp_contatos.Enabled         := False;
      btn_fon.Enabled             := False;
      btn_dados.Enabled           := True;
   end
   else
   begin
      showmessage('Preencha o campo nome');
      activecontrol:= Edit_NomeContato;
      gp_telefones.Enabled        := False;
      gp_contatos.Enabled         := True;
   end;

end;

procedure Tfrm_CadClientes.fcShapeBtn8Click(Sender: TObject);
begin
   if edit_fone.Text = '' then
   begin
      showmessage('Preencha o campo Telefone');
      activecontrol:= edit_fone;
   end
   else
      if edit_tfone.Text = '' then
      begin
         showmessage('Selecione o Tipo de Telefone');
         activecontrol:= edit_tfone;
      end
      else
      begin
         Pessoa_ativa.Contatos.Telefones.P_id_pessoa := Pessoa_ativa.P_id_pessoa;
         Pessoa_ativa.Contatos.Telefones.P_nome      := Pessoa_ativa.Contatos.P_nome;
         Pessoa_ativa.Contatos.Telefones.P_telefone  := Edit_fone.Text;
         Pessoa_ativa.Contatos.Telefones.P_tipo_fone := Edit_tfone.Text;

         if ws_Nfone = '' then
            Pessoa_ativa.Contatos.Telefones.Salvar(0)
         else
            Pessoa_ativa.Contatos.Telefones.Salvar(1);

         showmessage('Dados Salvos');
         Edit_fone.Text := '';
         Edit_tfone.Text:='';
         ws_Nfone       := '';
         mostra_telefones;
      end;

end;

procedure Tfrm_CadClientes.fcShapeBtn1Click(Sender: TObject);
begin
   sair_contatos;
end;
```

```
procedure Tfrm_CadClientes.btn_fonClick(Sender: TObject);
begin
  inherited;
   gp_telefones.Enabled        := True;
   gp_contatos.Enabled         := False;
   btn_fon.Enabled             := False;
   btn_dados.Enabled           := True;
end;

procedure Tfrm_CadClientes.btn_dadosClick(Sender: TObject);
begin
  inherited;
   gp_telefones.Enabled        := False;
   gp_contatos.Enabled         := True;
   btn_fon.Enabled             := True;
   btn_dados.Enabled           := False;
end;

procedure Tfrm_CadClientes.grid_telefonesMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
wl_Itemfones : TListItem;
begin
  inherited;
  wb_selecionoulinhafones:=false;
  wl_Itemfones := grid_telefones.GetItemAt(x, y);
  if (wl_Itemfones <> nil) then
  begin
    if (wl_Itemfones.Selected) then
      wb_selecionoulinhafones:= true;
  end
  else
    wb_selecionoulinhafones:=false;
end;

procedure Tfrm_CadClientes.MenuItem2Click(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinhafones then
   begin
      //---> Altera o telefone
      ws_Nfone := trim(grid_telefones.Selected.Caption);
      Pessoa_ativa.Contatos.Telefones.SelecionarPdes(Pessoa_ativa.P_id_pessoa,ws_Nfone);
      Edit_fone.Text              := Pessoa_ativa.Contatos.Telefones.P_telefone;
      Edit_tfone.Text             := Pessoa_ativa.Contatos.Telefones.P_tipo_fone;
      Edit_NomeContato.Enabled    := False;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinhafones:= false;
end;

procedure Tfrm_CadClientes.MenuItem3Click(Sender: TObject);
begin
  inherited;
  if wb_selecionoulinhafones then
  begin
   if Application.MessageBox('Deseja realmente EXCLUIR este
Telefone???','Exclusão',mb_YesNo) = IdYes then
   begin
      //---> Exclui o telefone
      ws_Nfone := trim(grid_telefones.Selected.Caption);
      Pessoa_ativa.Contatos.Telefones.SelecionarPdes(Pessoa_ativa.P_id_pessoa,ws_Nfone);
      Pessoa_ativa.Contatos.Telefones.Salvar(2);
      Mostra_telefones;
   end;
  end
  else
  showmessage('Selecione um registro');
end;

procedure Tfrm_CadClientes.mostra_telefones;
var
   wl_ItemList : TListItem;
   fonpess : TcPessoa;
begin
```

```
   fonpess:= TcPessoa.Create(nil);

fonpess.Contatos.Telefones.SelecionaTodos(Pessoa_ativa.P_id_pessoa,Pessoa_ativa.Contatos
.P_nome);
   fonpess.Contatos.Telefones.Primeiro;
   grid_telefones.Clear;
   repeat
      fonpess.Contatos.Telefones.Coloca;
      if fonpess.Contatos.Telefones.P_id_fone <> 0 then
      begin
         wl_ItemList            := grid_telefones.Items.Add;
         wl_ItemList.Caption    := fonpess.Contatos.Telefones.P_telefone;
         wl_ItemList.SubItems.Add(fonpess.Contatos.Telefones.P_tipo_fone);
      end;

      fonpess.Contatos.Telefones.proximo;
   until fonpess.Contatos.Telefones.final;
   freeandnil(fonpess);

end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.7 unit uCContratos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TPessoa, funcoes, CurrEdit, Mask, ToolEdit;

type
  Tfrm_CContratos = class(Tfrm_TelaCadastroPadrao)
    GroupBox2: TGroupBox;
    Label6: TLabel;
    edit_data_bloqueio: TDateEdit;
    cb_bloqueio: TCheckBox;
    GroupBox3: TGroupBox;
    Label9: TLabel;
    edit_valormensal: TCurrencyEdit;
    Label10: TLabel;
    edit_dia_vcto: TCurrencyEdit;
    GroupBox4: TGroupBox;
    Label11: TLabel;
    edit_dprorog: TCurrencyEdit;
    edit_data_Fimvig: TDateEdit;
    Label4: TLabel;
    Label3: TLabel;
    edit_data_inivig: TDateEdit;
    Label2: TLabel;
    e_descpessoa: TEdit;
    label1: TLabel;
    e_ncontrato: TCurrencyEdit;
    e_idpessoa: TCurrencyEdit;
    edit_data_cadastro: TDateEdit;
    Label5: TLabel;
    btn_pesq: TfcShapeBtn;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
    procedure cb_bloqueioClick(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CContratos: Tfrm_CContratos;
  ContrPessoa   : TcPessoa;

implementation

uses uLContratos, uPesqCliente;

{$R *.dfm}

procedure Tfrm_CContratos.btn_SalvaClick(Sender: TObject);
```

```
begin
   inherited;
   if (edit_data_Fimvig.Date <> 0) and (edit_data_inivig.Date <> 0) then
   begin
      if (edit_data_Fimvig.Date > edit_data_inivig.Date ) then
      begin
         ContrPessoa.Contrato.P_id_contrato           := e_ncontrato.AsInteger;
         ContrPessoa.Contrato.P_id_pessoa             := e_idpessoa.AsInteger;
         ContrPessoa.Contrato.P_data_inicio_contrato  := edit_data_cadastro.Date;
         ContrPessoa.Contrato.P_data_inicio_vigencia  := edit_data_inivig.Date;
         ContrPessoa.Contrato.P_data_fim_vigencia     := edit_data_Fimvig.Date;
         ContrPessoa.Contrato.P_dias_prorog           := edit_dprorog.AsInteger;
         ContrPessoa.Contrato.P_bloqueado             := cb_bloqueio.Tag;
         ContrPessoa.Contrato.P_data_bloqueio         := edit_data_bloqueio.Date;
         ContrPessoa.Contrato.P_valor_mensal          := edit_valormensal.Value;
         ContrPessoa.Contrato.P_dia_vcto              := edit_dia_vcto.AsInteger;

         //---> salva o registro
         if uLContratos.wi_IDContrato = 0 then
         begin
            ContrPessoa.Contrato.Salvar(0);
         end
         else
         begin
            ContrPessoa.Contrato.Salvar(1);
         end;
         Close;
      end
      else
      begin
         showmessage('A data final do contrato é menor que a data inicial');
      end;
   end
   else
   begin
      showmessage('Preencha os campos de data inicial e final do contrato');
   end;
end;

procedure Tfrm_CContratos.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   ContrPessoa.freeonrelease;
   frm_LContratos.FormShow(self);
   frm_CContratos.Release;
   frm_CContratos := nil;
end;

procedure Tfrm_CContratos.FormShow(Sender: TObject);
begin
  inherited;
   ContrPessoa := TcPessoa.Create(frm_CContratos);
   ContrPessoa.Contrato.Selecionar(uLContratos.wi_IDPessoa,uLContratos.wi_IDContrato);

   if uLContratos.wi_IDContrato = 0 then
   begin
      btn_limparclick(self);
      e_ncontrato.AsInteger      :=   ContrPessoa.Contrato.P_id_contrato;
      e_idpessoa.AsInteger       :=   ContrPessoa.Contrato.P_id_pessoa;
      edit_data_cadastro.Date    :=   ContrPessoa.Contrato.P_data_inicio_contrato;
   end
   else
   Begin
      e_ncontrato.AsInteger      :=   ContrPessoa.Contrato.P_id_contrato;
      e_idpessoa.AsInteger       :=   ContrPessoa.Contrato.P_id_pessoa;
      e_descpessoa.Text          :=   uLContratos.wi_nome;
      edit_data_cadastro.Date    :=   ContrPessoa.Contrato.P_data_inicio_contrato;
      edit_data_inivig.Date      :=   ContrPessoa.Contrato.P_data_inicio_vigencia;
      edit_data_Fimvig.Date      :=   ContrPessoa.Contrato.P_data_fim_vigencia;
      edit_dprorog.AsInteger     :=   ContrPessoa.Contrato.P_dias_prorog;
      cb_bloqueio.Tag            :=   ContrPessoa.Contrato.P_bloqueado;
      if cb_bloqueio.Tag = 1 then
      begin
         cb_bloqueio.Checked := true;
         edit_data_bloqueio.Enabled := true;
      end
      else
```

```
      begin
         cb_bloqueio.Checked := false;
         edit_data_bloqueio.Enabled := False;
      end;
      edit_data_bloqueio.Date    :=   ContrPessoa.Contrato.P_data_bloqueio;
      edit_valormensal.Value     :=   ContrPessoa.Contrato.P_valor_mensal;
      edit_dia_vcto.AsInteger    :=   ContrPessoa.Contrato.P_dia_vcto;
      btn_pesq.Enabled           :=   False;
   end;
end;

procedure Tfrm_CContratos.cb_bloqueioClick(Sender: TObject);
begin
  inherited;
   if cb_bloqueio.Checked then
   begin
      cb_bloqueio.Tag:= 1;
      edit_data_bloqueio.Enabled := true;
   end
   else
   begin
      cb_bloqueio.Tag:= 0;
      edit_data_bloqueio.Enabled := False;
   end;
end;

procedure Tfrm_CContratos.btn_pesqClick(Sender: TObject);
begin
  inherited;
     Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqCliente) then
         frm_PesqCliente := Tfrm_PesqCliente.Create(Self);

      Screen.Cursor := crDefault;
      frm_PesqCliente.Tag := 1;
      frm_PesqCliente.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.8 unit uCEquipamentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, fcButton, fcImgBtn, fcShapeBtn, ExtCtrls,
  StdCtrls, TEquipamento, funcoes;

type
  Tfrm_CEquipamentos = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Label1: TLabel;
    editUF: TComboBox;
    Label2: TLabel;
    Edit_OBS: TMemo;
    Edit_DGrupo: TEdit;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
  private
     procedure montar_tipoequip;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CEquipamentos: Tfrm_CEquipamentos;
  cadEq            : TcEquipamento;

implementation

uses uLEquipamentos;

{$R *.dfm}

procedure Tfrm_CEquipamentos.montar_tipoequip;
var
   tpoequip : TcEquipamento;
begin
   tpoequip:= TcEquipamento.Create(nil);
   tpoequip.Tipo_de_Equipamento.Seleciona;
   tpoequip.Tipo_de_Equipamento.Primeiro;
   Edituf.Items.Clear;
   repeat
      tpoequip.Tipo_de_Equipamento.Coloca;

Edituf.Items.Add((ZerosEsq(inttostr(tpoequip.Tipo_de_Equipamento.P_id_tipo_equipamento),
4) + '-' + tpoequip.Tipo_de_Equipamento.P_descricao));
      tpoequip.Tipo_de_Equipamento.proximo;
   until tpoequip.Tipo_de_Equipamento.final;
   freeandnil(tpoequip);

end;

procedure Tfrm_CEquipamentos.FormShow(Sender: TObject);
```

```
begin
  inherited;
   cadEq := TcEquipamento.Create(frm_CEquipamentos);
   cadEq.Selecionar(uLEquipamentos.wi_IDEquipamento);

   if uLEquipamentos.wi_IDEquipamento = 0 then
      btn_limparclick(self)
   else
   Begin
      cadEq.Tipo_de_Equipamento.Selecionar(cadEq.P_id_tipo_equipamento);
      Edituf.Text      :=
ZerosEsq(inttostr(cadEq.Tipo_de_Equipamento.P_id_tipo_equipamento),4) + '-' +
cadEq.Tipo_de_Equipamento.P_descricao;
      Edit_DGrupo.Text := cadEq.P_numero_serie;
      edit_obs.Text    := cadEq.P_descricao;
   end;
   montar_tipoequip;
   ActiveControl := Edituf;

end;

procedure Tfrm_CEquipamentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   cadEq.freeonrelease;
   frm_LEquipamentos.monta_lvequip(uLEquipamentos.wi_IDContrato);
   frm_CEquipamentos.Release;
   frm_CEquipamentos := nil;
end;

procedure Tfrm_CEquipamentos.btn_SalvaClick(Sender: TObject);
begin
  inherited;
   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Nª de Série deve ser preenchido');
   end
   else
      if (Edituf.Text = '') then
      begin
         showmessage('O campo Tipo de Equipamento deve ser selecionado');
      end
      else
         if (edit_obs.Text = '') then
         begin
            showmessage('O campo Descrição do Equipamento deve ser preenchido');
         end
         else
         begin
            cadEq.P_id_equipamento      := uLEquipamentos.wi_IDEquipamento;
            cadEq.P_id_tipo_equipamento := StrToInt(copy(editUF.Text,1,4));
            cadEq.P_id_contrato         := uLEquipamentos.wi_IDContrato;
            cadEq.P_descricao           := edit_obs.Text;
            cadEq.P_numero_serie        := Edit_DGrupo.Text;

            //---> salva o registro
            if uLEquipamentos.wi_IDEquipamento = 0 then
            begin
               cadEq.Salvar(0);
            end
            else
            begin
               cadEq.Salvar(1);
            end;
            Close;
         end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.9 unit uCEstados;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TUf;

type
  Tfrm_CEstados = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    Label1: TLabel;
    editUF: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CEstados: Tfrm_CEstados;
  ufestado    : TcUf;

implementation

uses uLEstados;

{$R *.dfm}

procedure Tfrm_CEstados.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Estado deve ser preenchido');
   end
   else
      if (Edituf.Text = '') then
      begin
         showmessage('O campo da Sigla da UF deve ser preenchida');
      end
      else
      begin
         ufestado.P_id_uf     := ULEstados.wi_IDEstado;
         ufestado.P_abreviada := editUF.Text;
         ufestado.P_descricao := Edit_DGrupo.Text;

         //---> salva o registro
         if ULEstados.wi_IDEstado = 0 then
         begin
            ufestado.Salvar(0);
```

```
         end
         else
         begin
            ufestado.Salvar(1);
         end;
         Close;
      end;
end;

procedure Tfrm_CEstados.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   ufestado.freeonrelease;
   frm_LEstados.monta_lv;
   frm_CEstados.Release;
   frm_CEstados := nil;
end;

procedure Tfrm_CEstados.FormShow(Sender: TObject);
begin
  inherited;
   ufestado := TcUf.Create(frm_CEstados);
   ufestado.Selecionar(uLEstados.wi_IDEstado);

   if ULEstados.wi_IDEstado = 0 then
      btn_limparclick(self)
   else
   Begin
      Edit_DGrupo.Text := ufestado.P_descricao;
      Edituf.Text      := ufestado.P_abreviada;
   end;

   ActiveControl := Edit_DGRUPO;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.10 unit uCFornecedores;

```
interface

uses
  Windows, uTelaCadastroPadrao, StdCtrls, Menus, ToolEdit, CurrEdit, forms,
  Controls, ComCtrls, Mask, fcLabel, fcButton, fcImgBtn, fcShapeBtn,
  Classes, ExtCtrls, TPessoa, SysUtils, Dialogs, funcoes;

type
  Tfrm_CadFornecedores = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    Pg: TPageControl;
    tab_Contatos_Lst: TTabSheet;
    tab_Obs: TTabSheet;
    Edit_OBS: TMemo;
    menu_Contatos: TPopupMenu;
    ExcluirContato1: TMenuItem;
    Tab_Contatos_Cad: TTabSheet;
    Panel4: TPanel;
    IncluirContato1: TMenuItem;
    AlterarContato1: TMenuItem;
    grid_Contatos: TListView;
    tab_Geral: TTabSheet;
    Panel7: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    L_nome_documento: TLabel;
    L_nomeDoc2: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Edit_Endereco: TEdit;
    Edit_Bairro: TEdit;
    Edit_Cep: TMaskEdit;
    Edit_CNPJ: TMaskEdit;
    Edit_IE: TEdit;
    Edit_UF: TComboBox;
    Edit_Email: TEdit;
    Edit_Homepage: TEdit;
    Panel8: TPanel;
    Edit_idCliente: TCurrencyEdit;
    Label1: TLabel;
    Label27: TLabel;
    Edit_Tipopess: TComboBox;
    Edit_Contato: TEdit;
    Label28: TLabel;
    fcLabel1: TfcLabel;
    Label2: TLabel;
    Edit_RSocial: TEdit;
    gp_telefones: TGroupBox;
    grid_telefones: TListView;
    Label3: TLabel;
    edit_fone: TEdit;
    Label9: TLabel;
```

```
    edit_tfone: TComboBox;
    menu_telefones: TPopupMenu;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    Edit_cidade: TComboBox;
    gp_contatos: TPanel;
    Label16: TLabel;
    Label17: TLabel;
    Label19: TLabel;
    Label23: TLabel;
    Edit_emailContato: TEdit;
    Edit_CargoContato: TEdit;
    Edit_DiaNivContato: TCurrencyEdit;
    Label18: TLabel;
    Edit_MesNivContato: TCurrencyEdit;
    Edit_NomeContato: TEdit;
    Label21: TLabel;
    Edit_SetorContato: TEdit;
    fcShapeBtn8: TfcShapeBtn;
    btn_dados: TfcShapeBtn;
    btn_fon: TfcShapeBtn;
    fcShapeBtn1: TfcShapeBtn;
    fcShapeBtn7: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
    procedure grid_ContatosEnter(Sender: TObject);
    procedure IncluirContato1Click(Sender: TObject);
    procedure AlterarContato1Click(Sender: TObject);
    procedure Edit_DiaNivContatoExit(Sender: TObject);
    procedure Edit_MesNivContatoExit(Sender: TObject);
    procedure tab_GeralEnter(Sender: TObject);
    procedure PgChange(Sender: TObject);
    procedure Edit_TipopessChange(Sender: TObject);
    procedure grid_ContatosMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Edit_CNPJExit(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit_UFChange(Sender: TObject);
    procedure fcShapeBtn7Click(Sender: TObject);
    procedure fcShapeBtn8Click(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure btn_fonClick(Sender: TObject);
    procedure btn_dadosClick(Sender: TObject);
    procedure grid_telefonesMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure MenuItem2Click(Sender: TObject);
    procedure MenuItem3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure SalvaCliente;

    procedure Mostra_Contatos;
    procedure Carrega_Contatos;
    procedure Sair_Contatos;
    procedure mostra_telefones;
    procedure CarregaCampos;
    procedure ChecaCampos;

    procedure MontaCidade(wp_uf: integer);
    procedure MontaUf;

  end;

var
  frm_CadFornecedores             : Tfrm_CadFornecedores;
  ws_NomeContato                : String;
  ws_Nfone                      : String;
  wb_entrando_na_tela           : Boolean;
  wb_selecionoulinhaContatos    : boolean;
  wb_selecionoulinhafones       : Boolean;
  wb_flag_cnpj                  : boolean;
  Campos_chave                  : Boolean;
  Pessoa_Ativa                  : TcPessoa;
```

```
implementation

uses uLFornecedores;

{$R *.dfm}

procedure Tfrm_CadFornecedores.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   Pessoa_ativa.FreeOnRelease;
   frm_LFornecedores.FormShow(self);
   frm_CadFornecedores.Release;
   frm_CadFornecedores := nil;
end;

procedure Tfrm_CadFornecedores.btn_SalvaClick(Sender: TObject);
begin
  inherited;

   ChecaCampos;
   if Campos_chave then
   begin
     salvaCliente;
     carregacampos;
     tab_contatos_lst.Visible := True;
     tab_contatos_lst.Enabled := True;
     Pg.ActivePage            := tab_contatos_lst;
   end;


end;

procedure Tfrm_CadFornecedores.SalvaCliente;
var
   clientecad : TcPessoa;
begin

   clientecad:= TcPessoa.Create(nil);
   clientecad.Selecionar(Edit_idcliente.AsInteger,1);
   clientecad.P_id_pessoa  := Edit_idcliente.AsInteger;
   clientecad.P_nome       := Edit_RSocial.Text;
   clientecad.P_endereco   := Edit_Endereco.Text;
   clientecad.P_bairro     := Edit_Bairro.Text;
   clientecad.P_cep        := Edit_Cep.Text;
   clientecad.P_id_cidade  := StrToInt(copy(Edit_Cidade.Text,1,4));
   clientecad.P_id_UF      := StrToInt(copy(Edit_UF.Text,1,4));
   clientecad.P_tipo_pessoa:= Edit_Tipopess.ItemIndex;
   clientecad.P_cgc_cnpj   := Edit_CNPJ.Text;
   clientecad.P_tipo       := 2;  // Fornecedor
   clientecad.P_ie_rg      := Edit_IE.Text;
   clientecad.P_contato    := Edit_Contato.Text;
   clientecad.P_email      := Edit_Email.Text;
   clientecad.P_homepage   := Edit_Homepage.Text;
   clientecad.P_obs        := Edit_OBS.Text;


   if Edit_idCliente.AsInteger = 0 then
   begin
      clientecad.Salvar(0,2);
   end
   else
   begin
      clientecad.Salvar(1,2)
   end;
   pessoa_ativa.Selecionar(clientecad.P_id_pessoa,2);
   freeandnil(clientecad);

end;

procedure Tfrm_CadFornecedores.grid_ContatosEnter(Sender: TObject);
begin
  inherited;
   SalvaCliente;
end;
```

```
procedure Tfrm_CadFornecedores.IncluirContato1Click(Sender: TObject);
begin
  inherited;
   ws_NomeContato              := '';
   Pessoa_ativa.Contatos.SelecionarUm(Pessoa_ativa.P_id_pessoa,ws_NomeContato);
   Carrega_Contatos;
   mostra_telefones;
   Edit_NomeContato.Enabled    := True;
   Tab_Contatos_Cad.TabVisible := True;
   Pg.ActivePage               := Tab_Contatos_Cad;
   gp_telefones.Enabled        := False;
   gp_contatos.Enabled         := True;
   btn_fon.Enabled             := False;
   btn_dados.Enabled           := False;
   Edit_NomeContato.SetFocus;
end;

procedure Tfrm_CadFornecedores.Mostra_Contatos;
var
   wl_ItemList : TListItem;
   contpess : TcPessoa;
begin
   contpess:= TcPessoa.Create(nil);
   contpess.Contatos.Seleciona(Pessoa_ativa.P_id_pessoa);
   contpess.Contatos.Primeiro;
   grid_Contatos.Clear;
   repeat
      contpess.Contatos.Coloca;
      if contpess.Contatos.P_id_pessoa <> 0 then
      begin
         wl_ItemList            := grid_Contatos.Items.Add;
         wl_ItemList.Caption    := contpess.Contatos.P_nome;
         wl_ItemList.SubItems.Add(contpess.Contatos.P_dia_nasc + '/'+
contpess.Contatos.P_mes_nasc);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_cargo);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_setor);
         wl_ItemList.SubItems.Add(contpess.Contatos.P_email);
      end;

      contpess.Contatos.proximo;
   until contpess.Contatos.final;
   freeandnil(contpess);
end;

procedure Tfrm_CadFornecedores.Sair_Contatos;
begin
   Tab_Contatos_Cad.TabVisible := False;
   Mostra_Contatos;
   Pg.ActivePage               := Tab_Contatos_Lst;
   ActiveControl               := grid_Contatos;
end;

procedure Tfrm_CadFornecedores.AlterarContato1Click(Sender: TObject);
begin
   inherited;
   if wb_selecionoulinhaContatos then
   begin
      ws_NomeContato := trim(grid_Contatos.Selected.Caption);
      Pessoa_ativa.Contatos.SelecionarUm(Pessoa_ativa.P_id_pessoa,ws_NomeContato);

      Tab_Contatos_Cad.TabVisible := True;
      Pg.ActivePage               := Tab_Contatos_Cad;
      gp_telefones.Enabled        := False;
      gp_contatos.Enabled         := True;
      Carrega_Contatos;
      btn_fon.Enabled             := True;
      btn_dados.Enabled           := False;
      Edit_NomeContato.Enabled    := False;
      mostra_telefones;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinhaContatos:= false;
end;

procedure Tfrm_CadFornecedores.Edit_DiaNivContatoExit(Sender: TObject);
begin
```

```
  inherited;
   if Edit_DiaNivContato.AsInteger > 31 then
   begin
      showmessage('Dia Invalido!!!');
      Edit_DiaNivContato.SetFocus;
      exit;
   end;
end;

procedure Tfrm_CadFornecedores.Edit_MesNivContatoExit(Sender: TObject);
begin
  inherited;
   if Edit_MesNivContato.AsInteger > 12 then
   begin
      showmessage('Mês Invalido!!!');
      Edit_MesNivContato.SetFocus;
      exit;
   end;
end;

procedure Tfrm_CadFornecedores.tab_GeralEnter(Sender: TObject);
begin
  inherited;
      Edit_Rsocial.Enabled        := true;
      btn_salva.Visible           := true;
end;

procedure Tfrm_CadFornecedores.PgChange(Sender: TObject);
begin
   inherited;
      if (pg.ActivePageIndex = 1) or (pg.ActivePageIndex = 2) then
         btn_salva.Visible := false
      else
         btn_salva.Visible := True;
end;

procedure Tfrm_CadFornecedores.CarregaCampos;
begin
   tab_Contatos_Lst.TabVisible:= False;
   tab_Contatos_cad.TabVisible:= False;
   if Pessoa_ativa.P_id_pessoa <> 0 then
   begin
      Pessoa_ativa.Cidade.Selecionar(Pessoa_ativa.P_id_cidade,Pessoa_ativa.P_id_UF);
      Pessoa_ativa.Cidade.UF.Selecionar(Pessoa_ativa.P_id_UF);
      Edit_idCliente.AsInteger := Pessoa_ativa.P_id_pessoa;
      Edit_RSocial.Text        := Pessoa_ativa.P_nome;
      Edit_Endereco.Text       := Pessoa_ativa.P_endereco;
      Edit_Bairro.Text         := Pessoa_ativa.P_bairro;
      Edit_Cep.Text            := Pessoa_ativa.P_cep;
      Edit_Cidade.Text         := ZerosEsq(inttostr(Pessoa_ativa.Cidade.P_id_cidade),4)
+ '-' + Pessoa_ativa.Cidade.P_descricao;
      Edit_UF.Text             := ZerosEsq(inttostr(Pessoa_ativa.Cidade.UF.P_id_uf),4) +
' - '+ Pessoa_ativa.Cidade.UF.P_abreviada;
      Edit_Email.Text          := Pessoa_ativa.P_email;
      Edit_Homepage.Text       := Pessoa_ativa.P_homepage;
      Edit_OBS.Text            := Pessoa_ativa.P_obs;
      Edit_Tipopess.ItemIndex  := Pessoa_ativa.P_tipo_pessoa;
      Edit_TipopessChange(Self);
      Edit_Contato.Text        := Pessoa_ativa.P_contato;
      Edit_CNPJ.Text           := Pessoa_ativa.P_cgc_cnpj;
      Edit_IE.Text             := Pessoa_ativa.P_ie_rg;
      Mostra_contatos;
      tab_Contatos_Lst.TabVisible:= True;
   end;
   MontaUf;

   Activecontrol            := Edit_RSocial;

end;

procedure Tfrm_CadFornecedores.Edit_TipopessChange(Sender: TObject);
begin
  if (Edit_Tipopess.ItemIndex = 1) then
  begin
     L_nome_documento.Caption := 'C.N.P.J.';
     Edit_cnpj.Visible        := true;
     Edit_cnpj.EditMask       := '##.###.###/####-##';
```

```
      L_nome_documento.Visible := true;
      L_nomeDoc2.Caption       := 'I.E.';
      L_nomeDoc2.Visible       := true;
      Edit_ie.Visible          := true;
      Edit_cnpj.Text           := '';
      Edit_ie.Text             := '';
      Activecontrol            := Edit_cnpj;
   end
   else
      if (Edit_Tipopess.ItemIndex = 0) then
      begin
         L_nome_documento.Caption := 'C.P.F.';
         Edit_cnpj.Visible        := true;
         Edit_cnpj.EditMask       := '###.###.###-##';
         L_nome_documento.Visible := true;
         L_nomeDoc2.Caption       := 'R.G.';
         L_nomeDoc2.Visible       := true;
         Edit_ie.Visible          := true;
         Edit_cnpj.Text           := '';
         Edit_ie.Text             := '';
         Activecontrol            := Edit_cnpj;
      end
      else
         begin
            L_nome_documento.Caption := '';
            L_nome_documento.Visible := false;
            Edit_cnpj.Visible        := false;
            Edit_cnpj.EditMask       := '';
            L_nomeDoc2.Caption       := '';
            L_nomeDoc2.Visible       := false;
            Edit_ie.Visible          := false;
            Edit_cnpj.Text           := '';
            Edit_ie.Text             := '';
         end;
end;

procedure Tfrm_CadFornecedores.grid_ContatosMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
wl_Itemcontatos : TListItem;
begin
  inherited;
  wb_selecionoulinhaContatos:=false;
  wl_Itemcontatos := grid_Contatos.GetItemAt(x, y);
  if (wl_Itemcontatos <> nil) then
  begin
    if (wl_Itemcontatos.Selected) then
      wb_selecionoulinhaContatos:= true;
  end
  else
    wb_selecionoulinhaContatos:=false;
end;

procedure Tfrm_CadFornecedores.Edit_CNPJExit(Sender: TObject);
begin
   inherited;
   wb_flag_cnpj := false;

   if (edit_cnpj.Text = '  .   .   /    -  ') then
            wb_flag_cnpj := false
         else
            if (edit_cnpj.Text = '   .   .   -  ') then
               wb_flag_cnpj := false
            else
               wb_flag_cnpj := true;

   if wb_flag_cnpj then
   begin
      if (Edit_Tipopess.ItemIndex = 2) then
      begin

         if not validacgc(edit_cnpj.Text) then
         begin
            showmessage('CNPJ INVÁLIDO');
            edit_cnpj.SetFocus;
            wb_flag_cnpj := false;
         end
```

```
            else
            begin
               wb_flag_cnpj := true;
            end;
         end
         else
            if (Edit_Tipopess.ItemIndex = 1) then
            begin

            if not validacpf(edit_cnpj.Text) then
            begin
               showmessage('CPF INVÁLIDO');
               edit_cnpj.SetFocus;
               wb_flag_cnpj := false;
            end
            else
            begin
               wb_flag_cnpj := true;
            end;
         end;
      end;

end;

procedure Tfrm_CadFornecedores.ChecaCampos;
begin

   Pg.ActivePage := tab_Geral;
   Campos_chave:= false;

   if (Edit_Tipopess.ItemIndex = 1) then
   begin
      if wb_flag_cnpj then
      begin
         Campos_chave:= true;
      end
      else
      begin
         showmessage('O campo CNPJ é um campo obrigatório.. Favor preencher');
         Edit_CNPJ.SetFocus;
         Campos_chave:= false;
      end;
   end
   else
      if (Edit_Tipopess.ItemIndex = 0) then
      begin
         if wb_flag_cnpj then
         begin
            Campos_chave:= true;
         end
         else
         begin
            showmessage('O campo CPF é um campo obrigatório.. Favor preencher');
            Edit_CNPJ.SetFocus;
            Campos_chave:= false;
         end;
      end
      else
      begin
         showmessage('O campo Tipo Pessoa é um campo obrigatório.. Selecione um tipo');
         edit_tipopess.SetFocus;
         Campos_chave:= false;
      end;

   if Campos_chave then
   if (edit_RSocial.Text <> '') then
   begin
      Campos_chave:= true;
   end
   else
   begin
      Campos_chave:= false;
      showmessage('O campo Razão Social é um campo obrigatório.. Favor Preencher');
      edit_RSocial.SetFocus;
   end;

   if Campos_chave then
```

```
   if (edit_Uf.ItemIndex >= 0) then
   begin
      Campos_chave:= true;
   end
   else
   begin
      Campos_chave:= false;
      showmessage('O campo Estado (UF) é um campo obrigatório.. Favor Preencher');
      edit_Uf.SetFocus;
   end;

   if Campos_chave then
   if (edit_Cidade.ItemIndex >= 0) then
   begin
      Campos_chave:= true;
   end
   else
   begin
      Campos_chave:= false;
      showmessage('O campo Estado (UF) é um campo obrigatório.. Favor Preencher');
      edit_Cidade.SetFocus;
   end;
end;

procedure Tfrm_CadFornecedores.FormCreate(Sender: TObject);
begin
  inherited;
   pg.ActivePage := tab_Geral;
   Pessoa_Ativa:= TcPessoa.Create(frm_CadFornecedores);
   Pessoa_ativa.Selecionar(uLFornecedores.wi_Cliente,2);
   carregacampos;
end;

procedure Tfrm_CadFornecedores.MontaCidade(wp_uf: integer);
var
   cidades : TcPessoa;
begin
   cidades:= TcPessoa.Create(nil);
   cidades.Cidade.SelecionacidUf(wp_uf);
   cidades.Cidade.Primeiro;
   Edit_cidade.Items.Clear;
   repeat
      cidades.Cidade.Coloca;
      Edit_cidade.Items.Add((ZerosEsq(inttostr(cidades.Cidade.P_id_cidade),4) + '-' +
cidades.Cidade.P_descricao));
      cidades.Cidade.proximo;
   until cidades.Cidade.final;
   freeandnil(cidades);
end;

procedure Tfrm_CadFornecedores.MontaUf;
var
   Estadosmont : TcPessoa;
begin
   Estadosmont:= TcPessoa.Create(nil);
   Estadosmont.Cidade.UF.Seleciona;
   Estadosmont.Cidade.UF.Primeiro;
   Edit_uf.Items.Clear;
   repeat
      Estadosmont.Cidade.UF.Coloca;
      Edit_uf.Items.Add((ZerosEsq(inttostr(Estadosmont.Cidade.UF.P_id_uf),4) + ' - '+
Estadosmont.Cidade.UF.P_abreviada));
      Estadosmont.Cidade.UF.proximo;
   until Estadosmont.Cidade.UF.final;
   freeandnil(Estadosmont);
end;

procedure Tfrm_CadFornecedores.Edit_UFChange(Sender: TObject);
begin
  inherited;
   montaCidade(StrToInt(copy(Edit_UF.Text,1,4)));
   Edit_Cidade.Enabled:= true;
end;

procedure Tfrm_CadFornecedores.Carrega_Contatos;
begin
   Edit_NomeContato.Text    := Pessoa_ativa.Contatos.P_nome;
```

```
   Edit_DiaNivContato.Text  := Pessoa_ativa.Contatos.P_dia_nasc;
   Edit_MesNivContato.Text  := Pessoa_ativa.Contatos.P_mes_nasc;
   Edit_CargoContato.Text   := Pessoa_ativa.Contatos.P_cargo;
   Edit_SetorContato.Text   := Pessoa_ativa.Contatos.P_setor;
   Edit_emailContato.Text   := Pessoa_ativa.Contatos.P_email;
end;

procedure Tfrm_CadFornecedores.fcShapeBtn7Click(Sender: TObject);
begin
   if Edit_NomeContato.Text <> '' then
   begin
      Pessoa_ativa.Contatos.P_id_pessoa := Pessoa_ativa.P_id_pessoa;
      Pessoa_ativa.Contatos.P_nome      := Edit_NomeContato.Text;
      Pessoa_ativa.Contatos.P_dia_nasc  := Edit_DiaNivContato.Text;
      Pessoa_ativa.Contatos.P_mes_nasc  := Edit_MesNivContato.Text;
      Pessoa_ativa.Contatos.P_cargo     := Edit_CargoContato.Text;
      Pessoa_ativa.Contatos.P_setor     := Edit_SetorContato.Text;
      Pessoa_ativa.Contatos.P_email     := Edit_emailContato.Text;
      if ws_NomeContato = '' then
         Pessoa_ativa.Contatos.Salvar(0)
      else
         Pessoa_ativa.Contatos.Salvar(1);
      showmessage('Dados do contato Salvos, Cadastre os telefones');
      gp_telefones.Enabled        := True;
      gp_contatos.Enabled         := False;
      btn_fon.Enabled             := False;
      btn_dados.Enabled           := True;
   end
   else
   begin
      showmessage('Preencha o campo nome');
      activecontrol:= Edit_NomeContato;
      gp_telefones.Enabled        := False;
      gp_contatos.Enabled         := True;
   end;

end;

procedure Tfrm_CadFornecedores.fcShapeBtn8Click(Sender: TObject);
begin
   if edit_fone.Text = '' then
   begin
      showmessage('Preencha o campo Telefone');
      activecontrol:= edit_fone;
   end
   else
      if edit_tfone.Text = '' then
      begin
         showmessage('Selecione o Tipo de Telefone');
         activecontrol:= edit_tfone;
      end
      else
      begin
         Pessoa_ativa.Contatos.Telefones.P_id_pessoa := Pessoa_ativa.P_id_pessoa;
         Pessoa_ativa.Contatos.Telefones.P_nome      := Pessoa_ativa.Contatos.P_nome;
         Pessoa_ativa.Contatos.Telefones.P_telefone  := Edit_fone.Text;
         Pessoa_ativa.Contatos.Telefones.P_tipo_fone := Edit_tfone.Text;

         if ws_Nfone = '' then
            Pessoa_ativa.Contatos.Telefones.Salvar(0)
         else
            Pessoa_ativa.Contatos.Telefones.Salvar(1);

         showmessage('Dados Salvos');
         Edit_fone.Text := '';
         Edit_tfone.Text:='';
         ws_Nfone       := '';
         mostra_telefones;
      end;

end;

procedure Tfrm_CadFornecedores.fcShapeBtn1Click(Sender: TObject);
begin
   sair_contatos;
end;
```

```
procedure Tfrm_CadFornecedores.btn_fonClick(Sender: TObject);
begin
  inherited;
   gp_telefones.Enabled        := True;
   gp_contatos.Enabled         := False;
   btn_fon.Enabled             := False;
   btn_dados.Enabled           := True;
end;

procedure Tfrm_CadFornecedores.btn_dadosClick(Sender: TObject);
begin
  inherited;
   gp_telefones.Enabled        := False;
   gp_contatos.Enabled         := True;
   btn_fon.Enabled             := True;
   btn_dados.Enabled           := False;
end;

procedure Tfrm_CadFornecedores.grid_telefonesMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
wl_Itemfones : TListItem;
begin
  inherited;
  wb_selecionoulinhafones:=false;
  wl_Itemfones := grid_telefones.GetItemAt(x, y);
  if (wl_Itemfones <> nil) then
  begin
    if (wl_Itemfones.Selected) then
      wb_selecionoulinhafones:= true;
  end
  else
    wb_selecionoulinhafones:=false;
end;

procedure Tfrm_CadFornecedores.MenuItem2Click(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinhafones then
   begin
      ws_Nfone := trim(grid_telefones.Selected.Caption);
      Pessoa_ativa.Contatos.Telefones.SelecionarPdes(Pessoa_ativa.P_id_pessoa,ws_Nfone);
      Edit_fone.Text              := Pessoa_ativa.Contatos.Telefones.P_telefone;
      Edit_tfone.Text             := Pessoa_ativa.Contatos.Telefones.P_tipo_fone;
      Edit_NomeContato.Enabled    := False;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinhafones:= false;
end;

procedure Tfrm_CadFornecedores.MenuItem3Click(Sender: TObject);
begin
  inherited;
  if wb_selecionoulinhafones then
  begin
   if Application.MessageBox('Deseja realmente EXCLUIR este
Telefone???','Exclusão',mb_YesNo) = IdYes then
   begin
      //---> Exclui o telefone
      ws_Nfone := trim(grid_telefones.Selected.Caption);
      Pessoa_ativa.Contatos.Telefones.SelecionarPdes(Pessoa_ativa.P_id_pessoa,ws_Nfone);
      Pessoa_ativa.Contatos.Telefones.Salvar(2);
      Mostra_telefones;
   end;
  end
  else
  showmessage('Selecione um registro');
end;

procedure Tfrm_CadFornecedores.mostra_telefones;
var
   wl_ItemList : TListItem;
   fonpess : TcPessoa;
begin
   fonpess:= TcPessoa.Create(nil);
```

```
fonpess.Contatos.Telefones.SelecionaTodos(Pessoa_ativa.P_id_pessoa,Pessoa_ativa.Contatos
.P_nome);
   fonpess.Contatos.Telefones.Primeiro;
   grid_telefones.Clear;
   repeat
      fonpess.Contatos.Telefones.Coloca;
      if fonpess.Contatos.Telefones.P_id_fone <> 0 then
      begin
         wl_ItemList            := grid_telefones.Items.Add;
         wl_ItemList.Caption    := fonpess.Contatos.Telefones.P_telefone;
         wl_ItemList.SubItems.Add(fonpess.Contatos.Telefones.P_tipo_fone);
      end;

      fonpess.Contatos.Telefones.proximo;
   until fonpess.Contatos.Telefones.final;
   freeandnil(fonpess);

end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.11 unit uCGrupoDeUsuarios;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TGrupo;

type
  Tfrm_CGrupo = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CGrupo: Tfrm_CGrupo;
  Grupocad  : TCGrupo;

implementation

uses uLGrupoDeUsuarios;

{$R *.dfm}

procedure Tfrm_CGrupo.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do grupo deve ser preenchido');
   end
   else
   begin
      Grupocad.P_id_grupo       := uLGrupoDeUsuarios.wi_IDGrupo;
      Grupocad.P_descricao      := Edit_DGrupo.Text;

      //---> salva o registro
      if uLGrupoDeUsuarios.wi_IDGrupo = 0 then
      begin
         Grupocad.Salvar(0);
      end
      else
      begin
         Grupocad.Salvar(1);
      end;
      Close;
   end;
end;
```

```
procedure Tfrm_CGrupo.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   Grupocad.freeonrelease;
   frm_LGrupos.monta_lv;
   frm_CGrupo.Release;
   frm_CGrupo := nil;
end;

procedure Tfrm_CGrupo.FormShow(Sender: TObject);
begin
  inherited;
   Grupocad := TCGrupo.Create(frm_CGrupo);
   Grupocad.Selecionar(uLGrupoDeUsuarios.wi_IDGrupo);

   if uLGrupoDeUsuarios.wi_IDGrupo = 0 then
      btn_limparclick(self)
   else
      Edit_DGrupo.Text        := Grupocad.P_descricao;

   ActiveControl := Edit_DGRUPO;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.12 unit uCOrdemServico;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, Mask, ToolEdit, CurrEdit,
  fcButton, fcImgBtn, fcShapeBtn, ExtCtrls, ComCtrls, TOrdemDeServico, TPeca, Tpessoa,
  JvExStdCtrls, JvMemo, JvExExtCtrls, JvComponent, JvCaptionPanel, TLOGOS;

type
  Tfrm_COS = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    btn_pesqcliente: TfcShapeBtn;
    e_descpessoa: TEdit;
    Label2: TLabel;
    e_idpessoa: TCurrencyEdit;
    Panel6: TPanel;
    p_dados: TPanel;
    p_desc: TPanel;
    GroupBox3: TGroupBox;
    Edit_desc_equip: TMemo;
    GroupBox4: TGroupBox;
    Edit_desc_prob: TMemo;
    E_nos: TCurrencyEdit;
    RG_TAtend: TRadioGroup;
    Panel5: TPanel;
    GroupBox1: TGroupBox;
    Panel9: TPanel;
    btn_Altera: TfcShapeBtn;
    btn_excluir: TfcShapeBtn;
    btn_inc_equip: TfcShapeBtn;
    gb_contrato: TGroupBox;
    label1: TLabel;
    Label3: TLabel;
    e_ncontrato: TCurrencyEdit;
    e_nequipamento: TCurrencyEdit;
    btn_pesqContrato: TfcShapeBtn;
    btn_pesqEquip: TfcShapeBtn;
    Label7: TLabel;
    btn_incluir_item: TfcShapeBtn;
    lv_Pessoa: TListView;
    p_csitens: TJvCaptionPanel;
    Panel4: TPanel;
    Label6: TLabel;
    Label8: TLabel;
    btn_pesq: TfcShapeBtn;
    desc_peca: TEdit;
    GroupBox2: TGroupBox;
    edit_desc_atu: TJvMemo;
    edit_qtdepeca: TCurrencyEdit;
    fcShapeBtn2: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    id_peca: TCurrencyEdit;
    P_Valores: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
```

```
    GroupBox5: TGroupBox;
    e_numOS: TLabel;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    GroupBox6: TGroupBox;
    Panel7: TPanel;
    lv: TListView;
    Panel8: TPanel;
    btn_alt_sit: TfcShapeBtn;
    btn_exc_sit: TfcShapeBtn;
    btn_inc_sit: TfcShapeBtn;
    btn_cancelarORC: TfcShapeBtn;
    Label9: TLabel;
    e_totpecas: TCurrencyEdit;
    fcShapeBtn1: TfcShapeBtn;
    procedure btn_pesqclienteClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_pesqContratoClick(Sender: TObject);
    procedure RG_TAtendClick(Sender: TObject);
    procedure btn_pesqEquipClick(Sender: TObject);
    procedure btn_incluir_itemClick(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure btn_excluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure p_dadosEnter(Sender: TObject);
    procedure Panel3Enter(Sender: TObject);
    procedure btn_inc_equipClick(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btn_alt_sitClick(Sender: TObject);
    procedure btn_exc_sitClick(Sender: TObject);
    procedure btn_inc_sitClick(Sender: TObject);
    procedure fcShapeBtn2Click(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
    procedure p_csitensButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
    procedure E_nosChange(Sender: TObject);
    procedure edit_valorMObraExit(Sender: TObject);
    procedure btn_cancelarORCClick(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    procedure monta_lvequip(wp_iddoItem: integer);
    procedure monta_item_os;
    procedure salva_dados_os;
    procedure limpar_p_csitens;
    procedure Salvar_SItem_OS;
    procedure Valor_Orcamento;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_COS      : Tfrm_COS;
  wi_IDSubitem : Integer;
  wi_IDitem    : Integer;

implementation

uses uPesqCliente, uPesqContratos, uPesqEquipamento, uPesqPecas, uMain;

{$R *.dfm}

procedure Tfrm_COS.btn_pesqclienteClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
```

```
      if not assigned(frm_PesqCliente) then
         frm_PesqCliente := Tfrm_PesqCliente.Create(Self);

      Screen.Cursor := crDefault;
      frm_PesqCliente.Tag := 2;
      frm_PesqCliente.show;

   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;

   end;
end;

procedure Tfrm_COS.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   if e_numos.Caption <> '' then
   begin
      Valor_Orcamento;
      salva_dados_os;
   end;
   frm_COS.Release;
   frm_COS:= nil;
end;

procedure Tfrm_COS.btn_pesqContratoClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PContrato) then
         frm_PContrato := Tfrm_PContrato.Create(Self);

      Screen.Cursor := crDefault;
      frm_PContrato.show;

   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;

   end;
end;

procedure Tfrm_COS.RG_TAtendClick(Sender: TObject);
var
   pescontr    : TCpessoa;
begin
  inherited;
  btn_inc_equip.Enabled:= False;
   if E_nos.Value = 0 then
   begin
      if RG_TAtend.ItemIndex = 1 then
      begin
         pescontr:= TCpessoa.Create(frm_PContrato);
         if pescontr.Contrato.TemRegistros(e_idpessoa.AsInteger) then
         begin
            panel3.Enabled    := false;
            RG_TAtend.enabled := False;
            pescontr.FREEONRELEASE;
         end
         else
         begin
            Showmessage('Este Cliente não possui contratos Ativos');
            RG_TAtend.ItemIndex  := -1;
            Activecontrol        := RG_TAtend;
            if e_numos.Caption <> '' then
               btn_inc_equip.Visible:= True;
            pescontr.FREEONRELEASE;
            Exit;
         end;
      end
      else
      begin
         panel3.Enabled    := false;
```

```
            RG_TAtend.enabled := False;
         end;
      end
      else
      begin
         if RG_TAtend.ItemIndex = 1 then
         begin
            pescontr:= TCpessoa.Create(frm_PContrato);
            if pescontr.Contrato.TemRegistros(e_idpessoa.AsInteger) then
            begin
               panel3.Enabled    := false;
               RG_TAtend.enabled := False;
               pescontr.FREEONRELEASE;
            end
            else
            begin
               Showmessage('Este Cliente não possui contratos Ativos');
               RG_TAtend.ItemIndex  := -1;
               Activecontrol        := RG_TAtend;
               btn_inc_equip.Enabled:= tRUE;
               pescontr.FREEONRELEASE;
               Exit;
            end;
         end;
      end;

      frm_COS.p_dados.Enabled   := True;

      case RG_TAtend.ItemIndex of
         0 :   begin
                  gb_contrato.Enabled  := False;
                  p_desc.Enabled       := True;
                  RG_TAtend.enabled    := False;
                  Edit_desc_equip.SetFocus;
               end;
         1 :   begin
                  gb_contrato.Enabled  := True;
                  p_desc.Enabled       := False;
                  RG_TAtend.enabled    := False;
                  btn_pesqContratoclick(self);
               end;
         2 :   begin
                  gb_contrato.Enabled  := False;
                  p_desc.Enabled       := True;
                  RG_TAtend.enabled    := False;
                  Edit_desc_equip.SetFocus;
               end;
      end;
   end;

   procedure Tfrm_COS.btn_pesqEquipClick(Sender: TObject);
   begin
     inherited;
      Screen.Cursor := crHourGlass;

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_PEquipamentos) then
            frm_PEquipamentos := Tfrm_PEquipamentos.Create(Self);

         Screen.Cursor := crDefault;
         frm_PEquipamentos.show;

      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;

      end;
   end;

   procedure Tfrm_COS.btn_incluir_itemClick(Sender: TObject);
   var
      os_abrindo  : TcOrdemDeServico;
      wi_cont     : Integer;
   begin
     inherited;
      if Edit_desc_equip.Text = '' then
```

```
   begin
      showmessage('A descrição do Equipamento deve ser preenchida');
      Edit_desc_equip.SetFocus;
   end
   else
   begin
      if Edit_desc_prob.Text = '' then
      begin
         showmessage('A descrição do Problema deve ser preenchida');
         Edit_desc_prob.SetFocus;
      end
      else
      begin
         If e_numos.caption = '' then
            btn_salvaclick(self);
         os_abrindo                              := TcOrdemDeServico.Create(nil);
         os_abrindo.Items_OS.P_id_os             := e_nos.asinteger;
         os_abrindo.Items_OS.P_id_contrato       := e_ncontrato.AsInteger;
         os_abrindo.Items_OS.P_id_equipamento    := e_nequipamento.AsInteger;
         os_abrindo.Items_OS.P_tipo_atendimento := RG_TAtend.ItemIndex;
         os_abrindo.Items_OS.P_desc_equip        := Edit_desc_equip.Text;
         os_abrindo.Items_OS.P_desc_prob         := Edit_desc_prob.Text;
         os_abrindo.Items_OS.P_id_item_os        := wi_IDitem;

         if wi_IDitem = 0 then
            os_abrindo.Items_OS.Salvar(0)
         else
            os_abrindo.Items_OS.Salvar(1);
         e_ncontrato.Clear;
         Edit_desc_equip.Clear;
         Edit_desc_prob.Clear;
         e_nequipamento.Clear;
         RG_TAtend.ItemIndex     := -1;
         os_abrindo.freeonrelease;
         lv.Clear;
         monta_item_os;
         p_desc.Enabled          := False;
         RG_TAtend.Enabled       := False;
         panel5.Enabled          := True;
         btn_inc_equip.Enabled   := TRUE;
         btn_cancelarORC.Enabled := True;
      end;
   end;
end;

procedure Tfrm_COS.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
   wb_salvou_os : Boolean;
   os_abrindo   : TcOrdemDeServico;
begin
  inherited;
   if e_idpessoa.AsInteger <> 0 then
   begin
      wb_salvou_os := false;
      try
         os_abrindo                          := TcOrdemDeServico.Create(nil);
         os_abrindo.P_id_pessoa              := e_idpessoa.AsInteger;
         os_abrindo.P_status                 := 'ORÇAMENTO';
         os_abrindo.P_data_cad               := now;
         os_abrindo.P_valor_orcado           := 0;
         os_abrindo.P_valor_mao_obra         := 0;
         os_abrindo.p_valor_cobrado          := 0;
         os_abrindo.p_valor_pecas            := 0;
         os_abrindo.p_valor_pecas_ncobrado := 0;
         os_abrindo.p_valor_desconto_mo      := 0;
         os_abrindo.p_valor_desconto_pecas := 0;
         os_abrindo.Salvar(0);

         wb_salvou_os := true;
      except
         wb_salvou_os := false;
         os_abrindo.Salvar(2);
      end;

      if wb_salvou_os then
      begin
```

```
         P_Valores.enabled  := True;
         e_nos.AsInteger := os_abrindo.P_id_os;
         os_abrindo.FreeOnRelease;
      end
      else
         showmessage('OCorreu um erro tente novamente');
   end;

end;

procedure Tfrm_COS.btn_excluirClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este Item!','Exclusão',mb_YesNo)
= IdYes then
   begin
      os_SItem                                    := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := e_nos.AsInteger;
      os_SItem.Items_OS.P_id_item_os            := wi_IDitem;
      os_SItem.Items_OS.P_id_os                 := e_nos.AsInteger;
      //---> exclui o registro
      if wi_IDitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(3);
         os_SItem.Items_OS.Salvar(2);
         lv.Clear;
         monta_item_os;
         monta_lvequip(wi_IDitem);
         btn_inc_equip.Enabled := True;
         btn_inc_sit.Enabled  := False;
         btn_exc_sit.Enabled  := False;
         btn_alt_sit.Enabled  := False;
         btn_Altera.Enabled   := False;
         btn_excluir.Enabled  := False;

      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_COS.monta_lvequip(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
   wd_v_peca   : double;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   wd_v_peca:= 0;
   pescontr.Items_OS.SubItensOS.Seleciona(e_nos.AsInteger,wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
            wd_v_peca:= peca_sitem.P_valor_unitario *
pescontr.Items_OS.SubItensOS.P_qtde_peca;
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wd_v_peca:= 0;
```

```
        end;
        wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);
        if wd_v_peca <> 0 then
           wl_ItemList.SubItems.Add(floattostr(wd_v_peca))
        else
           wl_ItemList.SubItems.Add('');


        pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_COS.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   lv.Clear;
   wi_IDSubitem            := strtoint(lv_Pessoa.ItemFocused.Caption);
   e_ncontrato.Text        := lv_Pessoa.ItemFocused.SubItems[0];
   e_nequipamento.Text     := lv_Pessoa.ItemFocused.SubItems[1];
   Edit_desc_equip.Text    := lv_Pessoa.ItemFocused.SubItems[2];
   Edit_desc_prob.Text     := lv_Pessoa.ItemFocused.SubItems[3];
   RG_TAtend.ItemIndex     := strtoint(lv_Pessoa.ItemFocused.SubItems[4]);
   btn_inc_sit.Enabled     := False;
   btn_exc_sit.Enabled     := False;
   btn_alt_sit.Enabled     := False;
   btn_Altera.Enabled      := False;
   btn_excluir.Enabled     := False;
   p_desc.Enabled          := True;
   lv_Pessoa.ItemFocused.Delete;
end;

procedure Tfrm_COS.p_dadosEnter(Sender: TObject);
begin
  inherited;
    btn_inc_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
    btn_alt_sit.Enabled  := False;
end;

procedure Tfrm_COS.Panel3Enter(Sender: TObject);
begin
  inherited;
  p_dados.Enabled:= false;
end;

procedure Tfrm_COS.btn_inc_equipClick(Sender: TObject);
begin
  inherited;
   wi_IDitem               := 0;
   lv.Clear;
   RG_TAtend.Enabled       := True;
   btn_inc_sit.Enabled     := False;
   btn_exc_sit.Enabled     := False;
   btn_alt_sit.Enabled     := False;
   btn_Altera.Enabled      := False;
   btn_excluir.Enabled     := False;
   btn_inc_equip.Enabled   := False;
   showmessage('Selecione o Tipo de Atendimento para o novo Equipamento');
   Activecontrol           := RG_TAtend;
end;

procedure Tfrm_COS.lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;  Shift:
TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv_Pessoa.GetItemAt(x, y);
  wi_IDitem       := 0;
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
       wi_IDitem           := strtoint(lv_Pessoa.Selected.Caption);
       btn_Altera.Enabled  := True;
```

```
        btn_excluir.Enabled := True;
        btn_inc_sit.Enabled := True;
      end
      else
      begin
        btn_Altera.Enabled  := False;
        btn_excluir.Enabled := False;
        btn_inc_sit.Enabled := False;
      end;
    end
    else
    begin
      btn_Altera.Enabled  := False;
      btn_excluir.Enabled := False;
      btn_inc_sit.Enabled := False;
    end;
    monta_lvequip(wi_IDitem);
end;

procedure Tfrm_COS.monta_item_os;
var
   os_abrindo  : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   inherited;
   os_abrindo := TcOrdemDeServico.Create(nil);
   os_abrindo.Selecionar(e_nos.asinteger);
   edit_valorOrc.Value  := os_abrindo.P_valor_orcado;
   edit_valorMObra.Value:= os_abrindo.P_valor_mao_obra;
   os_abrindo.Items_OS.Seleciona(e_nos.AsInteger);
   os_abrindo.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      os_abrindo.Items_OS.Coloca;
      if os_abrindo.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(os_abrindo.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_prob);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_tipo_atendimento));

         os_abrindo.Items_OS.proximo;
      end;
   until os_abrindo.Items_OS.final;
   os_abrindo.FreeOnRelease;
end;

procedure Tfrm_COS.lvMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDSubitem         := strtoint(lv.Selected.Caption);
      btn_alt_sit.Enabled  := True;
      btn_exc_sit.Enabled  := True;
    end
    else
    begin
       btn_alt_sit.Enabled  := False;
       btn_exc_sit.Enabled  := False;
       wi_IDSubitem         := 0;
    end;
  end
  else
  begin
    btn_alt_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
  end;
```

```
end;

procedure Tfrm_COS.btn_alt_sitClick(Sender: TObject);
var
   os_SItem     : TcOrdemDeServico;
   peca_sitem   : TcPeca;
begin
   limpar_p_csitens;
   p_csitens.Left:= (frm_COS.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top := (frm_COS.Height div 2) - (p_csitens.Height div 2);
   Panel2.Enabled:= False;
   os_SItem      := TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);

os_SItem.Items_OS.SubItensOS.Selecionar(strtoint(e_numos.Caption),wi_IDitem,wi_IDSubitem
);
   id_peca.AsInteger          := os_SItem.Items_OS.SubItensOS.P_id_peca;
   edit_qtdepeca.AsInteger    := os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   edit_desc_atu.Text         := os_SItem.Items_OS.SubItensOS.P_descricao;
   peca_sitem.Selecionar(id_peca.AsInteger);
   desc_peca.Text:= peca_sitem.P_descricao;
   peca_sitem.FreeOnRelease;
   os_SItem.FreeOnRelease;
   edit_qtdepeca.enabled := True;
   p_csitens.Visible := True;
end;

procedure Tfrm_COS.btn_exc_sitClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este
movimento!','Exclusão',mb_YesNo) = IdYes then
   begin
      os_SItem                                := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := E_nos.AsInteger;
      os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
      //---> exclui o registro
      if wi_IDSubitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(2);
         salva_dados_os;
         monta_lvequip(wi_IDitem);
      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_COS.btn_inc_sitClick(Sender: TObject);
begin
  inherited;
   limpar_p_csitens;
   p_csitens.Left    := (frm_COS.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top     := (frm_COS.Height div 2) - (p_csitens.Height div 2);
   wi_IDSubitem      := 0;
   Panel2.Enabled    := False;
   p_csitens.Visible := TRUE;
end;

procedure Tfrm_COS.fcShapeBtn2Click(Sender: TObject);
begin
  inherited;
   if edit_desc_atu.Text <> '' then
   begin
      if id_peca.AsInteger <> 0 then
      begin
         if edit_qtdepeca.AsInteger <> 0 then
         begin
            Salvar_SItem_OS;
            salva_dados_os;
            monta_lvequip(wi_IDitem);
            btn_fechaatuclick(self);
         end
         else
         begin
```

```
            Showmessage('Preencha a quantidade de Peças');
          end;
       end
       else
       begin
          Salvar_SItem_OS;
          salva_dados_os;
          monta_lvequip(wi_IDitem);
          btn_fechaatuclick(self);
       end;
    end
    else
       Showmessage('Preencha a Descrição');
  Valor_Orcamento;
end;
procedure Tfrm_COS.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;
procedure Tfrm_COS.btn_pesqClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqPecas) then
         frm_PesqPecas  := Tfrm_PesqPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;
      frm_PesqPecas.Tag := 1;
      frm_PesqPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;
procedure Tfrm_COS.p_csitensButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      p_csitens.Visible:= False;
      Panel2.Enabled   := True;
   end;
end;
procedure Tfrm_COS.salva_dados_os;
var
   os_atualiza   : TcOrdemDeServico;
   LOG           : TCLOGOS;
begin
   os_atualiza                    := TcOrdemDeServico.Create(nil);
   os_atualiza.Selecionar(strtoint(e_numOS.Caption));
   os_atualiza.P_valor_orcado     := edit_valorOrc.Value;
   os_atualiza.P_valor_mao_obra   := edit_valorMObra.Value;
   os_atualiza.P_valor_pecas      := e_totpecas.Value;
   os_atualiza.Salvar(1);
   os_atualiza.freeonrelease;
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'ABRIU O ORÇAMENTO';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;
procedure Tfrm_COS.limpar_p_csitens;
begin
   desc_peca.Clear;
   edit_qtdepeca.Clear;
   id_peca.Clear;
   edit_desc_atu.Clear;
   edit_qtdepeca.Enabled:= false;
end;
```

```
procedure Tfrm_COS.Salvar_SItem_OS;
var
   os_SItem    : TcOrdemDeServico;
begin
   os_SItem                                  := TcOrdemDeServico.Create(nil);
   os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
   os_SItem.Items_OS.SubItensOS.P_id_os      := strtoint(e_numOS.Caption);
   os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
   os_SItem.Items_OS.SubItensOS.P_id_peca    := id_peca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_qtde_peca  := edit_qtdepeca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_descricao  := edit_desc_atu.Text;

   //---> salva o registro
   if wi_IDSubitem = 0 then
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(0);
   end
   else
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(1);
   end;
   os_SItem.freeonrelease;
end;
procedure Tfrm_COS.E_nosChange(Sender: TObject);
begin
  inherited;
   e_numOS.Caption:= inttostr(E_nos.AsInteger);
end;

procedure Tfrm_COS.edit_valorMObraExit(Sender: TObject);
begin
   salva_dados_os;
   Valor_Orcamento;
end;

procedure Tfrm_COS.Valor_Orcamento;
var
  pescontr        : TcOrdemDeServico;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   edit_valororc.Value  :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1) +
edit_valorMObra.Value;
   e_totpecas.Value     :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1);
   pescontr.freeonrelease;
end;

procedure Tfrm_COS.btn_cancelarORCClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
   os_abrindo   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Cancelar este Orçamento!','A T E N Ç
ÃO',mb_YesNo) = IdYes then
   begin
      os_abrindo                        := TcOrdemDeServico.Create(nil);
      os_abrindo.P_id_os                := strtoint(e_numos.Caption);
      os_abrindo.Salvar(2);
      Close;
   end;
end;

procedure Tfrm_COS.fcShapeBtn1Click(Sender: TObject);
begin
  inherited;
   RG_TAtend.Enabled    := True;
   RG_TAtend.ItemIndex  := -1;
end;

procedure Tfrm_COS.FormCreate(Sender: TObject);
begin
  inherited;
   wi_IDitem:= 0;
   wi_IDSubitem:= 0;
END;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.13 unit uCOS_AprovOrcamento;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn,TOrdemDeServico, TPessoa,TPeca,tlogos;

type
  Tfrm_LAprovaOrcamento = class(Tfrm_ListaPadrao)
    fcShapeBtn1: TfcShapeBtn;
    procedure rgp_tipoClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
  private
    procedure listar(wp_tipo: integer);
    procedure baixar_estoque(wp_os:integer);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_LAprovaOrcamento: Tfrm_LAprovaOrcamento;

implementation

uses uV_OS, uMain;

{$R *.dfm}

procedure Tfrm_LAprovaOrcamento.listar(wp_tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(wp_tipo,7);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         wl_ItemList.SubItems.Add(floattostr(os_atu.P_valor_orcado));
      end;
      os_atu.proximo;
```

```
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_LAprovaOrcamento.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LAprovaOrcamento.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LAprovaOrcamento.Release;
   frm_LAprovaOrcamento:= nil;
end;

procedure Tfrm_LAprovaOrcamento.btn_IncluirClick(Sender: TObject);
var
   os_atualiza    : TcOrdemDeServico;
   log            : Tclogos;
begin
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você deseja APROVAR este
Orçamento!','Aprovação de Orçamento',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'APROVADA';
         os_atualiza.Salvar(1);
         baixar_estoque(strtoint(lv.Selected.Caption));
         os_atualiza.freeonrelease;
         log               := Tclogos.Create(nil);
         log.P_id_os       := strtoint(lv.Selected.Caption);
         log.P_id_usuario  := frm_main.Usuario.P_id_usuario;
         log.P_acao        := 'APROVOU O ORÇAMENTO';
         log.P_data        := now;
         log.Salvar(0);
         log.freeonrelease;
         rgp_tipoclick(self);
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LAprovaOrcamento.btn_AlteraClick(Sender: TObject);
var
   os_atualiza    : TcOrdemDeServico;
   log            : Tclogos;
begin
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você deseja CANCELAR este
Orçamento!','CANCELAMENTO',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'RECUSADA';
         os_atualiza.P_data_canc        := now;
         os_atualiza.Salvar(1);
         os_atualiza.Items_OS.Seleciona(strtoint(lv.Selected.Caption));
         os_atualiza.Items_OS.Primeiro;
         repeat
            os_atualiza.Items_OS.Coloca;
            os_atualiza.Items_OS.SubItensOS.P_id_os     :=
strtoint(lv.Selected.Caption);
            os_atualiza.Items_OS.SubItensOS.P_id_item_os:=
os_atualiza.Items_OS.P_id_item_os;
            os_atualiza.Items_OS.SubItensOS.Salvar(3);
            os_atualiza.Items_OS.proximo;
         until os_atualiza.Items_OS.final;
```

```
         os_atualiza.freeonrelease;
         log              := Tclogos.Create(nil);
         log.P_id_os      := strtoint(lv.Selected.Caption);
         log.P_id_usuario := frm_main.Usuario.P_id_usuario;
         log.P_acao       := 'RECUSOU O ORÇAMENTO';
         log.P_data       := now;
         log.Salvar(0);
         log.freeonrelease;
         rgp_tipoclick(self);
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LAprovaOrcamento.fcShapeBtn1Click(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      LockWindowUpDate(Handle);
      try
         if not assigned(frm_V_OS) then
            frm_V_OS  := Tfrm_V_OS.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;
         frm_V_OS.e_numOS.Caption  := lv.Selected.Caption;
         frm_V_OS.monta_item_os;
         frm_V_OS.e_descpessoa.Text:= lv.Selected.SubItems[0];
         frm_V_OS.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LAprovaOrcamento.baixar_estoque(wp_os:integer);
var
   estoque     : TCPeca;
   OS          : TcOrdemDeServico;
begin
   OS             := TcOrdemDeServico.Create(nil);
   os.Items_OS.SubItensOS.SelecionarPecas(wp_os);
   os.Items_OS.SubItensOS.Primeiro;
   repeat
      os.Items_OS.SubItensOS.Coloca;
      estoque := TCPeca.Create(nil);
      estoque.Selecionar(os.Items_OS.SubItensOS.P_id_peca);
      estoque.P_quantidade:= (estoque.P_quantidade -
os.Items_OS.SubItensOS.P_qtde_peca);
      estoque.Salvar(1);
      estoque.FreeOnRelease;
      os.Items_OS.SubItensOS.proximo;
   until os.Items_OS.SubItensOS.final;
   os.FreeOnRelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.14 unit uCOS_Atualizar;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, fcButton, fcImgBtn, fcShapeBtn, ExtCtrls,
  Mask, ToolEdit, CurrEdit, StdCtrls, TOrdemDeServico, ComCtrls, RXSplit,
  JvExComCtrls, JvListView, JvExStdCtrls, JvMemo, JvPanel, JvExExtCtrls,
  JvComponent, JvCaptionPanel, TPeca, Tlogos;

type
  Tfrm_COS_Atualizar = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    btn_SalvaOS: TfcShapeBtn;
    p_d_OS: TPanel;
    Label2: TLabel;
    e_descpessoa: TEdit;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    RG_TAtend: TRadioGroup;
    GroupBox1: TGroupBox;
    e_numOS: TLabel;
    Panel4: TPanel;
    Panel5: TPanel;
    RxSplitter1: TRxSplitter;
    lv_Pessoa: TListView;
    lv: TListView;
    P_VIEW: TJvCaptionPanel;
    JvPanel1: TJvPanel;
    RxSplitter2: TRxSplitter;
    JvPanel2: TJvPanel;
    GroupBox2: TGroupBox;
    M1: TJvMemo;
    GroupBox3: TGroupBox;
    M2: TJvMemo;
    Panel6: TPanel;
    btn_alt_sit: TfcShapeBtn;
    btn_exc_sit: TfcShapeBtn;
    btn_inc_sit: TfcShapeBtn;
    p_csitens: TJvCaptionPanel;
    Panel7: TPanel;
    Label1: TLabel;
    btn_pesq: TfcShapeBtn;
    desc_peca: TEdit;
    GroupBox4: TGroupBox;
    edit_desc_atu: TJvMemo;
    id_peca: TCurrencyEdit;
    Label3: TLabel;
    edit_qtdepeca: TCurrencyEdit;
    fcShapeBtn1: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure lv_PessoaDblClick(Sender: TObject);
```

```
    procedure P_VIEWButtonClick(Sender: TObject; Button: TJvCapBtnStyle);
    procedure lv_PessoaClick(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure p_csitensButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
    procedure btn_pesqClick(Sender: TObject);
    procedure btn_inc_sitClick(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_alt_sitClick(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btn_exc_sitClick(Sender: TObject);
    procedure btn_SalvaOSClick(Sender: TObject);
    procedure RG_TAtendClick(Sender: TObject);
    procedure edit_valorMObraExit(Sender: TObject);
  private
     procedure salva_dados_os;
     procedure Salvar_SItem_OS;
     procedure monta_lvequip(wp_iddoItem: integer);
     procedure limpar_p_csitens;
    procedure baixar_estoque(wp_peca,wp_qtde:integer);

    { Private declarations }
  public
     procedure Mostra_itens_OS;
    { Public declarations }
  end;

var
  frm_COS_Atualizar: Tfrm_COS_Atualizar;
  wi_id_os         : Integer;
  wi_IDitem        : Integer;
  wi_IDSubitem     : Integer;

implementation

uses uPesqPecas, uLOS_Atualiza, Funcoes, uMain;

{$R *.dfm}

procedure Tfrm_COS_Atualizar.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LOS_Atualiza.rgp_tipoClick(self);
   frm_COS_Atualizar.Release;
   frm_COS_Atualizar:= nil;
end;

procedure Tfrm_COS_Atualizar.salva_dados_os;
var
   os_atualiza   : TcOrdemDeServico;
   ws_status     : String;
   log           : Tclogos;
begin
   os_atualiza                    := TcOrdemDeServico.Create(nil);
   if RG_TAtend.ItemIndex = -1 then
      RG_TAtend.ItemIndex:= 0;
   ws_status:= 'APROVADA - '+Uppercase(RG_TAtend.Items.Strings[RG_TAtend.ItemIndex]);
   ws_status:= ReplaceStr(ws_status,'ç','Ç');
   os_atualiza.Selecionar(strtoint(e_numOS.Caption));
   os_atualiza.P_status           := ws_status;
   os_atualiza.P_data_alt         := now;
   os_atualiza.P_valor_mao_obra   := edit_valorMObra.Value;
   os_atualiza.Salvar(1);
   os_atualiza.freeonrelease;
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'ALTEROU O ESTATUS DA OS PARA '+ws_status;
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;
```

```
procedure Tfrm_COS_Atualizar.Mostra_itens_OS;
var
   pescontr    : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   pescontr:= TcOrdemDeServico.Create(frm_COS_Atualizar);
   pescontr.Selecionar(strtoint(e_numOS.Caption));
   for wi_cont:= 0 to 2 do
      if lowercase(pescontr.P_status) = Lowercase('APROVADA -
'+(RG_TAtend.Items.Strings[wi_cont])) then
         RG_TAtend.ItemIndex:= wi_cont;
   edit_valorOrc.Value:= pescontr.P_valor_orcado;
   edit_valorMObra.Value:= pescontr.P_valor_mao_obra;

   pescontr.Items_OS.Seleciona(strtoint(e_numOs.Caption));
   pescontr.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      pescontr.Items_OS.Coloca;
      if pescontr.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(pescontr.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(pescontr.Items_OS.P_desc_prob);

         pescontr.Items_OS.proximo;
      end;
   until pescontr.Items_OS.final;
   pescontr.freeonrelease;

end;

procedure Tfrm_COS_Atualizar.lv_PessoaDblClick(Sender: TObject);
begin
  inherited;
   M1.Text:= lv_Pessoa.Selected.SubItems[2];
   M2.Text:= lv_Pessoa.Selected.SubItems[3];
   p_view.Left:= (frm_COS_Atualizar.Width div 2) - (p_view.Width div 2);
   p_view.Top := (frm_COS_Atualizar.Height div 2) - (p_view.Height div 2);
   Panel2.Enabled:= False;
   P_VIEW.Visible:= TRUE;
end;

procedure Tfrm_COS_Atualizar.P_VIEWButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      P_VIEW.Visible   := False;
      Panel2.Enabled   := True;
   end;
end;

procedure Tfrm_COS_Atualizar.lv_PessoaClick(Sender: TObject);
begin
  inherited;
   btn_exc_sit.Enabled  := False;
   btn_alt_sit.Enabled  := False;
end;

procedure Tfrm_COS_Atualizar.lv_PessoaMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv_Pessoa.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDitem            := strtoint(lv_pessoa.Selected.Caption);
      btn_inc_sit.Enabled  := True;
```

```
      monta_lvequip(wi_IDitem);
    end
    else
    begin
       btn_inc_sit.Enabled  := False;
       wi_IDitem            := 0;
    end;
  end
  else
    btn_inc_sit.Enabled  := False;

end;

procedure Tfrm_COS_Atualizar.monta_lvequip(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   pescontr.Items_OS.SubItensOS.Seleciona(wi_id_os,wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
         end;
         wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);

         pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_COS_Atualizar.p_csitensButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      p_csitens.Visible:= False;
      Panel2.Enabled   := True;
   end;
end;

procedure Tfrm_COS_Atualizar.btn_pesqClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqPecas) then
         frm_PesqPecas  := Tfrm_PesqPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;
      frm_PesqPecas.Tag := 2;
      frm_PesqPecas.show;
```

```
    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;
    end;
end;

procedure Tfrm_COS_Atualizar.btn_inc_sitClick(Sender: TObject);
begin
  inherited;
   limpar_p_csitens;
   p_csitens.Left    := (frm_COS_Atualizar.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top     := (frm_COS_Atualizar.Height div 2) - (p_csitens.Height div 2);
   wi_IDSubitem      := 0;
   Panel2.Enabled    := False;
   btn_pesq.Enabled  := True;
   p_csitens.Visible := TRUE;
end;

procedure Tfrm_COS_Atualizar.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;

procedure Tfrm_COS_Atualizar.limpar_p_csitens;
begin
   desc_peca.Clear;
   edit_qtdepeca.Clear;
   id_peca.Clear;
   edit_desc_atu.Clear;
   edit_qtdepeca.Enabled:= false;
end;

procedure Tfrm_COS_Atualizar.btn_alt_sitClick(Sender: TObject);
var
   os_SItem     : TcOrdemDeServico;
   peca_sitem   : TcPeca;
begin
   limpar_p_csitens;
   p_csitens.Left:= (frm_COS_Atualizar.Width div 2) - (p_view.Width div 2);
   p_csitens.Top := (frm_COS_Atualizar.Height div 2) - (p_view.Height div 2);
   Panel2.Enabled:= False;
   os_SItem      := TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   os_SItem.Items_OS.SubItensOS.Selecionar(wi_id_os,wi_IDitem,wi_IDSubitem);
   id_peca.AsInteger          := os_SItem.Items_OS.SubItensOS.P_id_peca;
   edit_qtdepeca.AsInteger    := os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   edit_desc_atu.Text         := os_SItem.Items_OS.SubItensOS.P_descricao;
   peca_sitem.Selecionar(id_peca.AsInteger);
   desc_peca.Text:= peca_sitem.P_descricao;
   peca_sitem.FreeOnRelease;
   os_SItem.FreeOnRelease;
   edit_qtdepeca.enabled := True;
   btn_pesq.Enabled      := False;
   p_csitens.Visible := True;
end;

procedure Tfrm_COS_Atualizar.fcShapeBtn1Click(Sender: TObject);
begin
  inherited;
   if edit_desc_atu.Text <> '' then
   begin
      if id_peca.AsInteger <> 0 then
      begin
         if edit_qtdepeca.AsInteger <> 0 then
         begin
            Salvar_SItem_OS;
            if RG_TAtend.ItemIndex < 0 then
            begin
               RG_TAtend.ItemIndex:= 0;
            end;
            salva_dados_os;
            monta_lvequip(wi_IDitem);
            btn_fechaatuclick(self);
         end
         else
```

```
         begin
            Showmessage('Preencha a quantidade de Peças');
         end;
      end
      else
      begin
         Salvar_SItem_OS;
         if RG_TAtend.ItemIndex < 0 then
         begin
            RG_TAtend.ItemIndex:= 0;
         end;
         salva_dados_os;
         monta_lvequip(wi_IDitem);
         btn_fechaatuclick(self);
      end;
   end
   else
      Showmessage('Preencha a Descrição');

end;

procedure Tfrm_COS_Atualizar.baixar_estoque(wp_peca,wp_qtde:integer);
var
   estoque     : TCPeca;
   log         : Tclogos;
begin
   estoque := TCPeca.Create(nil);
   estoque.Selecionar(wp_peca);
   if wp_qtde <> 0 then
   begin
      estoque.P_quantidade:= (estoque.P_quantidade - wp_qtde);
      estoque.Salvar(1);
   end;
   estoque.FreeOnRelease;
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'ALTEROU A PEÇA '+INTTOSTR(wp_peca)+' EM '+INTTOSTR(-wp_qtde)+'
UNIDADES NO ESTOQUE';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;

end;

procedure Tfrm_COS_Atualizar.Salvar_SItem_OS;
var
   qtde_anterior  : integer;
   os_SItem       : TcOrdemDeServico;
begin
   os_SItem                                    := TcOrdemDeServico.Create(nil);
   os_SItem.Items_OS.SubItensOS.Selecionar(wi_id_os,wi_IDitem,wi_IDSubitem);
   os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
   os_SItem.Items_OS.SubItensOS.P_id_os      := wi_id_os;
   os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
   os_SItem.Items_OS.SubItensOS.P_id_peca    := id_peca.AsInteger;
   qtde_anterior                             :=
os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   os_SItem.Items_OS.SubItensOS.P_qtde_peca  := edit_qtdepeca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_descricao  := edit_desc_atu.Text;

   //---> salva o registro
   if wi_IDSubitem = 0 then
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(0);
      // baixar do estoque a quantidade de peças
      baixar_estoque(id_peca.AsInteger,edit_qtdepeca.AsInteger);
   end
   else
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(1);
      // baixar do estoque a quantidade de peças
      baixar_estoque(id_peca.AsInteger,edit_qtdepeca.AsInteger - qtde_anterior);
   end;
   os_SItem.freeonrelease;
end;
```

```
procedure Tfrm_COS_Atualizar.lvMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDSubitem         := strtoint(lv.Selected.Caption);
      btn_alt_sit.Enabled  := True;
      btn_exc_sit.Enabled  := True;
    end
    else
    begin
       btn_alt_sit.Enabled  := False;
       btn_exc_sit.Enabled  := False;
       wi_IDSubitem         := 0;
    end;
  end
  else
  begin
    btn_alt_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
  end;
end;

procedure Tfrm_COS_Atualizar.btn_exc_sitClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este
movimento!','Exclusão',mb_YesNo) = IdYes then
   begin
      os_SItem                                  := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := wi_id_os;
      os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
      //---> exclui o registro
      if wi_IDSubitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Selecionar(wi_id_os,wi_IDitem,wi_IDSubitem);
         baixar_estoque(os_SItem.Items_OS.SubItensOS.P_id_peca,-
(os_SItem.Items_OS.SubItensOS.P_qtde_peca));
         os_SItem.Items_OS.SubItensOS.Salvar(2);
         if RG_TAtend.ItemIndex < 0 then
         begin
            RG_TAtend.ItemIndex:= 0;
         end;
         salva_dados_os;

         monta_lvequip(wi_IDitem);
      end;

      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_COS_Atualizar.btn_SalvaOSClick(Sender: TObject);
begin
  inherited;
      salva_dados_os;
end;

procedure Tfrm_COS_Atualizar.RG_TAtendClick(Sender: TObject);
begin
  inherited;
   btn_SalvaOSClick(Self);
end;
procedure Tfrm_COS_Atualizar.edit_valorMObraExit(Sender: TObject);
begin
  inherited;
   btn_SalvaOSClick(Self);
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.15 unit uCOS_Fechar;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, Mask, ToolEdit, CurrEdit, TCaixa,
  ExtCtrls, JvExExtCtrls, JvComponent, JvCaptionPanel, RXSplit, ComCtrls,
  fcButton, fcImgBtn, fcShapeBtn, TOrdemDeServico, Tpeca, Tlogos;

type
  Tfrm_COS_Fechar = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    lv_Pessoa: TListView;
    Panel6: TPanel;
    lv: TListView;
    RxSplitter1: TRxSplitter;
    Panel7: TPanel;
    GroupBox1: TGroupBox;
    e_numOS: TLabel;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    GroupBox2: TGroupBox;
    Label3: TLabel;
    Label6: TLabel;
    edit_valorpecas: TCurrencyEdit;
    edit_valorpecasNcobrar: TCurrencyEdit;
    GroupBox3: TGroupBox;
    Label7: TLabel;
    Label8: TLabel;
    edit_valordescontoMobra: TCurrencyEdit;
    edit_valordescontoPecas: TCurrencyEdit;
    GroupBox4: TGroupBox;
    Label1: TLabel;
    edit_valorTotal: TCurrencyEdit;
    e_descpessoa: TEdit;
    Label2: TLabel;
    p_csitens: TJvCaptionPanel;
    Panel8: TPanel;
    fcShapeBtn1: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    GroupBox5: TGroupBox;
    Label9: TLabel;
    v_total: TCurrencyEdit;
    rg_fpgto: TRadioGroup;
    GroupBox6: TGroupBox;
    Label10: TLabel;
    e_parc: TCurrencyEdit;
    Label11: TLabel;
    e_vcto: TCurrencyEdit;
    Label12: TLabel;
    v_parcela: TCurrencyEdit;
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure edit_valordescontoMobraExit(Sender: TObject);
    procedure edit_valordescontoPecasExit(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure e_parcExit(Sender: TObject);
  private
    procedure limpar_p_csitens;
    { Private declarations }
  public
     procedure Mostra_items_OS;
     procedure Valor_total_os;
     procedure Fechar_OS;
     procedure monta_lvequipamentos(wp_iddoItem : integer);
     procedure lanca_conta_receber(wp_id_pessoa,wp_parcela: integer; wp_valor : double;
wp_data : tdate);
    { Public declarations }
  end;

var
  frm_COS_Fechar : Tfrm_COS_Fechar;
  wi_IDdoitem    : Integer;
  wi_IDdoSubitem : Integer;
implementation

uses uLOS_Fechar, uMain;

{$R *.dfm}

{ Tfrm_COS_Fechar }

procedure Tfrm_COS_Fechar.Mostra_items_OS;
var
   pescontr    : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   pescontr:= TcOrdemDeServico.Create(frm_COS_Fechar);
   pescontr.Selecionar(strtoint(e_numOS.Caption));
   edit_valorOrc.Value  := pescontr.P_valor_orcado;
   edit_valorMObra.Value:= pescontr.P_valor_mao_obra;

   pescontr.Items_OS.Seleciona(strtoint(e_numOs.Caption));
   pescontr.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      pescontr.Items_OS.Coloca;
      if pescontr.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList          := lv_Pessoa.Items.Add;
         wl_ItemList.Caption  := inttostr(pescontr.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(pescontr.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(pescontr.Items_OS.P_desc_prob);

wl_ItemList.SubItems.Add(floattostr(pescontr.Items_OS.SubItensOS.Valor_total_SItens_iten
s(pescontr.Items_OS.P_id_os,pescontr.Items_OS.P_id_item_os)));

         pescontr.Items_OS.proximo;
      end;
   until pescontr.Items_OS.final;
   pescontr.freeonrelease;
   Valor_total_os;
end;

procedure Tfrm_COS_Fechar.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_COS_Fechar.Release;
   frm_COS_Fechar:= nil;
```

```
end;

procedure Tfrm_COS_Fechar.monta_lvequipamentos(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
begin
   pescontr       := TcOrdemDeServico.Create(nil);
   peca_sitem     := TcPeca.Create(nil);
   pescontr.Items_OS.SubItensOS.Seleciona(strtoint(e_numos.Caption),wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_cobrar = 1 then
            wl_ItemList.Checked    := True;
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
            wl_ItemList.SubItems.Add(floattostr(peca_sitem.P_valor_unitario));
            wl_ItemList.SubItems.Add(floattostr(peca_sitem.P_valor_unitario *
pescontr.Items_OS.SubItensOS.P_qtde_peca));
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
         end;
         wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);

         pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_COS_Fechar.lvMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
  pescontr        : TcOrdemDeServico;
  log             : Tclogos;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   itemselecionado := lv.GetItemAt(x, y);
   if (itemselecionado <> nil) then
   begin
      wi_IDdoSubitem  := strtoint(itemselecionado.Caption);
      log             := Tclogos.Create(nil);
      log.P_id_os     := strtoint(e_numOS.Caption);
      log.P_id_usuario := frm_main.Usuario.P_id_usuario;
      if (itemselecionado.Checked) then
        begin

pescontr.Items_OS.SubItensOS.Altera_Cobranca_pecas(strtoint(e_numOS.Caption),wi_iddoitem
,wi_IDdoSubitem,1);
           log.P_acao       := 'MARCOU A COBRANÇA DE UM ITEM NA OS';
        end
      else
       if not (itemselecionado.Checked) then
         begin

pescontr.Items_OS.SubItensOS.Altera_Cobranca_pecas(strtoint(e_numOS.Caption),wi_iddoitem
,wi_IDdoSubitem,0);
```

```
            log.P_acao       := 'CANCELOU A COBRANÇA DE UM ITEM NA OS';
          end;
        log.P_data       := now;
        log.Salvar(0);
        log.freeonrelease;
        Mostra_items_OS;
    end;

    pescontr.freeonrelease;

end;

procedure Tfrm_COS_Fechar.lv_PessoaMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
    itemselecionado := lv_Pessoa.GetItemAt(x, y);
    if (itemselecionado <> nil) then
    begin
      if (itemselecionado.Selected) then
      begin
        wi_IDdoitem          := strtoint(lv_Pessoa.Selected.Caption);
        monta_lvequipamentos(wi_iddoitem);
      end
      else
      begin
         wi_IDdoitem         := 0;
      end;
    end;
end;

procedure Tfrm_COS_Fechar.Valor_total_os;
var
  pescontr        : TcOrdemDeServico;
begin
    pescontr:= TcOrdemDeServico.Create(nil);
    edit_valorpecas.Value:=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1);
    edit_valorpecasNcobrar.Value:=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),0);
    edit_valorTotal.Value:= (edit_valorMObra.Value + edit_valorpecas.Value)-
(edit_valordescontoMobra.Value+edit_valordescontoPecas.Value);
    pescontr.freeonrelease;
end;

procedure Tfrm_COS_Fechar.edit_valordescontoMobraExit(Sender: TObject);
begin
  inherited;
   Valor_total_os;
end;

procedure Tfrm_COS_Fechar.edit_valordescontoPecasExit(Sender: TObject);
begin
  inherited;
   Valor_total_os;
end;

procedure Tfrm_COS_Fechar.btn_SalvaClick(Sender: TObject);
begin
  inherited;
   if Application.MessageBox('Deseja realmente FECHAR esta OS!'+#13+'Não sera mais
possível alterá-la','Conclusão de Ordem de Serviço',mb_YesNo) = IdYes then
    begin
       limpar_p_csitens;
       v_total.Value      := edit_valorTotal.Value;
       v_parcela.Value    := edit_valorTotal.Value;
       p_csitens.Left     := (frm_COS_Fechar.Width div 2) - (p_csitens.Width div 2);
       p_csitens.Top      := (frm_COS_Fechar.Height div 2) - (p_csitens.Height div 2);
       p_csitens.Visible := TRUE;
    end;
end;

procedure Tfrm_COS_Fechar.Fechar_OS;
    function mesparc( Data : TDateTime ): Integer;
    var
      Dia,Mes,Ano:Word;
```

```
   begin
     DecodeDate(Data,Ano,Mes,Dia);
     Result := mes;
   end;
   function anoparc( Data : TDateTime ): Integer;
   var
     Dia,Mes,Ano:Word;
   begin
     DecodeDate(Data,Ano,Mes,Dia);
     Result := Ano;
   end;

var
   os_Fechar    : TcOrdemDeServico;
   x            : integer;
   mes_pgtoparc: integer;
   ano_pgtoparc: integer;
   data_pgto    : tdate;
   LOG          : TcLogos;
begin
   os_Fechar                    := TcOrdemDeServico.Create(nil);
   os_Fechar.Selecionar(strtoint(e_numOS.Caption));
   os_Fechar.P_status           := 'FECHADA';
   os_Fechar.P_data_fechou      := now;
   os_Fechar.P_valor_cobrado    := edit_valorTotal.Value;
   os_Fechar.Salvar(1);
   mes_pgtoparc:= mesparc(now);
   ano_pgtoparc:= anoparc(now);
   for x:= 1 to e_parc.AsInteger do
   begin
      data_pgto:=
strtodate(e_vcto.Text+'/'+inttostr(mes_pgtoparc)+'/'+inttostr(ano_pgtoparc));
      Lanca_conta_receber(os_Fechar.P_id_pessoa,x,v_parcela.Value,data_pgto);
      mes_pgtoparc:= mes_pgtoparc + 1;
      if mes_pgtoparc > 12 then
      begin
         ano_pgtoparc:= ano_pgtoparc + 1;
         mes_pgtoparc:= 1;
      end;
   end;
   os_Fechar.freeonrelease;
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'FECHOU A OS';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;

procedure Tfrm_COS_Fechar.fcShapeBtn1Click(Sender: TObject);
begin
  inherited;
      Fechar_OS;
      p_csitens.Visible:= False;
      showmessage('OS: '+e_numOS.Caption+' Foi fechada com sucesso;');
      frm_LOS_Fechar.rgp_tipoClick(self);
      Close;
end;

procedure Tfrm_COS_Fechar.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;

procedure Tfrm_COS_Fechar.e_parcExit(Sender: TObject);
begin
  inherited;
   v_parcela.Value := (v_total.Value / e_parc.Value);
end;

procedure Tfrm_COS_Fechar.limpar_p_csitens;

   function diadomes( Data : TDateTime ): Integer;
   var
```

```
    Dia,Mes,Ano:Word;
  begin
    DecodeDate(Data,Ano,Mes,Dia);
    Result := Dia;
  end;

begin
  v_parcela.Clear;
  v_total.Clear;
  e_parc.Clear;
  e_vcto.Clear;
  e_vcto.Value:= diadomes(now);
  rg_fpgto.ItemIndex:= 0;
end;


procedure Tfrm_COS_Fechar.lanca_conta_receber(wp_id_pessoa,wp_parcela: integer; wp_valor
: double; wp_data : tdate);
var
  caixa       : TCCaixa;
begin
     caixa:= TCCaixa.Create(nil);
     caixa.P_data_emissao     := now;
     caixa.P_id_tipomov       := 2;
     caixa.P_id_pessoa        := wp_id_pessoa;
     caixa.p_Historico        := 'REFERENTE OS N° '+E_NUMOS.Caption;
     caixa.P_valor_apagar     := wp_valor;
     caixa.P_num_doc          := strtoint(e_numos.Caption);
     caixa.P_num_parcela      := wp_parcela;
     caixa.P_tipo_pgto        := rg_fpgto.ItemIndex;
     caixa.P_data_vcto        := wp_data;
     caixa.P_situacao         := 'A';
     caixa.Salvar(0);
     caixa.FreeOnRelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.16 unit uCPecas;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TPeca, funcoes, Mask, ToolEdit, CurrEdit;

type
  Tfrm_CPecas = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    Label1: TLabel;
    editUF: TComboBox;
    Label2: TLabel;
    Label4: TLabel;
    e_qtde: TCurrencyEdit;
    e_vunit: TCurrencyEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
     procedure Montartpcs;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CPecas    : Tfrm_CPecas;
  cadpeca       : TcPeca;

implementation

uses uLPecas;

{$R *.dfm}

procedure Tfrm_CPecas.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Estado deve ser preenchido');
   end
   else
      if (Edituf.Text = '') then
      begin
         showmessage('O campo da Sigla da UF deve ser preenchida');
      end
      else
      begin
         cadpeca.P_id_peca       := uLPecas.wi_IDdapeca;
         cadpeca.P_id_tipo_peca  := StrToInt(copy(editUF.Text,1,4));
         cadpeca.P_descricao     := Edit_DGrupo.Text;
```

```
         cadpeca.P_quantidade    := e_qtde.AsInteger;
         cadpeca.P_valor_unitario:= e_vunit.Value;

         //---> salva o registro
         if uLPecas.wi_IDdapeca = 0 then
         begin
            cadpeca.Salvar(0);
         end
         else
         begin
            cadpeca.Salvar(1);
         end;
         Close;
      end;
end;

procedure Tfrm_CPecas.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   cadpeca.freeonrelease;
   frm_LPecas.monta_lv;
   frm_CPecas.Release;
   frm_CPecas := nil;
end;

procedure Tfrm_CPecas.FormShow(Sender: TObject);
begin
  inherited;
   cadpeca := TcPeca.Create(frm_CPecas);
   cadpeca.Selecionar(uLPecas.wi_IDdapeca);

   if uLPecas.wi_IDdapeca = 0 then
      btn_limparclick(self)
   else
   Begin
      Edit_DGrupo.Text := cadpeca.P_descricao;
      e_qtde.AsInteger := cadpeca.P_quantidade;
      e_vunit.Value    := cadpeca.P_valor_unitario;
      cadpeca.Tipo_da_Peca.Selecionar(cadpeca.P_id_tipo_peca);
      Edituf.Text      := ZerosEsq(inttostr(cadpeca.Tipo_da_Peca.P_id_tipo_peca),4) + '-
' + cadpeca.Tipo_da_Peca.P_descricao;
   end;
   Montartpcs;
   ActiveControl := Edit_DGRUPO;
end;

procedure Tfrm_CPecas.Montartpcs;
var
   tpopec : TcPeca;
begin
   tpopec:= TcPeca.Create(nil);
   tpopec.Tipo_da_Peca.Seleciona;
   tpopec.Tipo_da_Peca.Primeiro;
   Edituf.Items.Clear;
   repeat
      tpopec.Tipo_da_Peca.Coloca;
      Edituf.Items.Add((ZerosEsq(inttostr(tpopec.Tipo_da_Peca.P_id_tipo_peca),4) + '-' +
tpopec.Tipo_da_Peca.P_descricao));
      tpopec.Tipo_da_Peca.proximo;
   until tpopec.Tipo_da_Peca.final;
   freeandnil(tpopec);
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.17 unit uCTDocumentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TTipoDocumento;

type
  Tfrm_CTDocumentos = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CTDocumentos: Tfrm_CTDocumentos;
  TDoc            : TcTipoDocumento;

implementation

uses uLTDocumentos;

{$R *.dfm}

procedure Tfrm_CTDocumentos.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Tipo deve ser preenchida');
   end
   else
   begin
      TDoc.P_id_tipo_doc    := ULTDocumentos.wi_IDTDoc;
      TDoc.P_descricao      := Edit_DGrupo.Text;

      //---> salva o registro
      if uLTDocumentos.wi_IDTDoc = 0 then
      begin
         TDoc.Salvar(0);
      end
      else
      begin
         TDoc.Salvar(1);
      end;
      Close;
   end;
end;
```

```
procedure Tfrm_CTDocumentos.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   TDoc.freeonrelease;
   frm_LTDocumentos.monta_lv;
   frm_CTDocumentos.Release;
   frm_CTDocumentos := nil;
end;

procedure Tfrm_CTDocumentos.FormShow(Sender: TObject);
begin
  inherited;
   TDoc := TcTipoDocumento.Create(frm_CTDocumentos);
   TDoc.Selecionar(uLTDocumentos.wi_IDTDoc);

   if uLTDocumentos.wi_IDTDoc = 0 then
      btn_limparclick(self)
   else
      Edit_DGrupo.Text        := TDoc.P_descricao;

   ActiveControl := Edit_DGRUPO;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.18 unit uCTEquipamentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TTipoDeEquipamento;

type
  Tfrm_CTEquipamentos = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CTEquipamentos: Tfrm_CTEquipamentos;
  Tequipament       : TcTipoDeEquipamento;

implementation

uses uLTEquipamentos;

{$R *.dfm}

procedure Tfrm_CTEquipamentos.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Tipo deve ser preenchida');
   end
   else
   begin
      Tequipament.P_id_tipo_equipamento := uLTEquipamentos.wi_IDTEquip;
      Tequipament.P_descricao           := Edit_DGrupo.Text;

      //---> salva o registro
      if uLTEquipamentos.wi_IDTEquip = 0 then
      begin
         Tequipament.Salvar(0);
      end
      else
      begin
         Tequipament.Salvar(1);
      end;
      Close;
   end;
end;
```

```
procedure Tfrm_CTEquipamentos.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   Tequipament.freeonrelease;
   frm_LTEquipamentos.monta_lv;
   frm_CTEquipamentos.Release;
   frm_CTEquipamentos := nil;
end;

procedure Tfrm_CTEquipamentos.FormShow(Sender: TObject);
begin
  inherited;
   Tequipament := TcTipoDeEquipamento.Create(frm_CTEquipamentos);
   Tequipament.Selecionar(uLTEquipamentos.wi_IDTEquip);

   if uLTEquipamentos.wi_IDTEquip = 0 then
      btn_limparclick(self)
   else
      Edit_DGrupo.Text        := Tequipament.P_descricao;

   ActiveControl := Edit_DGRUPO;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.19 unit uCTPecas;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TTipodePeca;

type
  Tfrm_CTPecas = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Edit_DGrupo: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CTPecas: Tfrm_CTPecas;
  TPeca      : TcTipodePeca;

implementation

uses uLTPecas;

{$R *.dfm}

procedure Tfrm_CTPecas.btn_SalvaClick(Sender: TObject);
begin
   inherited;

   if (Edit_DGrupo.Text = '') then
   begin
      showmessage('O campo Descrição do Tipo deve ser preenchida');
   end
   else
   begin
      TPeca.P_id_tipo_peca   := uLTPecas.wi_IDTpeca;
      TPeca.P_descricao      := Edit_DGrupo.Text;

      //---> salva o registro
      if uLTPecas.wi_IDTpeca = 0 then
      begin
         TPeca.Salvar(0);
      end
      else
      begin
         TPeca.Salvar(1);
      end;
      Close;
   end;
end;
```

```
procedure Tfrm_CTPecas.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   TPeca.freeonrelease;
   frm_LTPecas.monta_lv;
   frm_CTPecas.Release;
   frm_CTPecas := nil;
end;

procedure Tfrm_CTPecas.FormShow(Sender: TObject);
begin
  inherited;
   TPeca := TcTipodePeca.Create(frm_CTPecas);
   TPeca.Selecionar(uLTPecas.wi_IDTpeca);

   if uLTPecas.wi_IDTpeca = 0 then
      btn_limparclick(self)
   else
      Edit_DGrupo.Text        := TPeca.P_descricao;

   ActiveControl := Edit_DGRUPO;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.20 unit uCUsuarios;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, fcButton, fcImgBtn, fcShapeBtn, ExtCtrls,
  StdCtrls, TGrupo, TUsuario, funcoes;

type
  Tfrm_CUsuario = class(Tfrm_TelaCadastroPadrao)
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label11: TLabel;
    Edit_Nome: TEdit;
    rgp_Status: TRadioGroup;
    Combo_Grupo: TComboBox;
    Edit_Ramal: TEdit;
    Edit_Cargo: TEdit;
    Edit_EMail: TEdit;
    grp_Datas: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Edit_DtCadastro: TEdit;
    Edit_DtUAcesso: TEdit;
    Edit_NomeCompl: TEdit;
    EDIT_SENHA: TEdit;
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btn_limparClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    procedure Monta_Grupo;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_CUsuario: Tfrm_CUsuario;
  usuarioCA : TCUsuario;

implementation

uses uLUsuarios;

{$R *.dfm}

procedure Tfrm_CUsuario.btn_SalvaClick(Sender: TObject);
begin
  inherited;
      if (edit_nome.Text = '') then
```

```
      begin
         showmessage('O campo login não foi preenchido');
      end
      else
         if (edit_senha.Text = '') then
         begin
            showmessage('O campo senha não foi preenchido');
         end
         else
            if (combo_grupo.Text = '') then
            begin
               showmessage('O campo tipo grupo deve ser selecionado');
            end
            else
               begin
                  usuarioCA.P_id_usuario       := uLUsuarios.wi_IDUsu;
                  usuarioCA.P_login            := Edit_Nome.Text;
                  usuarioCA.P_nomecompleto     := Edit_NomeCompl.Text;
                  usuarioCA.P_status           := rgp_Status.ItemIndex;
                  usuarioCA.P_id_grupo         := StrToInt(copy(combo_Grupo.Text,1,4));
                  usuarioCA.P_ramal            := Edit_Ramal.Text;
                  usuarioCA.p_EMail            := trim(Edit_EMail.Text);
                  usuarioCA.P_cargo            := Edit_Cargo.Text;
                  usuarioCA.P_datacadastro     := now;
                  usuarioCA.P_dataultimoacesso := now;
                  usuarioCA.P_senha            := Edit_senha.Text;

                  //---> salva o registro
                  if uLUsuarios.wi_IDUsu = 0 then
                  begin
                     usuarioCA.Salvar(0);
                  end
                  else
                  begin
                     usuarioCA.Salvar(1);
                  end;
                  Close;
               end;
end;

procedure Tfrm_CUsuario.Monta_Grupo;
var
   Grupo : TCGrupo;
begin
   grupo:= TCGrupo.Create(nil);
   grupo.Seleciona;
   grupo.Primeiro;
   Combo_Grupo.Items.Clear;
   repeat
      grupo.Coloca;
      Combo_Grupo.Items.Add((ZerosEsq(inttostr(grupo.P_id_grupo),4) + '-' +
grupo.P_descricao));
      grupo.proximo;
   until grupo.final;
   freeandnil(grupo);
end;

procedure Tfrm_CUsuario.FormShow(Sender: TObject);
begin
   inherited;
   usuarioCA := TCUsuario.Create(frm_CUsuario);
   usuarioCA.Carregar_id(uLUsuarios.wi_IDUsu);
   if uLUsuarios.wi_IDUsu = 0 then
   begin
      btn_limparclick(self);
      rgp_Status.ItemIndex := 1;
      Edit_DTCadastro.Text := DateTimeToStr(Now);
      ActiveControl := Edit_Nome;
   end
   else
   begin
      Edit_Nome.Text          := usuarioCA.P_login;
      Edit_NomeCompl.Text     := usuarioCA.P_nomecompleto;
      Edit_senha.Text         := usuarioCA.P_senha;
      Edit_Ramal.Text         := usuarioCA.P_ramal;
      Edit_EMail.Text         := usuarioCA.P_email;
      Edit_Cargo.Text         := usuarioCA.P_cargo;
```

```
      Edit_DTCadastro.Text    := datetostr(usuarioCA.P_datacadastro);
      Edit_DtUAcesso.Text     := datetostr(usuarioCA.P_dataultimoacesso);
      rgp_Status.ItemIndex    := usuarioCA.P_status;
      usuarioCA.Grupo.Selecionar(usuarioCA.P_id_grupo);
      Combo_Grupo.Text        := (ZerosEsq(inttostr(usuarioCA.Grupo.P_id_grupo),4) + '-'
+ (trim(usuarioCA.Grupo.P_descricao)));
      ActiveControl           := Edit_Nome;
   end;
   Monta_Grupo;
end;

procedure Tfrm_CUsuario.btn_limparClick(Sender: TObject);
begin
  inherited;
  rgp_Status.ItemIndex := -1;
  Monta_Grupo;
end;

procedure Tfrm_CUsuario.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   usuarioCA.freeonrelease;
   frm_LUsuario.rgp_tipoClick(self);
   frm_CUsuario.Release;
   frm_CUsuario := nil;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.21 unit uDm_Base;

```
interface

uses
  SysUtils, Classes, DB, ADODB, Controls,Forms,Strutils, PSQLDbTables;

type
  Tdm_Base = class(TDataModule)
    my_conn: TPSQLDatabase;
    sql_Acao: TPSQLQuery;
    procedure my_connAfterConnect(Sender: TObject);
  private

    { Private declarations }
  public
    { Public declarations }
  end;

var
  dm_Base          : Tdm_Base;
  ws_sql           : String;

implementation

Uses uMain, funcoes;

{$R *.dfm}

procedure Tdm_Base.my_connAfterConnect(Sender: TObject);
begin
my_conn.CharSet    := 'LATIN2';
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.22 unit uLCidades;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TCidade, RXSpin, RXCtrls;

type
  Tfrm_LCidades = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LCidades: Tfrm_LCidades;
  wi_IDCidade : Integer;
  wi_IDEstauf : Integer;

implementation

uses uCCidades;


{$R *.dfm}

procedure Tfrm_LCidades.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LCidades.Release;
   frm_LCidades:= nil;
end;

procedure Tfrm_LCidades.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor     := crHourGlass;
   wi_IDCidade       := 0;
   wi_IDEstauf       := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CCidades) then
         frm_CCidades  := Tfrm_CCidades.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CCidades.show;
```

```
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LCidades.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor     := crHourGlass;
      wi_IDCidade       := strtoint(lv.Selected.Caption);
      wi_IDEstauf       := strtoint(lv.Selected.SubItems[1]);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CCidades) then
            frm_CCidades  := Tfrm_CCidades.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CCidades.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LCidades.monta_lv;
var
   cidtw       : TCCidade;
   wl_ItemList : TListItem;
begin
   cidtw:= TCCidade.Create(nil);
   cidtw.Seleciona;
   cidtw.Primeiro;
   lv.Items.Clear;
   repeat
      cidtw.Coloca;
      if cidtw.P_id_cidade <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(cidtw.P_id_cidade);
         wl_ItemList.SubItems.Add(cidtw.P_descricao);
         cidtw.UF.Selecionar(cidtw.P_id_uf);
         wl_ItemList.SubItems.Add(inttostr(cidtw.P_id_uf));
         wl_ItemList.SubItems.Add(cidtw.UF.P_abreviada+' - '+cidtw.UF.P_descricao);
      end;
      cidtw.proximo;
   until cidtw.final;
   cidtw.freeonrelease;
end;

procedure Tfrm_LCidades.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.23 unit uLClientes;

```
interface

uses
  Windows, uTela_ListaPadrao, Controls, ComCtrls, StdCtrls, ExtCtrls, forms,
  fcButton, fcImgBtn, fcShapeBtn, Classes, SysUtils, Dialogs, TPessoa;

type
  Tfrm_LCliente = class(Tfrm_ListaPadrao)
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    procedure listar_lv;
    { Public declarations }
  end;

var
  wi_Cliente    : Integer;
  frm_LCliente  : Tfrm_LCliente;
  Pessoalist    : TcPessoa;

implementation

uses uCClientes;


{$R *.dfm}

procedure Tfrm_LCliente.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;

   wi_Cliente       := 0;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CadClientes) then
         frm_CadClientes  := Tfrm_CadClientes.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CadClientes.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LCliente.btn_AlteraClick(Sender: TObject);
begin
```

```
    inherited;

    if wb_selecionoulinha then
    begin
    wi_Cliente       := strtoint(lv.Selected.Caption);

       if wi_Cliente <> 0 then
       begin
          Screen.Cursor    := crHourGlass;
          LockWindowUpDate(Handle);
          try
             if not assigned(frm_CadClientes) then
                frm_CadClientes  := Tfrm_CadClientes.Create(Self);

             // Seta Cursor normal
             Screen.Cursor := crDefault;

             frm_CadClientes.show;
          finally
             LockWindowUpdate(0);
             Screen.Cursor := crDefault;
             wb_selecionoulinha:=false;
          end;
       end;

    end
    else
    showmessage('Selecione um registro');

    wb_selecionoulinha:=false;

end;

procedure Tfrm_LCliente.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LCliente.Release;
   frm_LCliente := nil;
end;

procedure Tfrm_LCliente.FormShow(Sender: TObject);
begin
  inherited;
  Pessoalist:= TcPessoa.Create(frm_LCliente);
  listar_lv;
end;

procedure Tfrm_LCliente.listar_lv;
var
   wl_ItemList : TListItem;
begin
   Pessoalist.Seleciona(1);
   Pessoalist.Primeiro;
   lv.Items.Clear;
   repeat
      Pessoalist.Coloca;
      if Pessoalist.P_id_pessoa <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Pessoalist.P_id_pessoa);
         wl_ItemList.SubItems.Add(Pessoalist.P_nome);
         wl_ItemList.SubItems.Add(Pessoalist.P_endereco);
         wl_ItemList.SubItems.Add(Pessoalist.P_contato);
      end;
      Pessoalist.Proximo;
   until Pessoalist.Final;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.24 unit uLContratos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, Tpessoa, RXSpin, RXCtrls;

type
  Tfrm_LContratos = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LContratos: Tfrm_LContratos;
  wi_IDContrato : Integer;
  wi_IDPessoa   : Integer;
  wi_nome       : String;

implementation

uses uCContratos;


{$R *.dfm}

procedure Tfrm_LContratos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LContratos.Release;
   frm_LContratos:= nil;
end;

procedure Tfrm_LContratos.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDContrato    := 0;
   wi_IDPessoa      := 0;
   wi_nome          := '';
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CContratos) then
         frm_CContratos  := Tfrm_CContratos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;
```

```
      frm_CContratos.show;
      frm_CContratos.btn_pesqClick(SELF);

   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LContratos.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDContrato    := strtoint(lv.Selected.Caption);
      wi_IDPessoa      := strtoint(lv.Selected.SubItems[0]);
      wi_nome          := lv.Selected.SubItems[1];

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CContratos) then
            frm_CContratos  := Tfrm_CContratos.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CContratos.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LContratos.monta_lv;
var
   pescontr    : TCpessoa;
   wl_ItemList : TListItem;
begin
   pescontr:= TCpessoa.Create(frm_LContratos);
   pescontr.Contrato.Seleciona;
   pescontr.Contrato.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Contrato.Coloca;
      if pescontr.Contrato.P_id_contrato <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Contrato.P_id_contrato);
         pescontr.Selecionar(pescontr.Contrato.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(inttostr(pescontr.P_id_pessoa));
         wl_ItemList.SubItems.Add(pescontr.P_nome);
         wl_ItemList.SubItems.Add(datetostr(pescontr.Contrato.P_data_inicio_vigencia));
         wl_ItemList.SubItems.Add(datetostr(pescontr.Contrato.P_data_fim_vigencia));
         wl_ItemList.SubItems.Add(floattostr(pescontr.Contrato.P_valor_mensal));
         pescontr.Contrato.proximo;
      end;
   until pescontr.Contrato.final;
   pescontr.freeonrelease;
end;

procedure Tfrm_LContratos.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.25 unit uLCPagar;

```
interface

uses
  Windows, SysUtils,Graphics, uTela_ListaPadrao, StdCtrls, Mask, ToolEdit, forms,
dialogs,
  Controls, ComCtrls, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn, Classes, TCaixa,
TPessoa,
  CurrEdit, JvPanel, JvExExtCtrls, JvComponent, JvCaptionPanel;

type
  Tfrm_LCPagar = class(Tfrm_ListaPadrao)
    Panel3: TPanel;
    Edit_DataIni: TDateEdit;
    Edit_DataFim: TDateEdit;
    Label1: TLabel;
    Label2: TLabel;
    btn_Filtra: TfcShapeBtn;
    P_Cancelamento: TJvCaptionPanel;
    JvPanel2: TJvPanel;
    BTN_CANCELAR: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    GroupBox1: TGroupBox;
    e_numOS: TLabel;
    Label3: TLabel;
    val_juros: TCurrencyEdit;
    Label4: TLabel;
    val_pago: TCurrencyEdit;
    Label9: TLabel;
    edit_data_pgto: TDateEdit;
    btn_baixar: TfcShapeBtn;
    btn_Exclui: TfcShapeBtn;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_FiltraClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure btn_ExcluiClick(Sender: TObject);
    procedure edit_data_pgtoEnter(Sender: TObject);
    procedure BTN_CANCELARClick(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_baixarClick(Sender: TObject);
    procedure val_pagoExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure Monta_Lista;
    procedure verifica_juros;

  end;

var
  frm_LCPagar     : Tfrm_LCPagar;
  wi_idLcto       : Integer;
```

```
  ws_Num          : String;
  wn_Num          : Double;
  Wd_valorpago    : Double;


implementation

uses uDm_Base, uMovCaixa, Funcoes;

{$R *.dfm}

procedure Tfrm_LCPagar.FormShow(Sender: TObject);
begin
  inherited;
   Edit_DataIni.Date := Date;
   Edit_DataFim.Date := Date;
   Monta_Lista;
end;

procedure Tfrm_LCPagar.Monta_Lista;
var
   caixa       : TCCaixa;
   pessoa      : TcPessoa;
   wl_ItemList : TListItem;
begin
   caixa := TCCaixa.Create(nil);
   caixa.Seleciona(Edit_Dataini.Date,Edit_DataFim.Date,1);
   caixa.Primeiro;
   lv.Items.Clear;
   repeat
      caixa.Coloca;
      if caixa.P_id_lcto <> 0 then
      begin
         pessoa:= TcPessoa.Create(nil);
         pessoa.Selecionar(caixa.P_id_pessoa,2);
         wl_ItemList             := lv.Items.Add;
         wl_ItemList.Caption     := inttostr(caixa.P_id_lcto);
         wl_ItemList.SubItems.Add(inttostr(caixa.P_num_doc));
         wl_ItemList.SubItems.Add(datetostr(caixa.P_data_vcto));
         wl_ItemList.SubItems.Add(pessoa.P_nome);
         wl_ItemList.SubItems.Add(caixa.P_historico);
         wl_ItemList.SubItems.Add(floattostr(caixa.P_valor_apagar));
         wl_ItemList.SubItems.Add(floattostr(caixa.P_id_tipomov));
         pessoa.FreeOnRelease;
      end;
      caixa.Proximo;
   until caixa.Final;
   caixa.FreeOnRelease;
end;

procedure Tfrm_LCPagar.FormClose(Sender: TObject;  var Action: TCloseAction);
begin
  inherited;
   frm_LCPagar.Release;
   frm_LCPagar := nil;
end;

procedure Tfrm_LCPagar.btn_IncluirClick(Sender: TObject);
begin
  inherited;

   Screen.Cursor := crHourGlass;

   wi_idLcto     := 0;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_MovCaixa) then
         frm_MovCaixa := Tfrm_MovCaixa.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_MovCaixa.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LCPagar.btn_FiltraClick(Sender: TObject);
begin
  inherited;

   if Edit_DataFim.Date < Edit_DataIni.Date then
   begin
      Edit_DataFim.Date := Edit_DataIni.Date;
      Edit_DataFim.Refresh;
   end;

   Monta_Lista;
end;

procedure Tfrm_LCPagar.btn_AlteraClick(Sender: TObject);
begin
  inherited;

   if wb_selecionoulinha then
   begin
      Screen.Cursor := crHourGlass;

      wi_idLcto     := StrToInt(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_MovCaixa) then
            frm_MovCaixa := Tfrm_MovCaixa.Create(Self);


         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_MovCaixa.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCPagar.btn_ExcluiClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
  inherited;

   if wb_selecionoulinha then
   begin
      wi_idLcto     := StrToInt(lv.Selected.Caption);

      if Application.MessageBox('Deseja realmente Excluir este
Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
      begin
         caixa := TCCaixa.Create(nil);
         caixa.Selecionar(wi_idLcto);
         caixa.P_situacao    := 'C';
         caixa.Salvar(1);
         caixa.FreeOnRelease;

         Monta_Lista;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCPagar.edit_data_pgtoEnter(Sender: TObject);
begin
```

```
  inherited;
   edit_data_pgto.Date := Date;
end;

procedure Tfrm_LCPagar.BTN_CANCELARClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
  inherited;
   verifica_juros;
   caixa := TCCaixa.Create(nil);
   caixa.Selecionar(wi_idLcto);
   caixa.P_data_pgto   := edit_data_pgto.Date;
   caixa.P_valor_juro  := val_juros.Value;
   caixa.P_valor_pago  := val_pago.Value;
   caixa.P_situacao    := 'P';
   caixa.Salvar(1);
   caixa.FreeOnRelease;

   Monta_Lista;
   btn_fechaatuClick(self);
end;

procedure Tfrm_LCPagar.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
   P_Cancelamento.Visible  := False;
end;

procedure Tfrm_LCPagar.btn_baixarClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      wi_idLcto     := StrToInt(lv.Selected.Caption);

      if Application.MessageBox('Deseja realmente Baixar este
Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
      begin
         P_Cancelamento.Visible  := True;
         e_numos.Caption         := lv.Selected.SubItems[0];
         val_pago.Text           := lv.Selected.SubItems[4];
         Wd_valorpago            := val_pago.Value;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCPagar.val_pagoExit(Sender: TObject);
begin
  inherited;
   verifica_juros;
end;

procedure Tfrm_LCPagar.verifica_juros;
begin
   if val_pago.Value > Wd_valorpago then
   begin
     val_juros.Value:= val_pago.Value - Wd_valorpago;
     val_pago.Value := Wd_valorpago
   end;

end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.26 unit uLCReceber;

```
interface

uses
  Windows, SysUtils,Graphics, uTela_ListaPadrao, StdCtrls, Mask, ToolEdit, forms,
dialogs,
  Controls, ComCtrls, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn, Classes, TCaixa,
TPessoa,
  CurrEdit, JvPanel, JvExExtCtrls, JvComponent, JvCaptionPanel;

type
  Tfrm_LCReceber = class(Tfrm_ListaPadrao)
    Panel3: TPanel;
    Edit_DataIni: TDateEdit;
    Edit_DataFim: TDateEdit;
    Label1: TLabel;
    Label2: TLabel;
    btn_Filtra: TfcShapeBtn;
    P_Cancelamento: TJvCaptionPanel;
    JvPanel2: TJvPanel;
    BTN_CANCELAR: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    GroupBox1: TGroupBox;
    e_numOS: TLabel;
    Label3: TLabel;
    val_juros: TCurrencyEdit;
    Label4: TLabel;
    val_pago: TCurrencyEdit;
    Label9: TLabel;
    edit_data_pgto: TDateEdit;
    btn_baixar: TfcShapeBtn;
    btn_Exclui: TfcShapeBtn;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_FiltraClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure btn_ExcluiClick(Sender: TObject);
    procedure edit_data_pgtoEnter(Sender: TObject);
    procedure BTN_CANCELARClick(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_baixarClick(Sender: TObject);
    procedure val_pagoExit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure Monta_Lista;
    procedure verifica_juros;

  end;

var
  frm_LCReceber    : Tfrm_LCReceber;
  wi_idLcto      : Integer;
```

```
  ws_Num          : String;
  wn_Num          : Double;
  Wd_valorpago    : Double;


implementation

uses Funcoes, uMovCaixaCReceber;

{$R *.dfm}

procedure Tfrm_LCReceber.FormShow(Sender: TObject);
begin
  inherited;
   Edit_DataIni.Date := Date;
   Edit_DataFim.Date := Date;
   Monta_Lista;
end;

procedure Tfrm_LCReceber.Monta_Lista;
var
   caixa       : TCCaixa;
   pessoa      : TcPessoa;
   wl_ItemList : TListItem;
begin
   caixa := TCCaixa.Create(nil);
   caixa.Seleciona(Edit_Dataini.Date,Edit_DataFim.Date,0);
   caixa.Primeiro;
   lv.Items.Clear;
   repeat
      caixa.Coloca;
      if caixa.P_id_lcto <> 0 then
      begin
         pessoa:= TcPessoa.Create(nil);
         pessoa.Selecionar(caixa.P_id_pessoa,1);
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(caixa.P_id_lcto);
         wl_ItemList.SubItems.Add(inttostr(caixa.P_num_doc));
         wl_ItemList.SubItems.Add(datetostr(caixa.P_data_vcto));
         wl_ItemList.SubItems.Add(pessoa.P_nome);
         wl_ItemList.SubItems.Add(caixa.P_historico);
         wl_ItemList.SubItems.Add(floattostr(caixa.P_valor_apagar));
         wl_ItemList.SubItems.Add(floattostr(caixa.P_id_tipomov));
         pessoa.FreeOnRelease;
      end;
      caixa.Proximo;
   until caixa.Final;
   caixa.FreeOnRelease;
end;

procedure Tfrm_LCReceber.FormClose(Sender: TObject;  var Action: TCloseAction);
begin
  inherited;
   frm_LCReceber.Release;
   frm_LCReceber := nil;
end;

procedure Tfrm_LCReceber.btn_IncluirClick(Sender: TObject);
begin
  inherited;

   Screen.Cursor := crHourGlass;

   wi_idLcto     := 0;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_MovCaixaCReceber) then
         frm_MovCaixaCReceber := Tfrm_MovCaixaCReceber.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_MovCaixaCReceber.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LCReceber.btn_FiltraClick(Sender: TObject);
begin
  inherited;

   if Edit_DataFim.Date < Edit_DataIni.Date then
   begin
      Edit_DataFim.Date := Edit_DataIni.Date;
      Edit_DataFim.Refresh;
   end;

   Monta_Lista;
end;

procedure Tfrm_LCReceber.btn_AlteraClick(Sender: TObject);
begin
  inherited;

   if wb_selecionoulinha then
   begin
      Screen.Cursor := crHourGlass;

      wi_idLcto     := StrToInt(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_MovCaixaCReceber) then
            frm_MovCaixaCReceber := Tfrm_MovCaixaCReceber.Create(Self);


         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_MovCaixaCReceber.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCReceber.btn_ExcluiClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
  inherited;

   if wb_selecionoulinha then
   begin
      wi_idLcto     := StrToInt(lv.Selected.Caption);

      if Application.MessageBox('Deseja realmente Excluir este
Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
      begin
         caixa := TCCaixa.Create(nil);
         caixa.Selecionar(wi_idLcto);
         caixa.P_situacao    := 'C';
         caixa.Salvar(1);
         caixa.FreeOnRelease;

         Monta_Lista;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCReceber.edit_data_pgtoEnter(Sender: TObject);
begin
```

```
  inherited;
   edit_data_pgto.Date := Date;
end;

procedure Tfrm_LCReceber.BTN_CANCELARClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
  inherited;
   verifica_juros;
   caixa := TCCaixa.Create(nil);
   caixa.Selecionar(wi_idLcto);
   caixa.P_data_pgto    := edit_data_pgto.Date;
   caixa.P_valor_juro   := val_juros.Value;
   caixa.P_valor_pago   := val_pago.Value;
   caixa.P_situacao     := 'P';
   caixa.Salvar(1);
   caixa.FreeOnRelease;

   Monta_Lista;
   btn_fechaatuClick(self);
end;

procedure Tfrm_LCReceber.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
   P_Cancelamento.Visible  := False;
end;

procedure Tfrm_LCReceber.btn_baixarClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      wi_idLcto     := StrToInt(lv.Selected.Caption);

      if Application.MessageBox('Deseja realmente Baixar este
Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
      begin
         P_Cancelamento.Visible  := True;
         e_numos.Caption         := lv.Selected.SubItems[0];
         val_pago.Text           := lv.Selected.SubItems[4];
         Wd_valorpago            := val_pago.Value;
      end;
   end
   else
   begin
      showmessage('Selecione um Registro!!!');
   end;
end;

procedure Tfrm_LCReceber.val_pagoExit(Sender: TObject);
begin
  inherited;
   verifica_juros;
end;

procedure Tfrm_LCReceber.verifica_juros;
begin
   if val_pago.Value > Wd_valorpago then
   begin
     val_juros.Value:= val_pago.Value - Wd_valorpago;
     val_pago.Value := Wd_valorpago
   end;

end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.27 unit uLEquipamentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, RXSplit, ComCtrls, StdCtrls, ExtCtrls,
  fcButton, fcImgBtn, fcShapeBtn, TPessoa;

type
  Tfrm_LEquipamentos = class(Tfrm_ListaPadrao)
    lv_Pessoa: TListView;
    RxSplitter1: TRxSplitter;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure lv_PessoaClick(Sender: TObject);
    procedure lvClick(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    procedure monta_lv_Pessoa;
    { Private declarations }
  public
    procedure monta_lvequip(wp_iddoContrato : integer);
    { Public declarations }
  end;

var
  frm_LEquipamentos           : Tfrm_LEquipamentos;
  wi_IDContrato               : Integer;
  wi_IDEquipamento            : Integer;
  wb_selecionoulinhaContrato  : Boolean;

implementation

uses uCEquipamentos;

{$R *.dfm}

procedure Tfrm_LEquipamentos.FormShow(Sender: TObject);
begin
  inherited;
  monta_lv_Pessoa;
end;

procedure Tfrm_LEquipamentos.monta_lv_Pessoa;
var
   pescontr    : TCpessoa;
   wl_ItemList : TListItem;
begin
   pescontr:= TCpessoa.Create(frm_LEquipamentos);
   pescontr.Contrato.Seleciona;
   pescontr.Contrato.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
```

```
      pescontr.Contrato.Coloca;
      if pescontr.Contrato.P_id_contrato <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Contrato.P_id_contrato);
         pescontr.Selecionar(pescontr.Contrato.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(inttostr(pescontr.P_id_pessoa));
         wl_ItemList.SubItems.Add(pescontr.P_nome);
         pescontr.Contrato.proximo;
      end;
   until pescontr.Contrato.final;
   pescontr.freeonrelease;
end;

procedure Tfrm_LEquipamentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LEquipamentos.Release;
   frm_LEquipamentos:= nil;
end;

procedure Tfrm_LEquipamentos.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDEquipamento := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CEquipamentos) then
         frm_CEquipamentos  := Tfrm_CEquipamentos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CEquipamentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LEquipamentos.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDEquipamento := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CEquipamentos) then
            frm_CEquipamentos  := Tfrm_CEquipamentos.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CEquipamentos.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LEquipamentos.lv_PessoaClick(Sender: TObject);
begin
  inherited;
   btn_Altera.Enabled  := False;
end;

procedure Tfrm_LEquipamentos.lvClick(Sender: TObject);
```

```
begin
  inherited;
   if wi_IDContrato <> 0 then
   begin
      btn_Altera.Enabled  := True;
   end;
end;

procedure Tfrm_LEquipamentos.monta_lvequip(wp_iddoContrato : integer);
var
   pescontr    : TCpessoa;
   wl_ItemList : TListItem;
begin
   pescontr:= TCpessoa.Create(frm_LEquipamentos);
   pescontr.Equipamento.Seleciona(wp_iddoContrato);
   pescontr.Equipamento.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Equipamento.Coloca;
      if pescontr.Equipamento.P_id_equipamento <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Equipamento.P_id_equipamento);
         wl_ItemList.SubItems.Add(pescontr.Equipamento.P_numero_serie);
         wl_ItemList.SubItems.Add(pescontr.Equipamento.P_descricao);
         pescontr.Equipamento.proximo;
      end;
   until pescontr.Equipamento.final;
   pescontr.freeonrelease;
end;

procedure Tfrm_LEquipamentos.lv_PessoaMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv_Pessoa.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDContrato        := strtoint(lv_pessoa.Selected.Caption);
      btn_Incluir.Enabled  := True;
      monta_lvequip(wi_IDContrato);
    end
    else
    begin
       btn_Incluir.Enabled  := False;
       wi_IDContrato        := 0;
    end;
  end
  else
    btn_Incluir.Enabled  := False;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.28 unit uLEstados;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TUf, RXSpin, RXCtrls;

type
  Tfrm_LEstados = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LEstados: Tfrm_LEstados;
  wi_IDEstado : Integer;

implementation

uses uCTDocumentos, uCEstados;

{$R *.dfm}

procedure Tfrm_LEstados.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LEstados.Release;
   frm_LEstados:= nil;
end;

procedure Tfrm_LEstados.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDEstado        := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CEstados) then
         frm_CEstados  := Tfrm_CEstados.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CEstados.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LEstados.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDEstado        := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CEstados) then
            frm_CEstados  := Tfrm_CEstados.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CEstados.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LEstados.monta_lv;
var
   estuf       : TCuf;
   wl_ItemList : TListItem;
begin
   estuf:= TCuf.Create(nil);
   estuf.Seleciona;
   estuf.Primeiro;
   lv.Items.Clear;
   repeat
      estuf.Coloca;
      if estuf.P_id_uf <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(estuf.P_id_uf);
         wl_ItemList.SubItems.Add(estuf.P_abreviada);
         wl_ItemList.SubItems.Add(estuf.P_descricao);
      end;
      estuf.proximo;
   until estuf.final;
   estuf.freeonrelease;
end;

procedure Tfrm_LEstados.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.29 unit uLFornecedores;

```
interface

uses
  Windows, uTela_ListaPadrao, Controls, ComCtrls, StdCtrls, ExtCtrls, forms,
  fcButton, fcImgBtn, fcShapeBtn, Classes, SysUtils, Dialogs, TPessoa,
  RXSpin, RXCtrls;

type
  Tfrm_LFornecedores = class(Tfrm_ListaPadrao)
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private

    { Private declarations }
  public
     procedure listar_lv;
    { Public declarations }
  end;

var
  wi_Cliente         : Integer;
  frm_LFornecedores  : Tfrm_LFornecedores;
  Pessoalist         : TcPessoa;

implementation

uses uCFornecedores;


{$R *.dfm}

procedure Tfrm_LFornecedores.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;

   wi_Cliente       := 0;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CadFornecedores) then
         frm_CadFornecedores  := Tfrm_CadFornecedores.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CadFornecedores.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;
```

```
procedure Tfrm_LFornecedores.btn_AlteraClick(Sender: TObject);
begin
   inherited;

   if wb_selecionoulinha then
   begin
   wi_Cliente       := strtoint(lv.Selected.Caption);

      if wi_Cliente <> 0 then
      begin
         Screen.Cursor    := crHourGlass;
         LockWindowUpDate(Handle);
         try
            if not assigned(frm_CadFornecedores) then
               frm_CadFornecedores  := Tfrm_CadFornecedores.Create(Self);

            // Seta Cursor normal
            Screen.Cursor := crDefault;

            frm_CadFornecedores.show;
         finally
            LockWindowUpdate(0);
            Screen.Cursor := crDefault;
            wb_selecionoulinha:=false;
         end;
      end;

   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;

end;

procedure Tfrm_LFornecedores.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LFornecedores.Release;
   frm_LFornecedores := nil;
end;

procedure Tfrm_LFornecedores.FormShow(Sender: TObject);
begin
  inherited;
  Pessoalist:= TcPessoa.Create(frm_LFornecedores);
  listar_lv;
end;

procedure Tfrm_LFornecedores.listar_lv;
var
   wl_ItemList : TListItem;
begin
   Pessoalist.Seleciona(2);
   Pessoalist.Primeiro;
   lv.Items.Clear;
   repeat
      Pessoalist.Coloca;
      if Pessoalist.P_id_pessoa <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Pessoalist.P_id_pessoa);
         wl_ItemList.SubItems.Add(Pessoalist.P_nome);
         wl_ItemList.SubItems.Add(Pessoalist.P_endereco);
         wl_ItemList.SubItems.Add(Pessoalist.P_contato);
      end;
      Pessoalist.Proximo;
   until Pessoalist.Final;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.30 unit uLGrupoDeUsuarios;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, funcoes, TGrupo;

type
  Tfrm_LGrupos = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LGrupos: Tfrm_LGrupos;
  wi_IDGrupo : Integer;

implementation

uses uCGrupoDeUsuarios;

{$R *.dfm}

procedure Tfrm_LGrupos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LGrupos.Release;
   frm_LGrupos:= nil;
end;

procedure Tfrm_LGrupos.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDGrupo       := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CGrupo) then
         frm_CGrupo  := Tfrm_CGrupo.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CGrupo.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LGrupos.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDGrupo       := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CGrupo) then
            frm_CGrupo  := Tfrm_CGrupo.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CGrupo.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LGrupos.monta_lv;
var
   Grupo       : TCGrupo;
   wl_ItemList : TListItem;
begin
   grupo:= TCGrupo.Create(nil);
   grupo.Seleciona;
   grupo.Primeiro;
   lv.Items.Clear;
   repeat
      grupo.Coloca;
      if grupo.P_id_grupo <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := ZerosEsq(inttostr(grupo.P_id_grupo),4);
         wl_ItemList.SubItems.Add(grupo.P_descricao);
      end;
      grupo.proximo;
   until grupo.final;
   freeandnil(grupo);
end;

procedure Tfrm_LGrupos.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.31 unit uListaCaixa;

```
interface

uses
  Windows, TCaixa, uTela_ListaPadrao, StdCtrls, Mask, ToolEdit, Controls,Graphics,
TTipoMov,
  ComCtrls, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn, Classes, forms, SysUtils,TPessoa,
  fcLabel;

type
  Tfrm_ListaCaixa = class(Tfrm_ListaPadrao)
    Panel3: TPanel;
    Edit_DataIni: TDateEdit;
    Edit_DataFim: TDateEdit;
    Label1: TLabel;
    Label2: TLabel;
    btn_Filtra: TfcShapeBtn;
    GroupBox1: TGroupBox;
    fcLabel1: TfcLabel;
    fcLabel2: TfcLabel;
    fcLabel3: TfcLabel;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_FiltraClick(Sender: TObject);
    procedure lvCustomDrawItem(Sender: TCustomListView; Item: TListItem; State:
TCustomDrawState; var DefaultDraw: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure Monta_Lista;
    procedure verifica_pendencias;

  end;

var
  frm_ListaCaixa : Tfrm_ListaCaixa;
  wi_idLcto      : Integer;
  ws_Num         : String;
  wn_Num         : Double;
  wn_saldo       : Double;

implementation

uses uMovCaixa, Funcoes;

{$R *.dfm}

procedure Tfrm_ListaCaixa.FormShow(Sender: TObject);
begin
  inherited;

   Edit_DataIni.Date := Date;
   Edit_DataFim.Date := Date;
```

```
   Monta_Lista;
end;

procedure Tfrm_ListaCaixa.Monta_Lista;
var
   caixa          : TCCaixa;
   pessoa         : TcPessoa;
   wl_ItemList    : TListItem;
   tipomovimento  : TcTipoMov;
   wn_valor       : Double;
begin
   caixa := TCCaixa.Create(nil);
   lv.Items.Clear;
   verifica_pendencias;
   caixa.Seleciona(Edit_Dataini.Date,Edit_DataFim.Date,2);
   caixa.Primeiro;
   repeat
      caixa.Coloca;
      if caixa.P_id_lcto <> 0 then
      begin
         tipomovimento:= TcTipoMov.Create(nil);
         tipomovimento.Selecionar(caixa.P_id_tipomov);
         pessoa:= TcPessoa.Create(nil);
         if tipomovimento.P_tipomov = 1 then
            pessoa.Selecionar(caixa.P_id_pessoa,2)
         else
            pessoa.Selecionar(caixa.P_id_pessoa,1);

         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(caixa.P_id_lcto);
         wl_ItemList.SubItems.Add(inttostr(caixa.P_num_doc));
         wl_ItemList.SubItems.Add(datetostr(caixa.P_data_vcto));
         wl_ItemList.SubItems.Add(pessoa.P_nome);
         wl_ItemList.SubItems.Add(caixa.P_historico);
         wl_ItemList.SubItems.Add(inttostr(tipomovimento.P_tipomov));
         if caixa.P_situacao = 'A' then
            wn_valor:= caixa.P_valor_apagar
         else
            wn_valor:= caixa.P_valor_pago + caixa.P_valor_juro;

         wl_ItemList.SubItems.Add(floattostr(wn_valor));

         if tipomovimento.P_tipomov = 1 then // a pagar
            wn_saldo:= (wn_saldo - wn_valor)
         else
            wn_saldo:= (wn_saldo + wn_valor);

         wl_ItemList.SubItems.Add(floattostr(wn_saldo));
         wl_ItemList.SubItems.Add(caixa.P_situacao);
         pessoa.FreeOnRelease;
      end;
      caixa.Proximo;
   until caixa.Final;
   caixa.FreeOnRelease;
end;


procedure Tfrm_ListaCaixa.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_ListaCaixa.Release;
   frm_ListaCaixa := nil;
end;

procedure Tfrm_ListaCaixa.btn_FiltraClick(Sender: TObject);
begin
  inherited;

   if Edit_DataFim.Date < Edit_DataIni.Date then
   begin
      Edit_DataFim.Date := Edit_DataIni.Date;
      Edit_DataFim.Refresh;
   end;

   Monta_Lista;
end;
```

```
procedure Tfrm_ListaCaixa.lvCustomDrawItem(Sender: TCustomListView;
  Item: TListItem; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
  inherited;

   ws_Num := Item.SubItems[6];

   ws_Num := ReplaceStr(ws_Num, '.', '');

   wn_Num := StrToFloat(ws_Num);

   if trim(Item.SubItems[4]) = '0' then
   begin
      lv.Canvas.Brush.Color:= clSkyblue;
      lv.Canvas.Font.Color := clBlue;
   end
   else
   begin
      lv.Canvas.Brush.Color:= clmoneygreen;
      lv.Canvas.Font.Color := clTeal;
   end;

   if wn_Num < 0 then
   begin
      lv.Canvas.Font.Color:= clRed;
   end;

end;

procedure Tfrm_ListaCaixa.verifica_pendencias;
var
   TOT_RECEB      : Double;
   tot_paga       : Double;
   caixa          : TCCaixa;
   wl_ItemList    : TListItem;
begin
   caixa := TCCaixa.Create(nil);

   //  total de contas a receber
   wl_ItemList            := lv.Items.Add;
   wl_ItemList.Caption    := '';
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('Ã RECEBER');
   wl_ItemList.SubItems.Add('0');
   TOT_RECEB:= caixa.calcular(Edit_Dataini.Date-1,0);
   wl_ItemList.SubItems.Add(floattostr(TOT_RECEB));
   wl_ItemList.SubItems.Add(floattostr(TOT_RECEB));
   wl_ItemList.SubItems.Add('');
   //  total de contas a pagar
   wl_ItemList            := lv.Items.Add;
   wl_ItemList.Caption    := '';
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('');
   wl_ItemList.SubItems.Add('Ã PAGAR');
   wl_ItemList.SubItems.Add('1');
   tot_paga:= caixa.calcular(Edit_Dataini.Date-1,1);
   wl_ItemList.SubItems.Add(floattostr(tot_paga));
   wl_ItemList.SubItems.Add(floattostr(TOT_RECEB - tot_paga));
   wl_ItemList.SubItems.Add('');
   wn_saldo := (TOT_RECEB - tot_paga);

   caixa.FreeOnRelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.32 unit uLLiberaOrcar;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn,TOrdemDeServico, TPessoa, Tlogos;

type
  Tfrm_LEfetua = class(Tfrm_ListaPadrao)
    fcShapeBtn1: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_AlteraClick(Sender: TObject);
    procedure rgp_tipoClick(Sender: TObject);
    procedure fcShapeBtn1Click(Sender: TObject);
    procedure btn_IncluirClick(Sender: TObject);
  private
    procedure listar_lv(wp_tipo: integer);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_LEfetua: Tfrm_LEfetua;
  wi_id_os   : Integer;

implementation

uses uCAltera_Orcamento, uMain;

{$R *.dfm}

procedure Tfrm_LEfetua.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LEfetua.Release;
   frm_LEfetua := nil;
end;

procedure Tfrm_LEfetua.listar_lv(wp_tipo:integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(WP_TIPO,5);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
```

```
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         wl_ItemList.SubItems.Add(floattostr(os_atu.P_valor_orcado));
      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_LEfetua.btn_AlteraClick(Sender: TObject);
var
   os_atualiza    : TcOrdemDeServico;
   log            : Tclogos;
begin
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você deseja Liberar este
Orçamento!','Aprovação de Orçamento',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'ORÇAR';
         os_atualiza.Salvar(1);
         os_atualiza.freeonrelease;
         log               := Tclogos.Create(nil);
         log.P_id_os       := strtoint(lv.Selected.Caption);
         log.P_id_usuario  := frm_main.Usuario.P_id_usuario;
         log.P_acao        := 'LIBEROU PARA A EFETIVAÇÃO DO ORÇAMENTO';
         log.P_data        := now;
         log.Salvar(0);
         log.freeonrelease;
         rgp_tipoclick(self);
      end;
   end
   else
   showmessage('Selecione um registro');
end;

procedure Tfrm_LEfetua.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar_lv(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LEfetua.fcShapeBtn1Click(Sender: TObject);
var
   os_atualiza    : TcOrdemDeServico;
   log            : Tclogos;
begin
  inherited;
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você CANCELAR Orçamento!','C A N C E L
A M E N T O',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'CANCELADA';
         os_atualiza.Salvar(1);
         os_atualiza.freeonrelease;
         rgp_tipoclick(self);
         log               := Tclogos.Create(nil);
         log.P_id_os       := strtoint(lv.Selected.Caption);
         log.P_id_usuario  := frm_main.Usuario.P_id_usuario;
         log.P_acao        := 'CANCELOU O ORÇAMENTO';
         log.P_data        := now;
         log.Salvar(0);
         log.freeonrelease;
         rgp_tipoclick(self);
      end;
   end
   else
   showmessage('Selecione um registro');
```

```
end;

procedure Tfrm_LEfetua.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_id_os         := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_AltOrcamento) then
            frm_AltOrcamento  := Tfrm_AltOrcamento.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;
         frm_AltOrcamento.e_nOS.AsInteger:= wi_id_os;
         frm_AltOrcamento.monta_item_os;
         frm_AltOrcamento.e_descpessoa.Text:= lv.Selected.SubItems[0];
         frm_AltOrcamento.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.33 unit uLogin;

```
interface
USES Windows, fcButton, fcImgBtn, fcShapeBtn, StdCtrls, Graphics, Controls,
  ExtCtrls, Classes, Forms, sysutils, dialogs;

type
  TFrm_Login = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    L_pwd: TLabel;
    Edit_Pwd: TEdit;
    btn_Ok: TfcShapeBtn;
    btn_Cancela: TfcShapeBtn;
    Edit_Usu: TEdit;
    Label2: TLabel;
    P_Server: TPanel;
    L_desc: TLabel;
    Edit_servidor: TEdit;
    Edit_senha_server: TEdit;
    Label3: TLabel;
    L_Usu: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit_usuario_server: TEdit;
    procedure Edit_UsuKeyPress(Sender: TObject; var Key: Char);
    procedure btn_OkClick(Sender: TObject);
    procedure btn_CancelaClick(Sender: TObject);
    procedure Edit_PwdKeyPress(Sender: TObject; var Key: Char);
    procedure FormShow(Sender: TObject);
    procedure Edit_PwdKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Frm_Login: TFrm_Login;

implementation

uses uMain, uDm_Base, Funcoes;

{$R *.dfm}

procedure TFrm_Login.Edit_UsuKeyPress(Sender: TObject; var Key: Char);
begin
   If Key = #13 Then
   Begin
      Key := #0;
      SelectNext((Sender as Twincontrol),True,True);
   End;
end;
```

```
procedure TFrm_Login.btn_OkClick(Sender: TObject);
var
   wb_erro : boolean;
begin
   wb_Erro:= false;
   Edit_Usu.Text:= trim(LowerCase(Edit_Usu.Text));

   Dm_Base.my_conn.Connected    := False;
   Dm_Base.my_conn.Host         := Edit_servidor.Text;
   Dm_Base.my_conn.UserPassword := Edit_senha_server.Text;
   Dm_Base.my_conn.UserName     := Edit_usuario_server.Text;
   Dm_Base.my_conn.Connected    := true;

   if Edit_Usu.Text = '' then
   begin
      showmessage('Usuário não Digitado..');
      wb_Erro := True;
      Edit_Usu.SetFocus;
   end;

   if not wb_erro then
      frm_Main.Usuario.Carregar(Edit_Usu.Text);


   if not wb_Erro then
   begin
      if (frm_Main.Usuario.P_login <> Edit_Usu.Text) Then
      begin
         showmessage('Usuário não Localizado..');
         wb_Erro := True;
         Edit_Usu.SetFocus;
      end;
   end;

   if not wb_Erro then
   begin
      if (frm_Main.Usuario.P_senha <> Edit_Pwd.Text) Then
      begin
         showmessage('Senha não Confere..');
         wb_Erro := true;
         Edit_Pwd.SetFocus;
      end;
   end;

   if not wb_Erro then
   begin
      if (frm_Main.Usuario.P_status <> 1) Then
      begin
         showmessage('Usuário Inativo. Comunique suporte..');
         wb_Erro := true;
      end;
   end;

   if wb_Erro then
   begin
      Frm_Login.Tag   := (Frm_Login.Tag + 1);
      if Frm_Login.Tag = 3 then
         ModalResult  := mrCancel
      else
         Edit_Usu.SetFocus;

      Abort;
   end;

   if not wb_Erro then
   begin
      Frm_Main.RodaPe.Panels[2].Text := frm_Main.Usuario.P_login;

      edit_usu.Enabled  := False;
      l_usu.Enabled     := False;
      edit_pwd.Enabled  := False;
      l_pwd.Enabled     := False;
      btn_ok.Enabled    := False;

      ModalResult := mrOK;

   end;
```

```
end;

procedure TFrm_Login.btn_CancelaClick(Sender: TObject);
begin
   ModalResult  := mrCancel
end;

procedure TFrm_Login.Edit_PwdKeyPress(Sender: TObject; var Key: Char);
begin
   If Key = #13 Then
   Begin
      Key := #0;
      btn_OkClick(Self);
   End;
end;

procedure TFrm_Login.FormShow(Sender: TObject);
begin
   edit_usu.Enabled        := True;
   l_usu.Enabled           := True;
   edit_pwd.Enabled        := True;
   l_pwd.Enabled           := True;
   btn_ok.Enabled          := True;

   Edit_Usu.Text := UserName;
   Edit_Pwd.Text := '';

   if (trim(Edit_Usu.Text)) <> '' then
      ActiveControl := Edit_Pwd
   else
      ActiveControl := Edit_Usu;

end;

procedure TFrm_Login.Edit_PwdKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
   if key = VK_F10 then
   begin
      if p_server.Visible then
      begin
         L_desc.Visible   := true;
         p_server.Visible := false;
      end
      else
      begin
         L_desc.Visible   := false;
         p_server.Visible := true;
      end;
      key:= 0;
   end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.34 unit uLOrcamentoRecusado;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa, TLOGOS;

type
  Tfrm_LOrcamentoRecusado = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure rgp_tipoClick(Sender: TObject);
    procedure btn_IncluirClick(Sender: TObject);
  private
    procedure listar_lv(wp_tipo: integer);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_LOrcamentoRecusado : Tfrm_LOrcamentoRecusado;
  ws_id_os   : String;

implementation

uses uLOrcarAltera, uMain;

{$R *.dfm}

procedure Tfrm_LOrcamentoRecusado.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LOrcamentoRecusado.Release;
   frm_LOrcamentoRecusado := nil;
end;

procedure Tfrm_LOrcamentoRecusado.listar_lv(wp_tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(WP_TIPO,8);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
```

```
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         wl_ItemList.SubItems.Add(floattostr(os_atu.P_valor_orcado));
      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_LOrcamentoRecusado.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar_lv(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LOrcamentoRecusado.btn_IncluirClick(Sender: TObject);
var
   os_atualiza   : TcOrdemDeServico;
   log           : Tclogos;
begin
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você Liberou este equipamento para
devolução!','Devolução de Equipamento',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'DEVOLVER';
         os_atualiza.Salvar(1);
         os_atualiza.freeonrelease;
         log               := Tclogos.Create(nil);
         log.P_id_os       := strtoint(lv.Selected.Caption);
         log.P_id_usuario  := frm_main.Usuario.P_id_usuario;
         log.P_acao        := 'LIBEROU EQUIPAMENTO PARA DEVOLUÇÃO';
         log.P_data        := now;
         log.Salvar(0);
         log.freeonrelease;
         rgp_tipoclick(self);
      end;
   end
   else
   showmessage('Selecione um registro');
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.35 unit uLOrcar;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa, TLOGOS;
type
  Tfrm_LOrcar = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure rgp_tipoClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure btn_IncluirClick(Sender: TObject);
  private
    procedure listar_lv(wp_tipo: integer);
    procedure Salva_log(wp_id_os: String);
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_LOrcar : Tfrm_LOrcar;
  ws_id_os   : String;
implementation
uses uLOrcarAltera, uMain;
{$R *.dfm}
procedure Tfrm_LOrcar.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LOrcar.Release;
   frm_LOrcar := nil;
end;

procedure Tfrm_LOrcar.listar_lv(wp_tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(WP_TIPO,6);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         wl_ItemList.SubItems.Add(floattostr(os_atu.P_valor_orcado));
      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
```

```
   os_cli.freeonrelease;
end;

procedure Tfrm_LOrcar.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar_lv(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LOrcar.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      ws_id_os         := lv.Selected.Caption;

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_LAltOrcar) then
            frm_LAltOrcar  := Tfrm_LAltOrcar.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;
         frm_LAltOrcar.e_numOS.Caption:= ws_id_os;
         frm_LAltOrcar.monta_item_os;
         frm_LAltOrcar.e_descpessoa.Text:= lv.Selected.SubItems[0];
         frm_LAltOrcar.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LOrcar.btn_IncluirClick(Sender: TObject);
var
   os_atualiza   : TcOrdemDeServico;
begin
   if wb_selecionoulinha then
   begin
      if Application.MessageBox('Tem certeza que você deseja Completar o
Orçamento!','Liberação de Orçamento',mb_YesNo) = IdYes then
      begin
         os_atualiza                    := TcOrdemDeServico.Create(nil);
         os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
         os_atualiza.P_status           := 'ORÇADO';
         os_atualiza.Salvar(1);
         os_atualiza.freeonrelease;
         rgp_tipoclick(self);
         salva_log(lv.Selected.Caption);
      end;
   end
   else
   showmessage('Selecione um registro');

end;

procedure Tfrm_LOrcar.Salva_log(wp_id_os: String);
var
   log   : Tclogos;
begin
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(wp_id_os);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'COMPLETOU O ORÇAMENTO';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.36 unit uLOrcarAltera;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, StdCtrls, Mask, ToolEdit, CurrEdit,
  fcButton, fcImgBtn, fcShapeBtn, ExtCtrls, ComCtrls, TOrdemDeServico, TPeca, Tpessoa,
  JvExStdCtrls, JvMemo, JvExExtCtrls, JvComponent, JvCaptionPanel,Tlogos;
type
  Tfrm_LAltOrcar = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    e_descpessoa: TEdit;
    Label2: TLabel;
    e_idpessoa: TCurrencyEdit;
    Panel6: TPanel;
    p_dados: TPanel;
    p_desc: TPanel;
    GroupBox3: TGroupBox;
    Edit_desc_equip: TMemo;
    GroupBox4: TGroupBox;
    Edit_desc_prob: TMemo;
    Panel5: TPanel;
    GroupBox1: TGroupBox;
    lv_Pessoa: TListView;
    p_csitens: TJvCaptionPanel;
    Panel4: TPanel;
    Label6: TLabel;
    Label8: TLabel;
    btn_pesq: TfcShapeBtn;
    desc_peca: TEdit;
    GroupBox2: TGroupBox;
    edit_desc_atu: TJvMemo;
    edit_qtdepeca: TCurrencyEdit;
    fcShapeBtn2: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    id_peca: TCurrencyEdit;
    P_Valores: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    GroupBox5: TGroupBox;
    e_numOS: TLabel;
    RB_VALORES: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    edit_valorOrc: TCurrencyEdit;
    edit_valorMObra: TCurrencyEdit;
    GroupBox6: TGroupBox;
    Panel7: TPanel;
    lv: TListView;
    Panel8: TPanel;
    btn_alt_sit: TfcShapeBtn;
    btn_exc_sit: TfcShapeBtn;
    btn_inc_sit: TfcShapeBtn;
    Label9: TLabel;
    e_totpecas: TCurrencyEdit;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
    procedure btn_excluirClick(Sender: TObject);
    procedure p_dadosEnter(Sender: TObject);
```

```
    procedure Panel3Enter(Sender: TObject);
    procedure lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure btn_alt_sitClick(Sender: TObject);
    procedure btn_exc_sitClick(Sender: TObject);
    procedure btn_inc_sitClick(Sender: TObject);
    procedure fcShapeBtn2Click(Sender: TObject);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
    procedure p_csitensButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
    procedure edit_valorMObraExit(Sender: TObject);
    procedure btn_FechaClick(Sender: TObject);
  private
    procedure monta_lvequip(wp_iddoItem: integer);
    procedure salva_dados_os;
    procedure limpar_p_csitens;
    procedure Salvar_SItem_OS;
    procedure Valor_Orcamento;
    procedure Salva_log;
    { Private declarations }
  public
    procedure monta_item_os;
    { Public declarations }
  end;
var
  frm_LAltOrcar  : Tfrm_LAltOrcar;
  wi_IDSubitem   : Integer;
  wi_IDitem      : Integer;
implementation
uses uPesqPecas, uLOrcar, uMain;
{$R *.dfm}
procedure Tfrm_LAltOrcar.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   frm_LAltOrcar.Release;
   frm_LAltOrcar:= nil;
end;
procedure Tfrm_LAltOrcar.btn_SalvaClick(Sender: TObject);
var
   os_abrindo   : TcOrdemDeServico;
begin
  inherited;
   if e_idpessoa.AsInteger <> 0 then
   begin
      os_abrindo                        := TcOrdemDeServico.Create(nil);
      os_abrindo.P_id_pessoa            := e_idpessoa.AsInteger;
      os_abrindo.P_data_alt             := now;
      os_abrindo.P_valor_orcado         := edit_valororc.AsInteger;
      os_abrindo.P_valor_mao_obra       := edit_valorMobra.AsInteger;
      os_abrindo.p_valor_pecas          := e_totpecas.AsInteger;
      os_abrindo.Salvar(0);
      salva_log;
   end;
end;

procedure Tfrm_LAltOrcar.btn_excluirClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este Item!','Exclusão',mb_YesNo)
= IdYes then
   begin
      os_SItem                                  := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := strtoint(e_numos.Caption);
      os_SItem.Items_OS.P_id_item_os            := wi_IDitem;
      os_SItem.Items_OS.P_id_os                 := strtoint(e_numos.Caption);
      //---> exclui o registro
      if wi_IDitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(3);
         os_SItem.Items_OS.Salvar(2);
         lv.Clear;
         monta_item_os;
```

```
         monta_lvequip(wi_IDitem);
         btn_inc_sit.Enabled  := False;
         btn_exc_sit.Enabled  := False;
         btn_alt_sit.Enabled  := False;
      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_LAltOrcar.monta_lvequip(wp_iddoItem : integer);
var
   pescontr    : TcOrdemDeServico;
   peca_sitem  : TcPeca;
   wl_ItemList : TListItem;
   wd_v_peca   : double;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);
   wd_v_peca:= 0;
   pescontr.Items_OS.SubItensOS.Seleciona(strtoint(e_numos.Caption),wp_iddoItem);
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_Sitem);
         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_id_peca));
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
            wd_v_peca:= peca_sitem.P_valor_unitario *
pescontr.Items_OS.SubItensOS.P_qtde_peca;
         end
         else
         begin
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wl_ItemList.SubItems.Add('');
            wd_v_peca:= 0;
         end;
         wl_ItemList.SubItems.Add(pescontr.Items_OS.SubItensOS.P_descricao);
         if wd_v_peca <> 0 then
            wl_ItemList.SubItems.Add(floattostr(wd_v_peca))
         else
            wl_ItemList.SubItems.Add('');


         pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_LAltOrcar.p_dadosEnter(Sender: TObject);
begin
  inherited;
    btn_inc_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
    btn_alt_sit.Enabled  := False;
end;

procedure Tfrm_LAltOrcar.Panel3Enter(Sender: TObject);
begin
  inherited;
  p_dados.Enabled:= false;
end;

procedure Tfrm_LAltOrcar.lv_PessoaMouseUp(Sender: TObject; Button: TMouseButton;  Shift:
TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
```

```
begin
  itemselecionado       := lv_Pessoa.GetItemAt(x, y);
  wi_IDitem             := 0;
  btn_inc_sit.Enabled   := False;
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
       wi_IDitem            := strtoint(lv_Pessoa.Selected.Caption);
       edit_desc_equip.Text := lv_Pessoa.Selected.SubItems[2];
       edit_desc_prob.Text  := lv_Pessoa.Selected.SubItems[3];
       btn_inc_sit.Enabled  := True;
    end
    else
    begin
       edit_desc_equip.Clear;
       edit_desc_prob.Clear;
    end;
  end
  else
  begin
    edit_desc_equip.Clear;
    edit_desc_prob.Clear;
  end;
  monta_lvequip(wi_IDitem);
end;

procedure Tfrm_LAltOrcar.monta_item_os;
var
   os_abrindo  : TcOrdemDeServico;
   wl_ItemList : TListItem;
   wi_cont     : Integer;
begin
   inherited;
   os_abrindo := TcOrdemDeServico.Create(nil);
   os_abrindo.Selecionar(strtoint(e_numos.Caption));
   edit_valorOrc.Value  := os_abrindo.P_valor_orcado;
   e_totpecas.Value     := os_abrindo.P_valor_pecas;
   edit_valorMObra.Value:= os_abrindo.P_valor_mao_obra;
   os_abrindo.Items_OS.Seleciona(strtoint(e_numos.Caption));
   os_abrindo.Items_OS.Primeiro;
   lv_Pessoa.Items.Clear;
   repeat
      os_abrindo.Items_OS.Coloca;
      if os_abrindo.Items_OS.P_id_item_os <> 0 then
      begin
         wl_ItemList            := lv_Pessoa.Items.Add;
         wl_ItemList.Caption    := inttostr(os_abrindo.Items_OS.P_id_item_os);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_contrato));
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_id_equipamento));
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_equip);
         wl_ItemList.SubItems.Add(os_abrindo.Items_OS.P_desc_prob);
         wl_ItemList.SubItems.Add(inttostr(os_abrindo.Items_OS.P_tipo_atendimento));

         os_abrindo.Items_OS.proximo;
      end;
   until os_abrindo.Items_OS.final;
   os_abrindo.FreeOnRelease;
end;

procedure Tfrm_LAltOrcar.lvMouseUp(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var
  itemselecionado : TListItem;
begin
  itemselecionado := lv.GetItemAt(x, y);
  if (itemselecionado <> nil) then
  begin
    if (itemselecionado.Selected) then
    begin
      wi_IDSubitem         := strtoint(lv.Selected.Caption);
      btn_alt_sit.Enabled  := True;
      btn_exc_sit.Enabled  := True;
    end
    else
    begin
       btn_alt_sit.Enabled  := False;
```

```
      btn_exc_sit.Enabled  := False;
      wi_IDSubitem         := 0;
    end;
  end
  else
  begin
    btn_alt_sit.Enabled  := False;
    btn_exc_sit.Enabled  := False;
  end;

end;

procedure Tfrm_LAltOrcar.btn_alt_sitClick(Sender: TObject);
var
   os_SItem     : TcOrdemDeServico;
   peca_sitem   : TcPeca;
begin
   limpar_p_csitens;
   p_csitens.Left:= (frm_LAltOrcar.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top := (frm_LAltOrcar.Height div 2) - (p_csitens.Height div 2);
   Panel2.Enabled:= False;
   os_SItem      := TcOrdemDeServico.Create(nil);
   peca_sitem    := TcPeca.Create(nil);

os_SItem.Items_OS.SubItensOS.Selecionar(strtoint(e_numos.Caption),wi_IDitem,wi_IDSubitem
);
   id_peca.AsInteger          := os_SItem.Items_OS.SubItensOS.P_id_peca;
   edit_qtdepeca.AsInteger    := os_SItem.Items_OS.SubItensOS.P_qtde_peca;
   edit_desc_atu.Text         := os_SItem.Items_OS.SubItensOS.P_descricao;
   peca_sitem.Selecionar(id_peca.AsInteger);
   desc_peca.Text:= peca_sitem.P_descricao;
   peca_sitem.FreeOnRelease;
   os_SItem.FreeOnRelease;
   edit_qtdepeca.enabled := True;
   p_csitens.Visible := True;
end;

procedure Tfrm_LAltOrcar.btn_exc_sitClick(Sender: TObject);
var
   os_SItem   : TcOrdemDeServico;
begin
   if Application.MessageBox('Deseja realmente Excluir este
movimento!','Exclusão',mb_YesNo) = IdYes then
   begin
      os_SItem                                := TcOrdemDeServico.Create(nil);
      os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
      os_SItem.Items_OS.SubItensOS.P_id_os      := strtoint(e_numos.Caption);
      os_SItem.Items_OS.SubItensOS.P_id_Sitem   := wi_IDSubitem;
      //---> exclui o registro
      if wi_IDSubitem <> 0 then
      begin
         os_SItem.Items_OS.SubItensOS.Salvar(2);
         monta_lvequip(wi_IDitem);
         Valor_Orcamento;
         salva_dados_os;
         btn_fechaatuclick(self);
      end;
      os_SItem.freeonrelease;
   end;
end;

procedure Tfrm_LAltOrcar.btn_inc_sitClick(Sender: TObject);
begin
  inherited;
   limpar_p_csitens;
   p_csitens.Left    := (frm_LAltOrcar.Width div 2) - (p_csitens.Width div 2);
   p_csitens.Top     := (frm_LAltOrcar.Height div 2) - (p_csitens.Height div 2);
   wi_IDSubitem      := 0;
   Panel2.Enabled    := False;
   p_csitens.Visible := TRUE;
end;

procedure Tfrm_LAltOrcar.fcShapeBtn2Click(Sender: TObject);
begin
  inherited;
   Valor_Orcamento;
   if edit_desc_atu.Text <> '' then
```

```
    begin
       if id_peca.AsInteger <> 0 then
       begin
          if edit_qtdepeca.AsInteger <> 0 then
          begin
             Salvar_SItem_OS;
             monta_lvequip(wi_IDitem);
             Valor_Orcamento;
             salva_dados_os;
             salva_log;
             btn_fechaatuclick(self);
          end
          else
          begin
             Showmessage('Preencha a quantidade de Peças');
          end;
       end
       else
       begin
          Salvar_SItem_OS;
          monta_lvequip(wi_IDitem);
          Valor_Orcamento;
          salva_dados_os;
          btn_fechaatuclick(self);
       end;
    end
    else
       Showmessage('Preencha a Descrição');
    Valor_Orcamento;
end;

procedure Tfrm_LAltOrcar.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
    p_csitens.Visible:= False;
    Panel2.Enabled   := True;
end;

procedure Tfrm_LAltOrcar.btn_pesqClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqPecas) then
         frm_PesqPecas  := Tfrm_PesqPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;
      frm_PesqPecas.Tag := 3;
      frm_PesqPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LAltOrcar.p_csitensButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      p_csitens.Visible:= False;
      Panel2.Enabled   := True;
   end;
end;

procedure Tfrm_LAltOrcar.salva_dados_os;
var
   os_atualiza   : TcOrdemDeServico;
begin
   os_atualiza                    := TcOrdemDeServico.Create(nil);
   os_atualiza.Selecionar(strtoint(e_numOS.Caption));
   os_atualiza.P_valor_orcado     := edit_valorOrc.Value;
   os_atualiza.P_valor_pecas      := e_totpecas.Value;
   os_atualiza.P_valor_mao_obra   := edit_valorMObra.Value;
```

```
   os_atualiza.Salvar(1);
   os_atualiza.freeonrelease;
end;

procedure Tfrm_LAltOrcar.limpar_p_csitens;
begin
   desc_peca.Clear;
   edit_qtdepeca.Clear;
   id_peca.Clear;
   edit_desc_atu.Clear;
   edit_qtdepeca.Enabled:= false;
end;

procedure Tfrm_LAltOrcar.Salvar_SItem_OS;
var
   os_SItem   : TcOrdemDeServico;
begin
   os_SItem                                    := TcOrdemDeServico.Create(nil);
   os_SItem.Items_OS.SubItensOS.P_id_item_os := wi_IDitem;
   os_SItem.Items_OS.SubItensOS.P_id_os       := strtoint(e_numOS.Caption);
   os_SItem.Items_OS.SubItensOS.P_id_Sitem    := wi_IDSubitem;
   os_SItem.Items_OS.SubItensOS.P_id_peca     := id_peca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_qtde_peca   := edit_qtdepeca.AsInteger;
   os_SItem.Items_OS.SubItensOS.P_descricao   := edit_desc_atu.Text;

   //---> salva o registro
   if wi_IDSubitem = 0 then
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(0);
   end
   else
   begin
      os_SItem.Items_OS.SubItensOS.Salvar(1);
   end;
   os_SItem.freeonrelease;
end;

procedure Tfrm_LAltOrcar.edit_valorMObraExit(Sender: TObject);
begin
   Valor_Orcamento;
   salva_dados_os;
   Valor_Orcamento;
end;

procedure Tfrm_LAltOrcar.Valor_Orcamento;
var
  pescontr        : TcOrdemDeServico;
begin
   pescontr:= TcOrdemDeServico.Create(nil);
   edit_valororc.Value  :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1) +
edit_valorMObra.Value;
   e_totpecas.Value     :=
pescontr.Items_OS.SubItensOS.Valor_total_pecas_os(strtoint(e_numOS.Caption),1);
   pescontr.freeonrelease;
end;

procedure Tfrm_LAltOrcar.btn_FechaClick(Sender: TObject);
begin
   inherited;
   frm_LOrcar.rgp_tipo.ItemIndex:= 0;
   frm_LOrcar.rgp_tipoClick(self);
end;

procedure Tfrm_LAltOrcar.Salva_log;
var
   log   : Tclogos;
begin
   log              := Tclogos.Create(nil);
   log.P_id_os      := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao       := 'ALTEROU ORÇAMENTO';
   log.P_data       := now;
   log.Salvar(0);
   log.freeonrelease;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.37 unit uLOS_Atualiza;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa;

type
  Tfrm_LOS_Atualiza = class(Tfrm_ListaPadrao)
    procedure rgp_tipoClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
     procedure listar(wp_tipo : integer);
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_LOS_Atualiza: Tfrm_LOS_Atualiza;
  wi_IDOS         : Integer;

implementation

uses uCOS_Atualizar, Funcoes;

{$R *.dfm}

procedure Tfrm_LOS_Atualiza.listar(wp_tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(wp_tipo,0);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(Replacestr(os_atu.P_status,'APROVADA - ',''));
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
```

```
end;

procedure Tfrm_LOS_Atualiza.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LOS_Atualiza.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      LockWindowUpDate(Handle);
      try
         if not assigned(frm_COS_Atualizar) then
            frm_COS_Atualizar  := Tfrm_COS_Atualizar.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;
         frm_COS_Atualizar.e_numOS.Caption   := lv.Selected.Caption;
         uCOS_Atualizar.wi_id_os             := strtoint(lv.Selected.Caption);
         frm_COS_Atualizar.e_descpessoa.Text := lv.Selected.SubItems[0];
         frm_COS_Atualizar.Mostra_itens_OS;
         frm_COS_Atualizar.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LOS_Atualiza.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LOS_Atualiza.Release;
   frm_LOS_Atualiza:= nil;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.38 unit uLOS_Cancelar;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa, JvPanel, JvExExtCtrls,
  JvComponent, JvCaptionPanel, tPeca,Tcaixa, Tlogos;
type
  Tfrm_LCancelaOS = class(Tfrm_ListaPadrao)
    P_Cancelamento: TJvCaptionPanel;
    JvPanel2: TJvPanel;
    lv_pecas: TListView;
    BTN_CANCELAR: TfcShapeBtn;
    btn_fechaatu: TfcShapeBtn;
    GroupBox1: TGroupBox;
    e_numOS: TLabel;
    procedure rgp_tipoClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_AlteraClick(Sender: TObject);
    procedure P_CancelamentoButtonClick(Sender: TObject;
      Button: TJvCapBtnStyle);
    procedure btn_fechaatuClick(Sender: TObject);
    procedure BTN_CANCELARClick(Sender: TObject);
  private
    procedure listar(Estado,Tipo: integer);
    procedure carrega_pecas_os;
    procedure Cancela_os;
    procedure reapropriar_estoque;
    procedure extorna_caixa;
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_LCancelaOS: Tfrm_LCancelaOS;
  WS_STATS      : String;
implementation
uses Funcoes, uMain;
{$R *.dfm}
procedure Tfrm_LCancelaOS.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   case rgp_tipo.ItemIndex of
      0 : listar(0,2);  // canceladas
      1 : listar(6,4);  // mostra todas
      2 : listar(0,5);  // orçamentos
      3 : listar(0,0);  // aprovadas
      4 : listar(2,4);  // em andamento
      5 : listar(3,4);  // aguardando peças
      6 : listar(4,4);  // liberadas
      7 : listar(2,3);  // fechadas
      8 : listar(0,6);  // Orçar
      9 : listar(0,7);  // Orçado
      10: listar(0,8);  // Recusada
      11: listar(3,3);  // Devolver
   end;
end;
```

```
procedure Tfrm_LCancelaOS.listar(Estado,Tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   os_atu.Pesquisar(Estado,Tipo);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(Replacestr(os_atu.P_status,'APROVADA - ',''));
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         if os_atu.P_data_fechou <> 0 then
            wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_fechou))
         else
            wl_ItemList.SubItems.Add('');
         if os_atu.P_data_canc <> 0 then
            wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_canc))
         else
            wl_ItemList.SubItems.Add('');

      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_LCancelaOS.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LCancelaOS.Release;
   frm_LCancelaOS:= nil;
end;

procedure Tfrm_LCancelaOS.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
   ws_stats:= lv.Selected.SubItems[1];
      if NOT (
         (ws_stats = 'ORÇAMENTO')   or
         (ws_stats = 'ORÇAR')       or
         (ws_stats = 'ORÇADO')  )  then
      begin
         if (ws_stats <> 'CANCELADA') then
         begin
            Screen.Cursor    := crHourGlass;
            LockWindowUpDate(Handle);
            try
               e_numos.Caption         := lv.Selected.Caption;
               P_Cancelamento.Left     := (frm_LCancelaOS.Width div 2)  -
(P_Cancelamento.Width  div 2);
               P_Cancelamento.Top      := (frm_LCancelaOS.Height div 2) -
(P_Cancelamento.Height div 2);
               carrega_pecas_os;
               Panel2.Enabled          := False;
               Panel1.Enabled          := False;
               P_Cancelamento.Visible  := True;
            finally
               LockWindowUpdate(0);
               Screen.Cursor := crDefault;
            end;
         end
         else
            showmessage('Esta OS já Foi Cancelada');
```

```
      end
      else
      begin
         if Application.MessageBox('Tem certeza que você deseja cancelar esta
OS!','Cancelar OS',mb_YesNo) = IdYes then
         begin
            e_numos.Caption         := lv.Selected.Caption;
            Cancela_os;
            ws_stats                := '';
            rgp_tipoclick(self);
            btn_fechaatuClick(self);
         end;
      end;
   end
   else
   showmessage('Selecione um registro');
end;

procedure Tfrm_LCancelaOS.carrega_pecas_os;
var
   peca_sitem  : TcPeca;
   pescontr    : TcOrdemDeServico;
   wl_ItemList : TListItem;
begin
   peca_sitem    := TcPeca.Create(nil);
   pescontr      := TcOrdemDeServico.Create(nil);
   pescontr.Items_OS.SubItensOS.SelecionarPecas(strtoint(e_numos.Caption));
   pescontr.Items_OS.SubItensOS.Primeiro;
   lv_pecas.Items.Clear;
   repeat
      pescontr.Items_OS.SubItensOS.Coloca;
      if pescontr.Items_OS.SubItensOS.P_id_Sitem <> 0 then
      begin

         if pescontr.Items_OS.SubItensOS.P_id_peca <> 0 then
         begin
            wl_ItemList            := lv_pecas.Items.Add;
            wl_ItemList.Caption    := inttostr(pescontr.Items_OS.SubItensOS.P_id_peca);
            peca_sitem.Selecionar(pescontr.Items_OS.SubItensOS.P_id_peca);
            wl_ItemList.SubItems.Add(peca_sitem.P_descricao);

wl_ItemList.SubItems.Add(inttostr(pescontr.Items_OS.SubItensOS.P_qtde_peca));
         end;

         pescontr.Items_OS.SubItensOS.proximo;
      end;
   until pescontr.Items_OS.SubItensOS.final;
   pescontr.freeonrelease;
   peca_sitem.FreeOnRelease;
end;

procedure Tfrm_LCancelaOS.P_CancelamentoButtonClick(Sender: TObject;
  Button: TJvCapBtnStyle);
begin
  inherited;
   IF Button = capClose then
   begin
      btn_fechaatuClick(self);
   end;
end;

procedure Tfrm_LCancelaOS.btn_fechaatuClick(Sender: TObject);
begin
  inherited;
   P_Cancelamento.Visible  := False;
   Panel2.Enabled          := True;
   Panel1.Enabled          := True;
   E_NUMOS.Caption         := '0';
end;

procedure Tfrm_LCancelaOS.BTN_CANCELARClick(Sender: TObject);
begin
  inherited;
   if Application.MessageBox('Tem certeza que você deseja cancelar esta OS!','Cancelar
OS',mb_YesNo) = IdYes then
   begin
      Cancela_os;
```

```
      reapropriar_estoque;
      if ws_stats = 'FECHADA' THEN
      begin
         extorna_caixa;
      end;
      rgp_tipoclick(self);
      btn_fechaatuClick(self);
   end;
end;

procedure Tfrm_LCancelaOS.reapropriar_estoque;
var
   estoque     : TCPeca;
   OS          : TcOrdemDeServico;
begin
   OS             := TcOrdemDeServico.Create(nil);
   os.Items_OS.SubItensOS.SelecionarPecas(strtoint(e_numOS.Caption));
   os.Items_OS.SubItensOS.Primeiro;
   repeat
      os.Items_OS.SubItensOS.Coloca;
      estoque := TCPeca.Create(nil);
      estoque.Selecionar(os.Items_OS.SubItensOS.P_id_peca);
      estoque.P_quantidade:= (estoque.P_quantidade +
os.Items_OS.SubItensOS.P_qtde_peca);
      estoque.Salvar(1);
      estoque.FreeOnRelease;
      os.Items_OS.SubItensOS.proximo;
   until os.Items_OS.SubItensOS.final;
   os.FreeOnRelease;
end;

procedure Tfrm_LCancelaOS.Cancela_os;
var
   os_atualiza   : TcOrdemDeServico;
   log   : Tclogos;
begin
   os_atualiza                    := TcOrdemDeServico.Create(nil);
   os_atualiza.Selecionar(strtoint(e_numOS.Caption));
   os_atualiza.P_status           := 'CANCELADA';
   os_atualiza.P_data_canc        := now;
   os_atualiza.Salvar(1);
   os_atualiza.freeonrelease;
   log               := Tclogos.Create(nil);
   log.P_id_os       := strtoint(e_numOS.Caption);
   log.P_id_usuario := frm_main.Usuario.P_id_usuario;
   log.P_acao        := 'CANCELOU A OS';
   log.P_data        := now;
   log.Salvar(0);
   log.freeonrelease;
end;

procedure Tfrm_LCancelaOS.extorna_caixa;
var
   caixa   : Tccaixa;
   caixa2  : Tccaixa;
begin
   caixa                    := Tccaixa.Create(nil);
   caixa.SelecionarPeloDocumento(strtoint(e_numOS.Caption));
   caixa.Primeiro;
   REPEAT
      if caixa.P_id_lcto <> 0 then
      begin
         caixa.Coloca;
         caixa2 := Tccaixa.Create(nil);
         caixa2.Selecionar(caixa.P_id_lcto);
         caixa2.P_situacao := 'C';
         caixa2.Salvar(1);
         caixa2.freeonrelease;
      end;
      CAIXA.Proximo;
   UNTIL caixa.Final;
   caixa.freeonrelease;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.39 unit uLOS_Fechar;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TOrdemDeServico, TPessoa, Tlogos;
type
  Tfrm_LOS_Fechar = class(Tfrm_ListaPadrao)
    procedure btn_AlteraClick(Sender: TObject);
    procedure rgp_tipoClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    procedure listar(wp_tipo: integer);
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_LOS_Fechar: Tfrm_LOS_Fechar;
implementation

uses uCOS_Fechar, Funcoes, uMain;

{$R *.dfm}

procedure Tfrm_LOS_Fechar.rgp_tipoClick(Sender: TObject);
begin
  inherited;
   listar(rgp_tipo.ItemIndex);
end;

procedure Tfrm_LOS_Fechar.listar(wp_tipo: integer);
var
   os_atu      : TcOrdemDeServico;
   os_cli      : TcPessoa;
   wl_ItemList : TListItem;
begin
   os_atu   := TcOrdemDeServico.Create(nil);
   os_cli   := TcPessoa.Create(nil);
   if wp_tipo = 0 then
      os_atu.Pesquisar(wp_tipo,9)
   else
      os_atu.Pesquisar(wp_tipo,3);
   os_atu.Primeiro;
   lv.Items.Clear;
   repeat
      os_atu.Coloca;
      if os_atu.P_id_os <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(os_atu.P_id_os);
         os_cli.Selecionar(os_atu.P_id_pessoa,1);
         wl_ItemList.SubItems.Add(os_cli.P_nome);
         wl_ItemList.SubItems.Add(Replacestr(os_atu.P_status,'APROVADA - ',''));
         wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_cad));
         if os_atu.P_data_fechou <> 0 then
            wl_ItemList.SubItems.Add(datetostr(os_atu.P_data_fechou))
         else
```

```
            wl_ItemList.SubItems.Add('');
      end;
      os_atu.proximo;
   until os_atu.final;
   os_atu.freeonrelease;
   os_cli.freeonrelease;
end;

procedure Tfrm_LOS_Fechar.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LOS_Fechar.Release;
   frm_LOS_Fechar:= nil;
end;

procedure Tfrm_LOS_Fechar.btn_AlteraClick(Sender: TObject);
var
   os_atualiza   : TcOrdemDeServico;
   log           : Tclogos;
begin
  inherited;
   if wb_selecionoulinha then
   begin
      if lv.Selected.SubItems[1] <> 'FECHADA' then
      begin
         if lv.Selected.SubItems[1] <> 'DEVOLVER' then
         begin
            Screen.Cursor    := crHourGlass;
            LockWindowUpDate(Handle);
            try
               if not assigned(frm_COS_Fechar) then
                  frm_COS_Fechar  := Tfrm_COS_Fechar.Create(Self);

               // Seta Cursor normal
               Screen.Cursor := crDefault;
               frm_COS_Fechar.e_numOS.Caption   := lv.Selected.Caption;
               frm_COS_Fechar.e_descpessoa.Text := lv.Selected.SubItems[0];
               frm_COS_Fechar.Mostra_items_OS;
               frm_COS_Fechar.show;
            finally
               LockWindowUpdate(0);
               Screen.Cursor := crDefault;
            end;
         end
         else
         begin
            if Application.MessageBox('Tem certeza que você Devolveu todos os
equipamentos desta OS!','Devolução de Equipamento com Orçamento RECUSADO',mb_YesNo) =
IdYes then
            begin
               os_atualiza                    := TcOrdemDeServico.Create(nil);
               os_atualiza.Selecionar(strtoint(lv.Selected.Caption));
               os_atualiza.P_status           := 'FECHADA';
               os_atualiza.P_data_fechou      := now;
               os_atualiza.Salvar(1);
               os_atualiza.freeonrelease;
               log              := Tclogos.Create(nil);
               log.P_id_os      := strtoint(lv.Selected.Caption);
               log.P_id_usuario := frm_main.Usuario.P_id_usuario;
               log.P_acao       := 'DEVOLVEU OS EQUIPAMENTOS DA OS';
               log.P_data       := now;
               log.Salvar(0);
               log.freeonrelease;
               rgp_tipoclick(self);
            end;
         end;
      end
      else
         showmessage('Esta OS já Foi Fechada');
   end
   else
   showmessage('Selecione um registro');
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.40 unit uLPecas;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TPeca, RXSpin, RXCtrls;

type
  Tfrm_LPecas = class(Tfrm_ListaPadrao)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    editUF: TComboBox;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure editUFChange(Sender: TObject);
  private
    procedure Montartpcs;
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LPecas   : Tfrm_LPecas;
  wi_IDdapeca  : Integer;
  wi_IDTdapeca : Integer;

implementation

uses uCPecas, Funcoes;

{$R *.dfm}

procedure Tfrm_LPecas.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LPecas.Release;
   frm_LPecas:= nil;
end;

procedure Tfrm_LPecas.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDdapeca      := 0;
   wi_IDTdapeca     := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CPecas) then
         frm_CPecas  := Tfrm_CPecas.Create(Self);
```

```
      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LPecas.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor := crHourGlass;
      wi_IDdapeca   := strtoint(lv.Selected.Caption);
      wi_IDTdapeca  := strtoint(lv.Selected.SubItems[0]);
      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CPecas) then
            frm_CPecas  := Tfrm_CPecas.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CPecas.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LPecas.monta_lv;
var
   listpeca    : TcPeca;
   wl_ItemList : TListItem;
   wi_tipo     : Integer;
begin
   listpeca:= TcPeca.Create(nil);
   if editUF.Text <> '' then
      wi_tipo:= strtoint(copy(editUF.Text,1,4))
   else
      wi_tipo:= 0;
   listpeca.Listar(wi_tipo);
   listpeca.Primeiro;
   lv.Items.Clear;
   repeat
      listpeca.Coloca;
      if listpeca.P_id_peca <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(listpeca.P_id_peca);
         listpeca.Tipo_da_Peca.Selecionar(listpeca.P_id_tipo_peca);
         wl_ItemList.SubItems.Add(inttostr(listpeca.Tipo_da_Peca.P_id_tipo_peca));
         wl_ItemList.SubItems.Add(listpeca.Tipo_da_Peca.P_descricao);
         wl_ItemList.SubItems.Add(listpeca.P_descricao);
         wl_ItemList.SubItems.Add(inttostr(listpeca.P_quantidade));
         wl_ItemList.SubItems.Add(floattostr(listpeca.P_valor_unitario));
      end;
      listpeca.proximo;
   until listpeca.final;
   listpeca.freeonrelease;
end;

procedure Tfrm_LPecas.FormShow(Sender: TObject);
begin
  inherited;
   Montartpcs;
   monta_lv;
end;
```

```
procedure Tfrm_LPecas.Montartpcs;
var
   tpopec : TcPeca;
begin
   tpopec:= TcPeca.Create(nil);
   tpopec.Tipo_da_Peca.Seleciona;
   tpopec.Tipo_da_Peca.Primeiro;
   Edituf.Items.Clear;
   repeat
      tpopec.Tipo_da_Peca.Coloca;
      Edituf.Items.Add((ZerosEsq(inttostr(tpopec.Tipo_da_Peca.P_id_tipo_peca),4) + '-' +
tpopec.Tipo_da_Peca.P_descricao));
      tpopec.Tipo_da_Peca.proximo;
   until tpopec.Tipo_da_Peca.final;
   freeandnil(tpopec);
end;

procedure Tfrm_LPecas.editUFChange(Sender: TObject);
begin
  inherited;
   monta_lv;
   activecontrol:= lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.41 unit uLTDocumentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TTipoDocumento, RXSpin, RXCtrls;

type
  Tfrm_LTDocumentos = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LTDocumentos: Tfrm_LTDocumentos;
  wi_IDTDoc : Integer;

implementation

uses uCTDocumentos;

{$R *.dfm}

procedure Tfrm_LTDocumentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LTDocumentos.Release;
   frm_LTDocumentos:= nil;
end;

procedure Tfrm_LTDocumentos.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDTDoc        := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CTDocumentos) then
         frm_CTDocumentos  := Tfrm_CTDocumentos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CTDocumentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LTDocumentos.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDTDoc        := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CTDocumentos) then
            frm_CTDocumentos  := Tfrm_CTDocumentos.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CTDocumentos.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LTDocumentos.monta_lv;
var
   Tipodocs  : TCTipoDocumento;
   wl_ItemList : TListItem;
begin
   Tipodocs:= TCTipoDocumento.Create(nil);
   Tipodocs.Seleciona;
   Tipodocs.Primeiro;
   lv.Items.Clear;
   repeat
      Tipodocs.Coloca;
      if Tipodocs.P_id_tipo_doc <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Tipodocs.P_id_tipo_doc);
         wl_ItemList.SubItems.Add(Tipodocs.P_descricao);
      end;
      Tipodocs.proximo;
   until Tipodocs.final;
   Tipodocs.freeonrelease;
end;

procedure Tfrm_LTDocumentos.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.42 unit uLTEquipamentos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TTipoDeEquipamento, RXSpin, RXCtrls;

type
  Tfrm_LTEquipamentos = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LTEquipamentos: Tfrm_LTEquipamentos;
  wi_IDTEquip : Integer;

implementation

uses uCTEquipamentos;

{$R *.dfm}

procedure Tfrm_LTEquipamentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LTEquipamentos.Release;
   frm_LTEquipamentos:= nil;
end;

procedure Tfrm_LTEquipamentos.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDTEquip        := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CTEquipamentos) then
         frm_CTEquipamentos  := Tfrm_CTEquipamentos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CTEquipamentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LTEquipamentos.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDTEquip        := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CTEquipamentos) then
            frm_CTEquipamentos  := Tfrm_CTEquipamentos.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CTEquipamentos.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LTEquipamentos.monta_lv;
var
   TipoEquip  : TcTipoDeEquipamento;
   wl_ItemList : TListItem;
begin
   TipoEquip:= TcTipoDeEquipamento.Create(nil);
   TipoEquip.Seleciona;
   TipoEquip.Primeiro;
   lv.Items.Clear;
   repeat
      TipoEquip.Coloca;
      if TipoEquip.P_id_tipo_equipamento <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(TipoEquip.P_id_tipo_equipamento);
         wl_ItemList.SubItems.Add(TipoEquip.P_descricao);
      end;
      TipoEquip.proximo;
   until TipoEquip.final;
   TipoEquip.freeonrelease;
end;

procedure Tfrm_LTEquipamentos.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.43 unit uLTPecas;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, ExtCtrls, fcButton,
  fcImgBtn, fcShapeBtn, TTipodePeca, RXSpin, RXCtrls;

type
  Tfrm_LTPecas = class(Tfrm_ListaPadrao)
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
     procedure monta_lv;
    { Public declarations }
  end;

var
  frm_LTPecas: Tfrm_LTPecas;
  wi_IDTpeca : Integer;

implementation

uses uCTPecas;

{$R *.dfm}

procedure Tfrm_LTPecas.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LTPecas.Release;
   frm_LTPecas:= nil;
end;

procedure Tfrm_LTPecas.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDTpeca       := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CTPecas) then
         frm_CTPecas  := Tfrm_CTPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CTPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
```

```
   end;
end;

procedure Tfrm_LTPecas.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor     := crHourGlass;
      wi_IDTpeca        := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CTPecas) then
            frm_CTPecas  := Tfrm_CTPecas.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CTPecas.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LTPecas.monta_lv;
var
   TipoPeca  : TCTipodePeca;
   wl_ItemList : TListItem;
begin
   TipoPeca:= TCTipodePeca.Create(nil);
   TipoPeca.Seleciona;
   TipoPeca.Primeiro;
   lv.Items.Clear;
   repeat
      TipoPeca.Coloca;
      if TipoPeca.P_id_tipo_peca <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(TipoPeca.P_id_tipo_peca);
         wl_ItemList.SubItems.Add(TipoPeca.P_descricao);
      end;
      TipoPeca.proximo;
   until TipoPeca.final;
   TipoPeca.freeonrelease;
end;

procedure Tfrm_LTPecas.FormShow(Sender: TObject);
begin
  inherited;
   monta_lv;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.44 unit uLUsuarios;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTela_ListaPadrao, ComCtrls, StdCtrls, RXSpin, RXCtrls,
  ExtCtrls, fcButton, fcImgBtn, fcShapeBtn, TUsuario;

type
  Tfrm_LUsuario = class(Tfrm_ListaPadrao)
    procedure FormShow(Sender: TObject);
    procedure rgp_tipoClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_IncluirClick(Sender: TObject);
    procedure btn_AlteraClick(Sender: TObject);
    procedure btn_DesativarClick(Sender: TObject);
    procedure btn_AtivarClick(Sender: TObject);
  private
     procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_LUsuario: Tfrm_LUsuario;
  Usuariolist : TCUsuario;
  wi_IDUsu    : Integer;
implementation

uses uCUsuarios;

{$R *.dfm}

procedure Tfrm_LUsuario.FormShow(Sender: TObject);
begin
  inherited;
  Usuariolist:= tcusuario.Create(frm_lusuario);
  rgp_tipo.ItemIndex:= 0;
  rgp_tipoclick(Self);
end;

procedure Tfrm_LUsuario.listar_lv;
var
   wl_ItemList : TListItem;
begin
   Usuariolist.Primeiro;
   lv.Items.Clear;
   repeat
      Usuariolist.Coloca;
      if Usuariolist.P_id_usuario <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Usuariolist.P_id_usuario);
         wl_ItemList.SubItems.Add(Usuariolist.P_login);
         Usuariolist.Grupo.Selecionar(Usuariolist.P_id_grupo);
```

```
         wl_ItemList.SubItems.Add(Usuariolist.Grupo.P_descricao);
         wl_ItemList.SubItems.Add(Usuariolist.P_ramal);
         wl_ItemList.SubItems.Add(Usuariolist.P_email);
         wl_ItemList.SubItems.Add(datetostr(Usuariolist.P_datacadastro));
         wl_ItemList.SubItems.Add(datetostr(Usuariolist.P_dataultimoacesso));
      end;
      Usuariolist.Proximo;
   until Usuariolist.Final;
end;

procedure Tfrm_LUsuario.rgp_tipoClick(Sender: TObject);
begin
  inherited;
      case rgp_Tipo.ItemIndex of    //para ativo,inativo, ambos
         0 : Usuariolist.SelecionaAtivos;
         1 : Usuariolist.SelecionaItivos;
         2 : Usuariolist.SelecionaTodos;
      end;
      listar_lv;
end;

procedure Tfrm_LUsuario.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  inherited;
   frm_LUsuario.Release;
   frm_LUsuario := nil;
end;

procedure Tfrm_LUsuario.btn_IncluirClick(Sender: TObject);
begin
  inherited;
   Screen.Cursor    := crHourGlass;
   wi_IDUsu         := 0;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_CUsuario) then
         frm_CUsuario  := Tfrm_CUsuario.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_CUsuario.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_LUsuario.btn_AlteraClick(Sender: TObject);
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      wi_IDUsu         := strtoint(lv.Selected.Caption);

      LockWindowUpDate(Handle);
      try
         if not assigned(frm_CUsuario) then
            frm_CUsuario  := Tfrm_CUsuario.Create(Self);

         // Seta Cursor normal
         Screen.Cursor := crDefault;

         frm_CUsuario.show;
      finally
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');

   wb_selecionoulinha:=false;
end;

procedure Tfrm_LUsuario.btn_DesativarClick(Sender: TObject);
```

```
var
   usuariodesativar : TCUsuario;

begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      LockWindowUpDate(Handle);
      try
        usuariodesativar := tcusuario.Create(nil);
        usuariodesativar.Desativar(strtoint(lv.Selected.Caption));
        rgp_tipoclick(self);
      finally
         usuariodesativar.FreeOnRelease;
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinha:=false;
end;

procedure Tfrm_LUsuario.btn_AtivarClick(Sender: TObject);
var
   usuarioativar : TCUsuario;
begin
  inherited;
   if wb_selecionoulinha then
   begin
      Screen.Cursor    := crHourGlass;
      LockWindowUpDate(Handle);
      try
        usuarioativar := tcusuario.Create(nil);
        usuarioativar.Ativar(strtoint(lv.Selected.Caption));
        rgp_tipoclick(self);
      finally
         usuarioativar.FreeOnRelease;
         LockWindowUpdate(0);
         Screen.Cursor := crDefault;
      end;
   end
   else
   showmessage('Selecione um registro');
   wb_selecionoulinha:=false;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.45 unit uMain;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, JvGradientHeaderPanel, fcLabel, JvExControls, TAcesso,
  JvComponent, JvGradient, ComCtrls, JvXPCore, JvXPBar, TUsuario, TMenu,
  RXSplit;

type
  Tfrm_Main = class(TForm)
    pnl_Menu: TPanel;
    M_AdmSist: TJvXPBar;
    M_OS: TJvXPBar;
    M_Relatorios: TJvXPBar;
    M_sair: TJvXPBar;
    RodaPe: TStatusBar;
    Pnl_Principal: TPanel;
    JvGradient1: TJvGradient;
    L_nome_empresa: TfcLabel;
    L_Desenvolvedores: TJvGradientHeaderPanel;
    L_Hora: TJvGradientHeaderPanel;
    L_Data: TJvGradientHeaderPanel;
    L_empresa_filial: TJvGradientHeaderPanel;
    Relogio: TTimer;
    M_Basico: TJvXPBar;
    M_Caixa: TJvXPBar;
    RxSplitter1: TRxSplitter;
    M_Estoque: TJvXPBar;
    procedure FormCreate(Sender: TObject);
    procedure RelogioTimer(Sender: TObject);
    procedure M_AdmSistItems0Click(Sender: TObject);
    procedure M_AdmSistItems1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure M_sairItems1Click(Sender: TObject);
    procedure M_sairItems0Click(Sender: TObject);
    procedure M_BasicoItems11Click(Sender: TObject);
    procedure M_BasicoItems4Click(Sender: TObject);
    procedure M_BasicoItems5Click(Sender: TObject);
    procedure M_BasicoItems6Click(Sender: TObject);
    procedure M_BasicoItems7Click(Sender: TObject);
    procedure M_BasicoItems0Click(Sender: TObject);
    procedure M_BasicoItems1Click(Sender: TObject);
    procedure M_BasicoItems8Click(Sender: TObject);
    procedure M_BasicoItems9Click(Sender: TObject);
    procedure M_OSItems0Click(Sender: TObject);
    procedure M_OSItems2Click(Sender: TObject);
    procedure M_OSItems3Click(Sender: TObject);
    procedure M_OSItems4Click(Sender: TObject);
    procedure M_OSItems5Click(Sender: TObject);
    procedure M_OSItems1Click(Sender: TObject);
    procedure M_OSItems6Click(Sender: TObject);
    procedure M_EstoqueItems0Click(Sender: TObject);
    procedure M_EstoqueItems1Click(Sender: TObject);
    procedure M_EstoqueItems2Click(Sender: TObject);
    procedure M_OSItems7Click(Sender: TObject);
```

```
    procedure M_OSItems8Click(Sender: TObject);
    procedure M_CaixaItems2Click(Sender: TObject);
    procedure M_CaixaItems0Click(Sender: TObject);
    procedure M_CaixaItems1Click(Sender: TObject);
    procedure M_RelatoriosItems0Click(Sender: TObject);
    procedure M_RelatoriosItems1Click(Sender: TObject);
    procedure M_RelatoriosItems2Click(Sender: TObject);
  private
    procedure Conecta;
    procedure Trata_Tela_Principal(Sender: TObject);
    procedure MontaMenu;
    procedure carrega_acesso_menu;
    { Private declarations }
  public
    { Public declarations }
    Usuario: TCUsuario;
  end;

var
  frm_Main     : Tfrm_Main;

implementation

uses uDm_Base, uLogin, uTAcesso, uLUsuarios, uLGrupoDeUsuarios, uLTPecas,
  uLTDocumentos, uLTEquipamentos, uLTMovimentos, uLEstados, uLCidades,
  uLPecas, uLClientes, uLFornecedores, uLContratos, uLEquipamentos,
  uCOrdemServico, uLOS_Atualiza, uLOS_Fechar, uLOS_Cancelar,
  uCOS_AprovOrcamento, uLLiberaOrcar, uLOrcar, uLOrcamentoRecusado,
  u_L_V_OS, uListaCaixa, uLCPagar, uLCReceber, uRelatOS, uRelatCaixa,
  uRelatEstoque;

{$R *.dfm}

procedure Tfrm_Main.FormCreate(Sender: TObject);
begin
   Screen.OnActiveFormChange    := Trata_Tela_Principal;
   DateSeparator := '/';
   ShortDateFormat := 'dd/mm/yyyy';
   Conecta;
end;

procedure TFrm_Main.Trata_Tela_Principal(Sender: TObject);
begin
   // Trata apresentação do painel geral
   if assigned(Pnl_Principal) then
      Pnl_Principal.Visible := Frm_Main.ActiveMDIChild = nil;
end;

procedure Tfrm_Main.Conecta;
begin
   //---> Abre as Conexoes com o BD
   If not assigned(dm_Base) Then
      dm_Base := Tdm_Base.Create(Self);

   Usuario := TCUsuario.Create(nil);

   //---> Efetua o login
   If not assigned(Frm_Login) Then
      Frm_Login := TFrm_Login.Create(Self);


   if Frm_Login.ShowModal <> mrOK then
   begin
      Frm_Login.Release;
      Frm_Login := nil;
      Application.Terminate;
   end
   else
   begin
      ws_Sql := ('Select * from adm_sysfile where (Statussistema = :p_status)');
      ws_sql:= LowerCase(ws_sql);
      dm_base.sql_Acao.Close;
      dm_base.sql_Acao.SQL.Clear;
      dm_base.sql_Acao.SQL.Add(ws_Sql);
      dm_base.sql_Acao.ParamByName('p_status').Value := '1';
      dm_base.sql_Acao.Open;
```

```
      if not dm_base.sql_Acao.IsEmpty then
      begin
         frm_Main.Caption   := (dm_base.sql_Acao.FieldByName('NomeEmpresa').Value + ' -
' + dm_base.sql_Acao.FieldByName('NomeSistema').Value + ' - ' + '(' + frm_Main.Name +
')');
         l_nome_empresa.Caption:= dm_base.sql_Acao.FieldByName('NomeEmpresa').Value;
         L_Desenvolvedores.LabelCaption:= 'Desenvolvido por: ' +
dm_base.sql_Acao.FieldByName('Desenvolvedores').Value;
         RodaPe.Panels[4].Text := 'Bem vindo ao ' +
dm_base.sql_Acao.FieldByName('NomeSistema').Value;

         if dm_base.sql_Acao.FieldByName('Montamenu').Value = '1' then
            MontaMenu;
      end
      else
      application.Terminate;

      carrega_acesso_menu;
   end;
end;

procedure Tfrm_Main.RelogioTimer(Sender: TObject);
begin
   L_Data.LabelCaption:= 'Data: ' + DateToStr(Date);
   L_Hora.LabelCaption:= 'Hora: ' + TimeToStr(Now);
end;

procedure Tfrm_Main.MontaMenu;
var
   wi_Nivel1, wi_Nivel2, wi_Seq, wi_MenuPai, wi_MenuPai2 : Integer;
   menu : TCMenu;
begin
   wi_Seq        := 0;
   menu := TCMenu.Create(frm_Main);

   //---> Primeiro Nivel
   for wi_Nivel1 := 0 to (pnl_menu.ControlCount - 1) do
   begin
      wi_MenuPai := 0;
      Inc(wi_Seq);
      menu.P_tipo_menu := 1;
      menu.P_sequencia := wi_Seq;
      menu.P_menupai   := wi_MenuPai;
      menu.P_descricao := lowercase((pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Caption);
      menu.P_id_modulo := (wi_Nivel1 + 1);
      menu.P_id_menu   := lowercase(pnl_menu.Controls[wi_Nivel1].Name);
      if (menu.Pesq_id_Menu(lowercase(pnl_menu.Controls[wi_Nivel1].Name))) then
      begin
         menu.Salvar(1);
      end
      else
      begin
         menu.Salvar(0);
      end;
      wi_MenuPai2:= wi_Seq;

      //---> Segundo Nivel
      for wi_Nivel2 := 0 to ((pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Items.Count - 1)
do
      begin
         Inc(wi_Seq);
         menu.P_tipo_menu := 2;
         menu.P_sequencia := wi_Seq;
         menu.P_menupai   := wi_MenuPai2;
         menu.P_descricao := lowercase((pnl_menu.Controls[wi_Nivel1] as
TjvXPBar).Items.Items[wi_Nivel2].Caption);
         menu.P_id_modulo := (wi_Nivel1 + 1);
         menu.P_id_menu   := lowercase((pnl_menu.Controls[wi_Nivel1] as
TjvXPBar).Items.Items[wi_Nivel2].Name);
         if (menu.Pesq_Id_Menu(lowercase((pnl_menu.Controls[wi_Nivel1] as
TjvXPBar).Items.Items[wi_Nivel2].Name))) then
         begin
            menu.Salvar(1);
         end
         else
         begin
            menu.Salvar(0);
```

```
         end;
      end;
   end;
   menu.FreeOnRelease;
end;

procedure Tfrm_Main.carrega_acesso_menu;
var
   wi_Nivel1, wi_Nivel2   : Integer;
   Acesso                 : TCAcesso;
begin
   // Carrega acesso do menu
   Acesso:= tcacesso.Create(frm_main);

   // Primeiro nível
   for wi_Nivel1 := 0 to (pnl_menu.ControlCount - 1) do
   begin
      if
(Acesso.SelecionarAcesso(usuario.P_id_usuario,lowercase((pnl_menu.Controls[wi_Nivel1] as
TjvXPBar).Name))) then
         (pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Visible := True
      else
         (pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Visible := False;

      // Segundo Nível
      for wi_Nivel2 := 0 to ((pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Items.Count - 1)
do
      begin
         if
(Acesso.SelecionarAcesso(usuario.P_id_usuario,lowercase((pnl_menu.Controls[wi_Nivel1] as
TjvXPBar).Items.Items[wi_Nivel2].Name))) then
            (pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Items.Items[wi_Nivel2].Visible :=
True
         else
            (pnl_menu.Controls[wi_Nivel1] as TjvXPBar).Items.Items[wi_Nivel2].Visible :=
False;
      end;
   end;
end;

procedure Tfrm_Main.M_AdmSistItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_TAcesso) then
         frm_TAcesso := Tfrm_TAcesso.Create(Self);

      Screen.Cursor := crDefault;

      frm_TAcesso.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_AdmSistItems1Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LUsuario) then
         frm_LUsuario := Tfrm_LUsuario.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LUsuario.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;
```

```
procedure Tfrm_Main.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   if Application.MessageBox('Deseja realmente SAIR do Sistema!','Saida',mb_YesNo) =
IdYes then
   begin
      usuario.FreeOnRelease;
      dm_Base.my_Conn.Connected := False;
   end
   else
   begin
      Action := caNone;
   end;
end;

procedure Tfrm_Main.M_sairItems1Click(Sender: TObject);
begin
   close;
end;

procedure Tfrm_Main.M_sairItems0Click(Sender: TObject);
var
   i : Integer;
begin
   for i:= MDIChildCount - 1 downto 0 do
       MDIChildren[I].Close;
   if dm_Base.my_Conn.Connected = True then
      dm_Base.my_Conn.Connected := False;
   dm_base.Free;
   dm_base:= nil;
   Conecta;
end;

procedure Tfrm_Main.M_BasicoItems11Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LGrupos) then
         frm_LGrupos := Tfrm_LGrupos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LGrupos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems4Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LTEquipamentos) then
         frm_LTEquipamentos := Tfrm_LTEquipamentos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LTEquipamentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems5Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LTMovimentos) then
```

```
        frm_LTMovimentos := Tfrm_LTMovimentos.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LTMovimentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems6Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LCidades) then
         frm_LCidades := Tfrm_LCidades.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LCidades.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems7Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LEstados) then
         frm_LEstados := Tfrm_LEstados.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LEstados.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LCliente) then
         frm_LCliente := Tfrm_LCliente.Create(Self);

      Screen.Cursor := crDefault;

      frm_LCliente.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems1Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LFornecedores) then
         frm_LFornecedores := Tfrm_LFornecedores.Create(Self);
```

```
      Screen.Cursor := crDefault;

      frm_LFornecedores.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems8Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LContratos) then
         frm_LContratos := Tfrm_LContratos.Create(Self);

      Screen.Cursor := crDefault;

      frm_LContratos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_BasicoItems9Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LEquipamentos) then
         frm_LEquipamentos := Tfrm_LEquipamentos.Create(Self);

      Screen.Cursor := crDefault;

      frm_LEquipamentos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_OSItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_COS) then
         frm_COS := Tfrm_COS.Create(Self);

      Screen.Cursor := crDefault;

      frm_COS.show;
      frm_COS.btn_PesqClienteClick(self);
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_OSItems2Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LAprovaOrcamento) then
         frm_LAprovaOrcamento := Tfrm_LAprovaOrcamento.Create(Self);

      Screen.Cursor := crDefault;

      frm_LAprovaOrcamento.show;
      frm_LAprovaOrcamento.rgp_tipoClick(self);
```

```
    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;
    end;
end;

procedure Tfrm_Main.M_OSItems3Click(Sender: TObject);
begin
    Screen.Cursor := crHourGlass;

    LockWindowUpDate(Handle);
    try
       if not assigned(frm_LOS_Atualiza) then
          frm_LOS_Atualiza := Tfrm_LOS_Atualiza.Create(Self);

       Screen.Cursor := crDefault;

       frm_LOS_Atualiza.show;
       frm_LOS_Atualiza.rgp_tipoClick(self);
    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;
    end;
end;

procedure Tfrm_Main.M_OSItems4Click(Sender: TObject);
begin
    Screen.Cursor := crHourGlass;

    LockWindowUpDate(Handle);
    try
       if not assigned(frm_LOS_Fechar) then
          frm_LOS_Fechar := Tfrm_LOS_Fechar.Create(Self);

       Screen.Cursor := crDefault;

       frm_LOS_Fechar.show;
       frm_LOS_Fechar.rgp_tipo.ItemIndex:= 1;
       frm_LOS_Fechar.rgp_tipoClick(self);
    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;
    end;

end;

procedure Tfrm_Main.M_OSItems5Click(Sender: TObject);
begin
    Screen.Cursor := crHourGlass;

    LockWindowUpDate(Handle);
    try
       if not assigned(frm_LCancelaOS) then
          frm_LCancelaOS := Tfrm_LCancelaOS.Create(Self);

       Screen.Cursor := crDefault;

       frm_LCancelaOS.show;
       frm_LCancelaOS.rgp_tipo.ItemIndex:= 0;
       frm_LCancelaOS.rgp_tipoClick(self);
    finally
       LockWindowUpdate(0);
       Screen.Cursor := crDefault;
    end;
end;

procedure Tfrm_Main.M_OSItems1Click(Sender: TObject);
begin
    Screen.Cursor := crHourGlass;

    LockWindowUpDate(Handle);
    try
       if not assigned(frm_LEfetua) then
          frm_LEfetua := Tfrm_LEfetua.Create(Self);

       Screen.Cursor := crDefault;
```

```
      frm_LEfetua.show;
      frm_LEfetua.rgp_tipo.ItemIndex:= 0;
      frm_LEfetua.rgp_tipoClick(self);
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_OSItems6Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LOrcar) then
         frm_LOrcar := Tfrm_LOrcar.Create(Self);

      Screen.Cursor := crDefault;

      frm_LOrcar.show;
      frm_LOrcar.rgp_tipo.ItemIndex:= 0;
      frm_LOrcar.rgp_tipoClick(self);
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_EstoqueItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LPecas) then
         frm_LPecas := Tfrm_LPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_EstoqueItems1Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LTPecas) then
         frm_LTPecas := Tfrm_LTPecas.Create(Self);

      // Seta Cursor normal
      Screen.Cursor := crDefault;

      frm_LTPecas.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_EstoqueItems2Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LFornecedores) then
         frm_LFornecedores := Tfrm_LFornecedores.Create(Self);

      Screen.Cursor := crDefault;
```

```
      frm_LFornecedores.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_OSItems7Click(Sender: TObject);
begin
// listar orçamentos com estatus RECUSADA

   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LOrcamentoRecusado) then
         frm_LOrcamentoRecusado := Tfrm_LOrcamentoRecusado.Create(Self);

      Screen.Cursor := crDefault;
      frm_LOrcamentoRecusado.rgp_tipo.ItemIndex:= 0;
      frm_LOrcamentoRecusado.rgp_tipoClick(self);
      frm_LOrcamentoRecusado.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_OSItems8Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_L_V_OS) then
         frm_L_V_OS := Tfrm_L_V_OS.Create(Self);

      Screen.Cursor := crDefault;

      frm_L_V_OS.show;
      frm_L_V_OS.rgp_tipo.ItemIndex:= 0;
      frm_L_V_OS.rgp_tipoClick(self);
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_CaixaItems2Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_ListaCaixa) then
         frm_ListaCaixa := Tfrm_ListaCaixa.Create(Self);

      Screen.Cursor := crDefault;

      frm_ListaCaixa.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_CaixaItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LCPagar) then
         frm_LCPagar := Tfrm_LCPagar.Create(Self);

      Screen.Cursor := crDefault;
```

```
      frm_LCPagar.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_CaixaItems1Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_LCReceber) then
         frm_LCReceber := Tfrm_LCReceber.Create(Self);

      Screen.Cursor := crDefault;

      frm_LCReceber.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_RelatoriosItems0Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_Relatos) then
         frm_Relatos := Tfrm_Relatos.Create(Self);

      Screen.Cursor := crDefault;

      frm_Relatos.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_RelatoriosItems1Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
   try
      if not assigned(frm_RelatCaixa) then
         frm_RelatCaixa := Tfrm_RelatCaixa.Create(Self);

      Screen.Cursor := crDefault;

      frm_RelatCaixa.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_Main.M_RelatoriosItems2Click(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_Relatestoque) then
         frm_Relatestoque := Tfrm_Relatestoque.Create(Self);
      Screen.Cursor := crDefault;
      frm_Relatestoque.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.46 unit uMovCaixa;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, fcButton, fcImgBtn, fcShapeBtn, ExtCtrls,
  StdCtrls, Mask, ToolEdit, CurrEdit, fcLabel, fcpanel, ComCtrls, TCaixa, TTipoMov;
type
  Tfrm_MovCaixa = class(Tfrm_TelaCadastroPadrao)
    Edit_Lcto: TCurrencyEdit;
    Edit_Dia: TDateEdit;
    Edit_Historico: TEdit;
    Edit_Valor: TCurrencyEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    lbl_Cliente: TLabel;
    Edit_idCliente: TCurrencyEdit;
    Edit_DescrCliente: TStaticText;
    edit_numdoc: TCurrencyEdit;
    Label3: TLabel;
    Label4: TLabel;
    edit_numparcela: TCurrencyEdit;
    edit_tipopagamento: TComboBox;
    Label6: TLabel;
    Edit_idTipoMov: TComboBox;
    btn_pesq: TfcShapeBtn;
    edit_data_vcto: TDateEdit;
    Label9: TLabel;
    Label10: TLabel;
    edit_prorrog: TDateEdit;
    CheckBox1: TCheckBox;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Edit_DiaEnter(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
    procedure edit_data_vctoEnter(Sender: TObject);
    procedure edit_prorrogEnter(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox1KeyPress(Sender: TObject; var Key: Char);
    procedure Edit_idTipoMovKeyPress(Sender: TObject; var Key: Char);
    procedure edit_tipopagamentoKeyPress(Sender: TObject; var Key: Char);
    procedure btn_SalvaClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure Mostra_Registro;
    procedure monta_tipomov;
  end;
var
  frm_MovCaixa : Tfrm_MovCaixa;
  wb_Altera    : Boolean;
  wb_mudou     : boolean;
  wc_NColuna   : TListColumn;
  wb_salvar    : boolean;
```

```
implementation
uses Funcoes, uLCPagar, uPesqFornecedor;
{$R *.dfm}
procedure Tfrm_MovCaixa.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LCPagar.Monta_Lista;
   frm_MovCaixa.Release;
   frm_MovCaixa := nil;
end;
procedure Tfrm_MovCaixa.FormActivate(Sender: TObject);
begin
  inherited;
   Color := clBtnFace;
end;

procedure Tfrm_MovCaixa.FormShow(Sender: TObject);
begin
  inherited;
   monta_tipomov;

   if uLCPagar.wi_idLcto = 0 then
   begin
      Edit_Lcto.Enabled  := True;
      Edit_Lcto.Enabled  := False;
      wb_mudou           := False;
      wb_Altera          := False;
   end
   else
   begin
      Edit_Lcto.Enabled := True;
      Edit_Lcto.Value   := uLCPagar.wi_idLcto;
      Edit_Lcto.Enabled := False;
      wb_mudou          := True;
      wb_Altera         := True;

      Mostra_Registro;
   end;
end;

procedure Tfrm_MovCaixa.Mostra_Registro;
VAR
   caixa         : TCCaixa;
   tipomovimento : TcTipoMov;
begin
   caixa:= TCCaixa.Create(nil);
   caixa.Selecionar(uLCPagar.wi_idLcto);

   tipomovimento:= TcTipoMov.Create(nil);
   tipomovimento.Selecionar(caixa.P_id_tipomov);

   Edit_Dia.Date                := caixa.P_data_emissao;
   Edit_idtipomov.Text          := ZerosEsq(inttostr(tipomovimento.P_id_tipomov),4) + '
- '+ tipomovimento.P_descricao;
   Edit_idCliente.AsInteger     := caixa.P_id_pessoa;
   Edit_Historico.Text          := caixa.p_Historico;
   Edit_Valor.Value             := caixa.P_valor_apagar;
   Edit_numdoc.AsInteger        := caixa.P_num_doc;
   Edit_numparcela.AsInteger    := caixa.P_num_parcela;
   Edit_tipopagamento.ItemIndex := caixa.P_tipo_pgto;
   edit_data_vcto.Date          := caixa.P_data_vcto;
   edit_prorrog.Date            := caixa.P_data_prorrog;

   caixa.FreeOnRelease;
end;

procedure Tfrm_MovCaixa.Edit_DiaEnter(Sender: TObject);
begin
   if not wb_Altera then
      Edit_Dia.Date := Date;
end;

procedure Tfrm_MovCaixa.btn_pesqClick(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;
```

```
   LockWindowUpDate(Handle);
   try
      if not assigned(frm_PesqFornec) then
         frm_PesqFornec := Tfrm_PesqFornec.Create(Self);

      Screen.Cursor := crDefault;
      frm_PesqFornec.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_MovCaixa.monta_tipomov;
var
   tipomovimento : TcTipoMov;
begin
   tipomovimento:= TcTipoMov.Create(nil);
   tipomovimento.Seleciona(1);
   tipomovimento.Primeiro;
   Edit_idtipomov.Items.Clear;
   repeat
      tipomovimento.Coloca;
      Edit_idtipomov.Items.Add((ZerosEsq(inttostr(tipomovimento.P_id_tipomov),4) + ' -
'+ tipomovimento.P_descricao));
      tipomovimento.proximo;
   until tipomovimento.final;
   freeandnil(tipomovimento);
end;

procedure Tfrm_MovCaixa.edit_data_vctoEnter(Sender: TObject);
begin
   if not wb_Altera then
      edit_data_vcto.Date := Date;
end;

procedure Tfrm_MovCaixa.edit_prorrogEnter(Sender: TObject);
begin
   if not wb_Altera then
      edit_prorrog.Date := Date;
end;

procedure Tfrm_MovCaixa.CheckBox1Click(Sender: TObject);
begin
   if CheckBox1.State = cbUnchecked then
      edit_prorrog.Enabled:= False
   else
      edit_prorrog.Enabled:= True;
end;

procedure Tfrm_MovCaixa.CheckBox1KeyPress(Sender: TObject; var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      key := #0;
      activecontrol:= edit_tipopagamento;
   end;
end;

procedure Tfrm_MovCaixa.Edit_idTipoMovKeyPress(Sender: TObject;
  var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      key := #0;
      activecontrol:= edit_valor;
   end;
end;

procedure Tfrm_MovCaixa.edit_tipopagamentoKeyPress(Sender: TObject;
  var Key: Char);
begin
  inherited;
   //---> Enter
```

```
   if key = #13 then
   begin
      key := #0;
      activecontrol:= edit_idtipomov;
   end;
end;

procedure Tfrm_MovCaixa.btn_SalvaClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
   if Edit_Dia.Date = 0 then
   begin
      showmessage('Digite uma Data Válida!!!');
      Edit_Dia.SetFocus;
      Exit;
   end;

   if not wb_mudou then
   if Edit_idTipoMov.ItemIndex < 0 then
   begin
      showmessage('Digite um Tipo de Movimento!!!');
      Edit_idTipoMov.SetFocus;
      Exit;
   end;

   if Edit_idcliente.AsInteger = 0 then
   begin
      showmessage('Selecione o fornecedor clicando em pesquisar!!!');
      btn_pesq.SetFocus;
      Exit;
   end;

   if edit_tipopagamento.ItemIndex < 0 then
   begin
      showmessage('Digite um Tipo de Pagamento!!!');
      edit_tipopagamento.SetFocus;
      Exit;
   end;

   if Edit_Historico.Text = '' then
   begin
      showmessage('Digite um Histórico diferente de Branco!!!');
      Edit_Historico.SetFocus;
      Exit;
   end;
   if Application.MessageBox('Salva Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
   begin
      caixa:= TCCaixa.Create(nil);
      caixa.P_id_lcto          := uLCPagar.wi_idLcto;
      caixa.Selecionar(uLCPagar.wi_idLcto);
      caixa.P_data_emissao     := Edit_Dia.Date;
      if not wb_mudou then
         caixa.P_id_tipomov    := StrToInt(copy(Edit_idtipomov.Text,1,4));
      caixa.P_id_pessoa        := Edit_idCliente.AsInteger;
      caixa.p_Historico        := Edit_Historico.Text;
      caixa.P_valor_apagar     := Edit_Valor.Value;
      caixa.P_num_doc          := Edit_numdoc.AsInteger;
      caixa.P_num_parcela      := Edit_numparcela.AsInteger;
      caixa.P_tipo_pgto        := Edit_tipopagamento.ItemIndex;
      caixa.P_data_vcto        := edit_data_vcto.Date;
      caixa.P_data_prorrog     := edit_prorrog.Date;
      caixa.P_situacao         := 'A';
      if not wb_Altera then
      begin
         caixa.Salvar(0);
      end
      else
      begin
         caixa.Salvar(1);
      end;
      caixa.FreeOnRelease;
      uLCPagar.wi_idLcto := 0;
      Close;
   end;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.47 unit uMovCaixaCReceber;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, fcButton, fcImgBtn, fcShapeBtn, ExtCtrls,
  StdCtrls, Mask, ToolEdit, CurrEdit, fcLabel, fcpanel, ComCtrls, TCaixa, TTipoMov;
type
  Tfrm_MovCaixaCReceber = class(Tfrm_TelaCadastroPadrao)
    Edit_Lcto: TCurrencyEdit;
    Edit_Dia: TDateEdit;
    Edit_Historico: TEdit;
    Edit_Valor: TCurrencyEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    lbl_Cliente: TLabel;
    Edit_idCliente: TCurrencyEdit;
    Edit_DescrCliente: TStaticText;
    edit_numdoc: TCurrencyEdit;
    Label3: TLabel;
    Label4: TLabel;
    edit_numparcela: TCurrencyEdit;
    edit_tipopagamento: TComboBox;
    Label6: TLabel;
    Edit_idTipoMov: TComboBox;
    btn_pesq: TfcShapeBtn;
    edit_data_vcto: TDateEdit;
    Label9: TLabel;
    Label10: TLabel;
    edit_prorrog: TDateEdit;
    CheckBox1: TCheckBox;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Edit_DiaEnter(Sender: TObject);
    procedure btn_pesqClick(Sender: TObject);
    procedure edit_data_vctoEnter(Sender: TObject);
    procedure edit_prorrogEnter(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox1KeyPress(Sender: TObject; var Key: Char);
    procedure Edit_idTipoMovKeyPress(Sender: TObject; var Key: Char);
    procedure edit_tipopagamentoKeyPress(Sender: TObject; var Key: Char);
    procedure btn_SalvaClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure Mostra_Registro;
    procedure monta_tipomov;
  end;
var
  frm_MovCaixaCReceber : Tfrm_MovCaixaCReceber;
  wb_Altera    : Boolean;
  wb_mudou     : boolean;
  wc_NColuna   : TListColumn;
  wb_salvar    : boolean;
```

```
implementation
uses Funcoes, uLCReceber, uPesqCliente;
{$R *.dfm}
procedure Tfrm_MovCaixaCReceber.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_LCReceber.Monta_Lista;
   frm_MovCaixaCReceber.Release;
   frm_MovCaixaCReceber := nil;
end;
procedure Tfrm_MovCaixaCReceber.FormActivate(Sender: TObject);
begin
  inherited;
   Color := clBtnFace;
end;
procedure Tfrm_MovCaixaCReceber.FormShow(Sender: TObject);
begin
  inherited;
   monta_tipomov;

   if uLCReceber.wi_idLcto = 0 then
   begin
      Edit_Lcto.Enabled  := True;
      Edit_Lcto.Enabled  := False;
      wb_mudou           := False;
      wb_Altera          := False;
   end
   else
   begin
      Edit_Lcto.Enabled := True;
      Edit_Lcto.Value   := uLCReceber.wi_idLcto;
      Edit_Lcto.Enabled := False;
      wb_mudou          := True;
      wb_Altera         := True;

      Mostra_Registro;
   end;
end;

procedure Tfrm_MovCaixaCReceber.Mostra_Registro;
VAR
   caixa         : TCCaixa;
   tipomovimento : TcTipoMov;
begin
   caixa:= TCCaixa.Create(nil);
   caixa.Selecionar(uLCReceber.wi_idLcto);

   tipomovimento:= TcTipoMov.Create(nil);
   tipomovimento.Selecionar(caixa.P_id_tipomov);

   Edit_Dia.Date                := caixa.P_data_emissao;
   Edit_idtipomov.Text          := ZerosEsq(inttostr(tipomovimento.P_id_tipomov),4) + '
- '+ tipomovimento.P_descricao;
   Edit_idCliente.AsInteger     := caixa.P_id_pessoa;
   Edit_Historico.Text          := caixa.p_Historico;
   Edit_Valor.Value             := caixa.P_valor_apagar;
   Edit_numdoc.AsInteger        := caixa.P_num_doc;
   Edit_numparcela.AsInteger    := caixa.P_num_parcela;
   Edit_tipopagamento.ItemIndex := caixa.P_tipo_pgto;
   edit_data_vcto.Date          := caixa.P_data_vcto;
   edit_prorrog.Date            := caixa.P_data_prorrog;

   caixa.FreeOnRelease;
end;

procedure Tfrm_MovCaixaCReceber.Edit_DiaEnter(Sender: TObject);
begin
   if not wb_Altera then
      Edit_Dia.Date := Date;
end;

procedure Tfrm_MovCaixaCReceber.btn_pesqClick(Sender: TObject);
begin
   Screen.Cursor := crHourGlass;

   LockWindowUpDate(Handle);
```

```
   try
      if not assigned(frm_Pesqcliente) then
         frm_Pesqcliente := Tfrm_Pesqcliente.Create(Self);

      Screen.Cursor := crDefault;
      frm_PesqCliente.Tag := 3;
      frm_Pesqcliente.show;
   finally
      LockWindowUpdate(0);
      Screen.Cursor := crDefault;
   end;
end;

procedure Tfrm_MovCaixaCReceber.monta_tipomov;
var
   tipomovimento : TcTipoMov;
begin
   tipomovimento:= TcTipoMov.Create(nil);
   tipomovimento.Seleciona(0);
   tipomovimento.Primeiro;
   Edit_idtipomov.Items.Clear;
   repeat
      tipomovimento.Coloca;
      Edit_idtipomov.Items.Add((ZerosEsq(inttostr(tipomovimento.P_id_tipomov),4) + ' -
'+ tipomovimento.P_descricao));
      tipomovimento.proximo;
   until tipomovimento.final;
   freeandnil(tipomovimento);
end;

procedure Tfrm_MovCaixaCReceber.edit_data_vctoEnter(Sender: TObject);
begin
   if not wb_Altera then
      edit_data_vcto.Date := Date;
end;

procedure Tfrm_MovCaixaCReceber.edit_prorrogEnter(Sender: TObject);
begin
   if not wb_Altera then
      edit_prorrog.Date := Date;
end;

procedure Tfrm_MovCaixaCReceber.CheckBox1Click(Sender: TObject);
begin
   if CheckBox1.State = cbUnchecked then
      edit_prorrog.Enabled:= False
   else
      edit_prorrog.Enabled:= True;
end;

procedure Tfrm_MovCaixaCReceber.CheckBox1KeyPress(Sender: TObject; var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      key := #0;
      activecontrol:= edit_tipopagamento;
   end;
end;

procedure Tfrm_MovCaixaCReceber.Edit_idTipoMovKeyPress(Sender: TObject;
  var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      key := #0;
      activecontrol:= edit_valor;
   end;
end;

procedure Tfrm_MovCaixaCReceber.edit_tipopagamentoKeyPress(Sender: TObject;
  var Key: Char);
begin
  inherited;
   //---> Enter
   if key = #13 then
```

```
    begin
       key := #0;
       activecontrol:= edit_idtipomov;
    end;
end;

procedure Tfrm_MovCaixaCReceber.btn_SalvaClick(Sender: TObject);
var
   caixa       : TCCaixa;
begin
   if Edit_Dia.Date = 0 then
   begin
      showmessage('Digite uma Data Válida!!!');
      Edit_Dia.SetFocus;
      Exit;
   end;

   if not wb_mudou then
   if Edit_idTipoMov.ItemIndex < 0 then
   begin
      showmessage('Digite um Tipo de Movimento!!!');
      Edit_idTipoMov.SetFocus;
      Exit;
   end;

   if Edit_idcliente.AsInteger = 0 then
   begin
      showmessage('Selecione o Cliente clicando em pesquisar!!!');
      btn_pesq.SetFocus;
      Exit;
   end;

   if edit_tipopagamento.ItemIndex < 0 then
   begin
      showmessage('Digite um Tipo de Pagamento!!!');
      edit_tipopagamento.SetFocus;
      Exit;
   end;

   if Edit_Historico.Text = '' then
   begin
      showmessage('Digite um Histórico diferente de Branco!!!');
      Edit_Historico.SetFocus;
      Exit;
   end;

   if Application.MessageBox('Salva Lançamento?!','Atenção!!!', mb_YesNo) = IdYes then
   begin
      caixa:= TCCaixa.Create(nil);
      caixa.P_id_lcto          := uLCReceber.wi_idLcto;
      caixa.Selecionar(uLCReceber.wi_idLcto);
      caixa.P_data_emissao     := Edit_Dia.Date;
      if not wb_mudou then
         caixa.P_id_tipomov    := StrToInt(copy(Edit_idtipomov.Text,1,4));
      caixa.P_id_pessoa        := Edit_idCliente.AsInteger;
      caixa.p_Historico        := Edit_Historico.Text;
      caixa.P_valor_apagar     := Edit_Valor.Value;
      caixa.P_num_doc          := Edit_numdoc.AsInteger;
      caixa.P_num_parcela      := Edit_numparcela.AsInteger;
      caixa.P_tipo_pgto        := Edit_tipopagamento.ItemIndex;
      caixa.P_data_vcto        := edit_data_vcto.Date;
      caixa.P_data_prorrog     := edit_prorrog.Date;
      caixa.P_situacao         := 'A';
      if not wb_Altera then
      begin
         caixa.Salvar(0);
      end
      else
      begin
         caixa.Salvar(1);
      end;
      caixa.FreeOnRelease;
      uLCReceber.wi_idLcto := 0;
      Close;
   end;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.48 unit uPesqCliente;

```
interface

uses
  Windows, TPessoa, uTelaCadastroPadrao, Controls, ComCtrls, StdCtrls, forms,
  ExtCtrls, Buttons, fcButton, fcImgBtn, fcShapeBtn, Classes, SysUtils;

type
  Tfrm_PesqCliente = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    btn_Pesquisa: TSpeedButton;
    Edit_Valor: TEdit;
    rgp_Sel: TRadioGroup;
    lv: TListView;
    procedure Edit_ValorKeyPress(Sender: TObject; var Key: Char);
    procedure btn_PesquisaClick(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure lvDblClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_FechaClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
     procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_PesqCliente: Tfrm_PesqCliente;
  wb_cancelou    : Boolean;

implementation

uses uCContratos, uCOrdemServico, uMovCaixaCReceber;

{$R *.dfm}

procedure Tfrm_PesqCliente.Edit_ValorKeyPress(Sender: TObject;
  var Key: Char);
begin
   if key = #13 then
   begin
      btn_PesquisaClick(Self);
   end;
end;

procedure Tfrm_PesqCliente.btn_PesquisaClick(Sender: TObject);
begin
  inherited;
     listar_lv;
end;

procedure Tfrm_PesqCliente.listar_lv;
var
   wl_ItemList : TListItem;
```

```
   Pessoalist  : TcPessoa;
begin
   Pessoalist:= TcPessoa.Create(frm_PesqCliente);

      //---> por CNPJ
      if rgp_Sel.ItemIndex = 0 then
      begin
         Pessoalist.Pesquisar(edit_valor.Text,0,1);
      end;
      //---> por RAZAO SOCIAL
      if rgp_Sel.ItemIndex = 1 then
      begin
         Pessoalist.Pesquisar(edit_valor.Text,1,1);
      end;
   btn_Salva.Enabled:= False;
   Pessoalist.Primeiro;
   lv.Items.Clear;
   repeat
      Pessoalist.Coloca;
      if Pessoalist.P_id_pessoa <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Pessoalist.P_id_pessoa);
         wl_ItemList.SubItems.Add(Pessoalist.P_nome);
         wl_ItemList.SubItems.Add(Pessoalist.P_cgc_cnpj);
         wl_ItemList.SubItems.Add(Pessoalist.P_contato);
         btn_Salva.Enabled:= true;
      end;
      Pessoalist.Proximo;
   until Pessoalist.Final;
end;

procedure Tfrm_PesqCliente.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
begin
  inherited;
   for wi_Cont_Cons:= 0 to (lv.Items.Count - 1) do
   begin
      if lv.Items.Item[wi_Cont_Cons].Selected = True then
      begin
         if frm_PesqCliente.Tag = 1 then
         begin
            frm_CContratos.e_idpessoa.Text   := lv.Items.Item[wi_Cont_Cons].Caption;
            frm_CContratos.e_descpessoa.Text := lv.Items.Item[wi_Cont_Cons].SubItems[0];
            frm_CContratos.btn_pesq.Enabled  := False;
         end;
         if frm_PesqCliente.Tag = 2 then
         begin
            frm_COS.e_idpessoa.Text    := lv.Items.Item[wi_Cont_Cons].Caption;
            frm_COS.e_descpessoa.Text  := lv.Items.Item[wi_Cont_Cons].SubItems[0];
            frm_COS.RB_VALORES.Enabled := True;
            frm_COS.RG_TAtend.Enabled  := True;
         end;
         if frm_PesqCliente.Tag = 3 then
         begin
            frm_MovCaixaCReceber.Edit_idCliente.Text       :=
lv.Items.Item[wi_Cont_Cons].Caption;
            frm_MovCaixaCReceber.Edit_DescrCliente.Caption :=
lv.Items.Item[wi_Cont_Cons].SubItems[0];
         end;
         wb_cancelou := False;
      end;
   end;
   Close;
end;

procedure Tfrm_PesqCliente.lvDblClick(Sender: TObject);
begin
  inherited;
   btn_SalvaClick(Self);
end;

procedure Tfrm_PesqCliente.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
```

```
   if wb_cancelou then
      if frm_PesqCliente.Tag = 2 then
      begin
         frm_COS.btn_FechaClick(self);
      end;
   frm_PesqCliente.Release;
   frm_PesqCliente:= nil;
end;

procedure Tfrm_PesqCliente.btn_FechaClick(Sender: TObject);
begin
   wb_cancelou := True;
   if frm_PesqCliente.Tag = 1 then
   begin
      frm_CContratos.btn_FechaClick(SELF);
   end;
      inherited;

end;

procedure Tfrm_PesqCliente.FormShow(Sender: TObject);
begin
  inherited;
   wb_cancelou := True;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.49 unit uPesqContratos;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, ComCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TPessoa;

type
  Tfrm_PContrato = class(Tfrm_TelaCadastroPadrao)
    lv: TListView;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
    procedure lvDblClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btn_FechaClick(Sender: TObject);
  private
     procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_PContrato: Tfrm_PContrato;

implementation

uses uCOrdemServico;

{$R *.dfm}

procedure Tfrm_PContrato.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_PContrato.Release;
   frm_PContrato:= nil;
end;

procedure Tfrm_PContrato.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
begin
  inherited;
   for wi_Cont_Cons:= 0 to (lv.Items.Count - 1) do
   begin
      if lv.Items.Item[wi_Cont_Cons].Selected = True then
      begin
            frm_COS.e_ncontrato.Text      := lv.Items.Item[wi_Cont_Cons].Caption;
            frm_COS.btn_pesqEquip.Enabled := true;
            frm_COS.btn_pesqEquipClick(Self);
      end;
   end;
   Close;
end;
```

```
procedure Tfrm_PContrato.lvDblClick(Sender: TObject);
begin
  inherited;
   btn_SalvaClick(Self);
end;

procedure Tfrm_PContrato.FormShow(Sender: TObject);
begin
  inherited;
   listar_lv;
end;

procedure Tfrm_PContrato.listar_lv;
var
   pescontr    : TCpessoa;
   wl_ItemList : TListItem;
begin
   pescontr:= TCpessoa.Create(frm_PContrato);
   pescontr.Contrato.Pesquisar(frm_COS.e_idpessoa.AsInteger);
   pescontr.Contrato.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Contrato.Coloca;
      if pescontr.Contrato.P_id_contrato <> 0 then
      begin
         wl_ItemList             := lv.Items.Add;
         wl_ItemList.Caption     := inttostr(pescontr.Contrato.P_id_contrato);
         wl_ItemList.SubItems.Add(datetostr(pescontr.Contrato.P_data_inicio_vigencia));
         wl_ItemList.SubItems.Add(datetostr(pescontr.Contrato.P_data_fim_vigencia));
         pescontr.Contrato.proximo;
      end;
   until pescontr.Contrato.final;
   pescontr.freeonrelease;

end;

procedure Tfrm_PContrato.btn_FechaClick(Sender: TObject);
begin
  inherited;
  frm_COS.RG_TAtend.itemindex     := -1;
  frm_COS.e_ncontrato.Value       := 0;
  frm_COS.e_nequipamento.Value    := 0;
  frm_COS.RG_TAtend.enabled       := True;
  frm_COS.p_desc.Enabled          := False;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.50 unit uPesqEquipamento;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, ComCtrls, fcButton, fcImgBtn, fcShapeBtn,
  ExtCtrls, TPessoa;

type
  Tfrm_PEquipamentos = class(Tfrm_TelaCadastroPadrao)
    lv: TListView;
    procedure FormShow(Sender: TObject);
    procedure lvDblClick(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_FechaClick(Sender: TObject);
  private
    procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_PEquipamentos: Tfrm_PEquipamentos;

implementation

uses uCOrdemServico;

{$R *.dfm}

procedure Tfrm_PEquipamentos.listar_lv;
var
   pescontr    : TCpessoa;
   wl_ItemList : TListItem;
begin
   pescontr:= TCpessoa.Create(frm_PEquipamentos);
   pescontr.Equipamento.Pesquisar(frm_COS.e_ncontrato.AsInteger);
   pescontr.Equipamento.Primeiro;
   lv.Items.Clear;
   repeat
      pescontr.Equipamento.Coloca;
      if pescontr.Equipamento.P_id_equipamento <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(pescontr.Equipamento.P_id_equipamento);
         wl_ItemList.SubItems.Add(pescontr.Equipamento.P_numero_serie);
         wl_ItemList.SubItems.Add(pescontr.Equipamento.P_descricao);
         btn_Salva.enabled := True;
         pescontr.Equipamento.proximo;
      end;
   until pescontr.Equipamento.final;
   pescontr.freeonrelease;

end;
```

```
procedure Tfrm_PEquipamentos.FormShow(Sender: TObject);
begin
  inherited;
   listar_lv;
end;

procedure Tfrm_PEquipamentos.lvDblClick(Sender: TObject);
begin
  inherited;
   btn_SalvaClick(Self);
end;

procedure Tfrm_PEquipamentos.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
begin
  inherited;
   for wi_Cont_Cons:= 0 to (lv.Items.Count - 1) do
   begin
      if lv.Items.Item[wi_Cont_Cons].Selected = True then
      begin
            frm_COS.e_nequipamento.Text   := lv.Items.Item[wi_Cont_Cons].Caption;
            frm_COS.Edit_desc_equip.Text  := lv.Items.Item[wi_Cont_Cons].SubItems[1];
            frm_COS.p_desc.Enabled        := True;
            frm_COS.Edit_desc_prob.SetFocus;
      end;
   end;
   Close;
end;

procedure Tfrm_PEquipamentos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_PEquipamentos.Release;
   frm_PEquipamentos:= nil;
end;

procedure Tfrm_PEquipamentos.btn_FechaClick(Sender: TObject);
begin
  inherited;
  frm_COS.p_desc.Enabled        := False;
  frm_COS.btn_pesqContratoClick(self);
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.51 unit uPesqFornecedor;

```
interface

uses
  Windows, TPessoa, uTelaCadastroPadrao, Controls, ComCtrls, StdCtrls, forms,
  ExtCtrls, Buttons, fcButton, fcImgBtn, fcShapeBtn, Classes, SysUtils;

type
  Tfrm_PesqFornec = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    btn_Pesquisa: TSpeedButton;
    Edit_Valor: TEdit;
    rgp_Sel: TRadioGroup;
    lv: TListView;
    procedure Edit_ValorKeyPress(Sender: TObject; var Key: Char);
    procedure btn_PesquisaClick(Sender: TObject);
    procedure btn_SalvaClick(Sender: TObject);
    procedure lvDblClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
  private
     procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_PesqFornec: Tfrm_PesqFornec;
  wb_cancelou    : Boolean;

implementation

uses uCContratos, uCOrdemServico, uMovCaixa;

{$R *.dfm}

procedure Tfrm_PesqFornec.Edit_ValorKeyPress(Sender: TObject;
  var Key: Char);
begin
   if key = #13 then
   begin
      btn_PesquisaClick(Self);
   end;
end;

procedure Tfrm_PesqFornec.btn_PesquisaClick(Sender: TObject);
begin
  inherited;
     listar_lv;
end;

procedure Tfrm_PesqFornec.listar_lv;
var
   wl_ItemList : TListItem;
   Pessoalist  : TcPessoa;
```

```
begin
   Pessoalist:= TcPessoa.Create(nil);

      //---> por CNPJ
      if rgp_Sel.ItemIndex = 0 then
      begin
         Pessoalist.Pesquisar(edit_valor.Text,0,2);
      end;
      //---> por RAZAO SOCIAL
      if rgp_Sel.ItemIndex = 1 then
      begin
         Pessoalist.Pesquisar(edit_valor.Text,1,2);
      end;
   btn_Salva.Enabled:= False;
   Pessoalist.Primeiro;
   lv.Items.Clear;
   repeat
      Pessoalist.Coloca;
      if Pessoalist.P_id_pessoa <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Pessoalist.P_id_pessoa);
         wl_ItemList.SubItems.Add(Pessoalist.P_nome);
         wl_ItemList.SubItems.Add(Pessoalist.P_cgc_cnpj);
         wl_ItemList.SubItems.Add(Pessoalist.P_contato);
         btn_Salva.Enabled:= true;
      end;
      Pessoalist.Proximo;
   until Pessoalist.Final;
end;

procedure Tfrm_PesqFornec.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
begin
  inherited;
   for wi_Cont_Cons:= 0 to (lv.Items.Count - 1) do
   begin
      if lv.Items.Item[wi_Cont_Cons].Selected = True then
      begin
            frm_MovCaixa.Edit_idCliente.Text       :=
lv.Items.Item[wi_Cont_Cons].Caption;
            frm_MovCaixa.Edit_DescrCliente.Caption :=
lv.Items.Item[wi_Cont_Cons].SubItems[0];
            wb_cancelou := False;
      end;
   end;
   Close;
end;

procedure Tfrm_PesqFornec.lvDblClick(Sender: TObject);
begin
  inherited;
   btn_SalvaClick(Self);
end;

procedure Tfrm_PesqFornec.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_PesqFornec.Release;
   frm_PesqFornec:= nil;
end;

procedure Tfrm_PesqFornec.FormShow(Sender: TObject);
begin
  inherited;
   wb_cancelou := True;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.52 unit uPesqPecas;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, uTelaCadastroPadrao, ComCtrls, StdCtrls, Buttons, fcButton,
  fcImgBtn, fcShapeBtn, ExtCtrls, TPeca;
type
  Tfrm_PesqPecas = class(Tfrm_TelaCadastroPadrao)
    Panel3: TPanel;
    btn_Pesquisa: TSpeedButton;
    Edit_Valor: TEdit;
    lv: TListView;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_SalvaClick(Sender: TObject);
    procedure lvDblClick(Sender: TObject);
    procedure btn_PesquisaClick(Sender: TObject);
    procedure Edit_ValorKeyPress(Sender: TObject; var Key: Char);
  private
    procedure listar_lv;
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_PesqPecas: Tfrm_PesqPecas;
implementation
uses uCOS_Atualizar, uCOrdemServico, uLOrcarAltera, uCAltera_Orcamento;
{$R *.dfm}
procedure Tfrm_PesqPecas.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  inherited;
   frm_PesqPecas.Release;
   frm_PesqPecas:= nil;
end;

procedure Tfrm_PesqPecas.btn_SalvaClick(Sender: TObject);
var
   wi_Cont_Cons : Integer;
begin
  inherited;
   for wi_Cont_Cons:= 0 to (lv.Items.Count - 1) do
   begin
      if lv.Items.Item[wi_Cont_Cons].Selected = True then
      begin
         if frm_PesqPecas.Tag = 1 then
         begin
            frm_COS.id_peca.Text            := lv.Items.Item[wi_Cont_Cons].Caption;
            frm_COS.desc_peca.Text          := lv.Items.Item[wi_Cont_Cons].SubItems[0];
            frm_COS.edit_qtdepeca.Enabled   := True;
            frm_COS.edit_qtdepeca.AsInteger := 1;
         end;
         if frm_PesqPecas.Tag = 2 then
         begin
            frm_COS_Atualizar.id_peca.Text            :=
lv.Items.Item[wi_Cont_Cons].Caption;
            frm_COS_Atualizar.desc_peca.Text          :=
lv.Items.Item[wi_Cont_Cons].SubItems[0];
```

```
            frm_COS_Atualizar.edit_qtdepeca.Enabled   := True;
            frm_COS_Atualizar.edit_qtdepeca.AsInteger := 1;
         end;
         if frm_PesqPecas.Tag = 3 then
         begin
            frm_LAltOrcar.id_peca.Text            :=
lv.Items.Item[wi_Cont_Cons].Caption;
            frm_LAltOrcar.desc_peca.Text          :=
lv.Items.Item[wi_Cont_Cons].SubItems[0];
            frm_LAltOrcar.edit_qtdepeca.Enabled   := True;
            frm_LAltOrcar.edit_qtdepeca.AsInteger := 1;
         end;
         if frm_PesqPecas.Tag = 4 then
         begin
            frm_AltOrcamento.id_peca.Text            :=
lv.Items.Item[wi_Cont_Cons].Caption;
            frm_AltOrcamento.desc_peca.Text          :=
lv.Items.Item[wi_Cont_Cons].SubItems[0];
            frm_AltOrcamento.edit_qtdepeca.Enabled   := True;
            frm_AltOrcamento.edit_qtdepeca.AsInteger := 1;
         end;
      end;
   end;
   Close;
end;

procedure Tfrm_PesqPecas.lvDblClick(Sender: TObject);
begin
  inherited;
   btn_SalvaClick(Self);
end;

procedure Tfrm_PesqPecas.listar_lv;
var
   wl_ItemList : TListItem;
   Pessoalist  : TcPeca;
begin
   Pessoalist:= TcPeca.Create(nil);
   Pessoalist.Pesquisar(edit_valor.Text);
   Pessoalist.Primeiro;
   lv.Items.Clear;
   repeat
      Pessoalist.Coloca;
      if Pessoalist.P_id_peca <> 0 then
      begin
         wl_ItemList            := lv.Items.Add;
         wl_ItemList.Caption    := inttostr(Pessoalist.P_id_peca);
         wl_ItemList.SubItems.Add(Pessoalist.P_descricao);
         Pessoalist.Tipo_da_Peca.Selecionar(Pessoalist.P_id_tipo_peca);
         wl_ItemList.SubItems.Add(Pessoalist.Tipo_da_Peca.P_descricao);
         wl_ItemList.SubItems.Add(inttostr(Pessoalist.P_quantidade));
         wl_ItemList.SubItems.Add('R$ '+floattostr(Pessoalist.P_valor_unitario));
      end;
      Pessoalist.Proximo;
   until Pessoalist.Final;
end;

procedure Tfrm_PesqPecas.btn_PesquisaClick(Sender: TObject);
begin
  inherited;
   listar_lv;
end;

procedure Tfrm_PesqPecas.Edit_ValorKeyPress(Sender: TObject;
  var Key: Char);
begin
  inherited;
   if key = #13 then
   begin
      btn_PesquisaClick(Self);
   end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.53 unit uRelatCaixa;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, fcLabel, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn,
  StdCtrls, Mask, ToolEdit, CurrEdit;

type
  Tfrm_RelatCaixa = class(TForm)
    pg: TPageControl;
    tab_RelFluxo: TTabSheet;
    Panel2: TPanel;
    Panel3: TPanel;
    btn_Fechar: TfcShapeBtn;
    Label1: TLabel;
    Edit_FlxDataIni: TDateEdit;
    Label2: TLabel;
    Edit_FlxDataFim: TDateEdit;
    btn_FlxVer: TfcShapeBtn;
    tab_DespGer: TTabSheet;
    Panel4: TPanel;
    Label3: TLabel;
    Label4: TLabel;
    Edit_DgDataIni: TDateEdit;
    Edit_DgDataFim: TDateEdit;
    btn_DgVer: TfcShapeBtn;
    Edit_DgTpMov: TCurrencyEdit;
    Edit_DgFornec: TCurrencyEdit;
    Label5: TLabel;
    Label6: TLabel;
    TabSheet1: TTabSheet;
    Panel5: TPanel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Edit_RgDataIni: TDateEdit;
    Edit_RgDataFim: TDateEdit;
    btn_RgVer: TfcShapeBtn;
    Edit_RgTpMov: TCurrencyEdit;
    Edit_RgCliente: TCurrencyEdit;
    procedure btn_FecharClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_RelatCaixa  : Tfrm_RelatCaixa;
  wc_NColuna : TListColumn;

implementation
```

```
uses uMain;

{$R *.dfm}

procedure Tfrm_RelatCaixa.btn_FecharClick(Sender: TObject);
begin
   frm_RelatCaixa.Release;
   frm_RelatCaixa := nil;

   frm_main.pnl_Menu.Enabled:= True;
end;

procedure Tfrm_RelatCaixa.FormShow(Sender: TObject);
begin
   Color                    := clBtnFace;

   frm_main.pnl_Menu.Enabled := false;

end;

procedure Tfrm_RelatCaixa.FormKeyPress(Sender: TObject; var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      Perform(Wm_NextDlgCtl,0,0);
      key := #0;
   end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.54 unit uRelatEstoque;

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, fcLabel, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn,
  StdCtrls, Mask, ToolEdit, CurrEdit;
type
  Tfrm_Relatestoque = class(TForm)
    Panel3: TPanel;
    btn_Fechar: TfcShapeBtn;
    pg: TPageControl;
    tab_RelFluxo: TTabSheet;
    Panel2: TPanel;
    rgp_tipo: TRadioGroup;
    btn_FlxVer: TfcShapeBtn;
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    procedure btn_FecharClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_Relatestoque  : Tfrm_Relatestoque;
  wc_NColuna : TListColumn;
implementation
uses  uMain;
{$R *.dfm}
procedure Tfrm_Relatestoque.btn_FecharClick(Sender: TObject);
begin
   frm_Relatestoque.Release;
   frm_Relatestoque := nil;

   frm_main.pnl_Menu.Enabled:= True;
end;

procedure Tfrm_Relatestoque.FormShow(Sender: TObject);
begin
   Color                    := clBtnFace;

   frm_main.pnl_Menu.Enabled := false;

end;

procedure Tfrm_Relatestoque.FormKeyPress(Sender: TObject; var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      Perform(Wm_NextDlgCtl,0,0);
      key := #0;
   end;
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.55 unit uRelatOS;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, fcLabel, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn,
  StdCtrls, Mask, ToolEdit, CurrEdit, ppVar, ppBands, ppCtrls, RxGIF,
  ppPrnabl, ppClass, ppCache, ppProd, ppReport, DB, ppComm, ppRelatv, ppDB,
  ppDBPipe, PSQLDbTables, ppStrtch, ppMemo;

type
  Tfrm_Relatos = class(TForm)
    Panel3: TPanel;
    btn_Fechar: TfcShapeBtn;
    pg: TPageControl;
    tab_RelFluxo: TTabSheet;
    Panel2: TPanel;
    tab_DespGer: TTabSheet;
    Panel4: TPanel;
    Label3: TLabel;
    Label4: TLabel;
    Edit_DgDataIni: TDateEdit;
    Edit_DgDataFim: TDateEdit;
    rgp_tipo: TRadioGroup;
    Label1: TLabel;
    Label2: TLabel;
    Edit_FlxDataIni: TDateEdit;
    Edit_FlxDataFim: TDateEdit;
    btn_pesq: TfcShapeBtn;
    Edit_DescrCliente: TStaticText;
    lbl_Cliente: TLabel;
    CurrencyEdit1: TCurrencyEdit;
    GroupBox1: TGroupBox;
    Label5: TLabel;
    Edit_N_OS: TCurrencyEdit;
    pp_acao: TppDBPipeline;
    ds_acao: TDataSource;
    pp_relOS: TppReport;
    ppHeaderBand1: TppHeaderBand;
    ppShape1: TppShape;
    ppImage1: TppImage;
    ppLabel1: TppLabel;
    ppDetailBand1: TppDetailBand;
    ppFooterBand1: TppFooterBand;
    sql_acao: TPSQLQuery;
    ppGroup1: TppGroup;
    ppGroupHeaderBand1: TppGroupHeaderBand;
    ppGroupFooterBand1: TppGroupFooterBand;
    ppGroup2: TppGroup;
    ppGroupHeaderBand2: TppGroupHeaderBand;
    ppGroupFooterBand2: TppGroupFooterBand;
    ppLabel3: TppLabel;
    ppDBText1: TppDBText;
    ppLabel4: TppLabel;
    ppDBText2: TppDBText;
    ppLabel5: TppLabel;
```

```
    ppDBText3: TppDBText;
    ppLabel6: TppLabel;
    ppDBText4: TppDBText;
    ppLabel15: TppLabel;
    ppDBMemo1: TppDBMemo;
    ppLabel16: TppLabel;
    ppDBText11: TppDBText;
    ppLabel17: TppLabel;
    ppDBText12: TppDBText;
    ppLabel18: TppLabel;
    ppDBText13: TppDBText;
    ppLabel19: TppLabel;
    ppDBText14: TppDBText;
    ppLabel20: TppLabel;
    ppDBText15: TppDBText;
    ppLabel21: TppLabel;
    ppDBText16: TppDBText;
    ppLabel22: TppLabel;
    ppDBText17: TppDBText;
    ppLabel23: TppLabel;
    ppDBText18: TppDBText;
    ppLabel24: TppLabel;
    ppDBText19: TppDBText;
    ppShape3: TppShape;
    ppShape4: TppShape;
    ppLabel25: TppLabel;
    ppDBText20: TppDBText;
    ppLabel26: TppLabel;
    ppLabel27: TppLabel;
    ppLabel28: TppLabel;
    ppDBText22: TppDBText;
    ppDBText23: TppDBText;
    ppLabel29: TppLabel;
    ppDBText24: TppDBText;
    ppLabel30: TppLabel;
    ppDBText25: TppDBText;
    ppLabel31: TppLabel;
    ppDBText26: TppDBText;
    ppLabel32: TppLabel;
    ppDBText27: TppDBText;
    ppShape5: TppShape;
    ppShape6: TppShape;
    ppShape7: TppShape;
    ppLabel33: TppLabel;
    ppDBText28: TppDBText;
    ppDBCalc1: TppDBCalc;
    ppLabel36: TppLabel;
    ppShape8: TppShape;
    ppSummaryBand1: TppSummaryBand;
    ppShape2: TppShape;
    ppSystemVariable1: TppSystemVariable;
    ppLabel2: TppLabel;
    ppSystemVariable2: TppSystemVariable;
    ppLabel81: TppLabel;
    ppLine1: TppLine;
    ppLabel7: TppLabel;
    ppLabel8: TppLabel;
    ppLine2: TppLine;
    ppLabel9: TppLabel;
    ppDBText5: TppDBText;
    ppLabel10: TppLabel;
    ppDBText6: TppDBText;
    ppLabel11: TppLabel;
    ppDBText7: TppDBText;
    ppLabel12: TppLabel;
    ppDBText8: TppDBText;
    ppLabel13: TppLabel;
    ppDBText9: TppDBText;
    ppLabel14: TppLabel;
    ppDBText10: TppDBText;
    ppShape9: TppShape;
    lbl_tpatend: TppLabel;
    btn_FlxVer: TfcShapeBtn;
    procedure btn_FecharClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure lbl_tpatendPrint(Sender: TObject);
```

```
    procedure btn_FlxVerClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_Relatos  : Tfrm_Relatos;
  wc_NColuna : TListColumn;

implementation

uses uMain, uDm_Base;



{$R *.dfm}

procedure Tfrm_Relatos.btn_FecharClick(Sender: TObject);
begin
   frm_Relatos.Release;
   frm_Relatos := nil;

   frm_main.pnl_Menu.Enabled:= True;
end;

procedure Tfrm_Relatos.FormShow(Sender: TObject);
begin
   Color                    := clBtnFace;

   frm_main.pnl_Menu.Enabled := false;
end;

procedure Tfrm_Relatos.FormKeyPress(Sender: TObject; var Key: Char);
begin
   //---> Enter
   if key = #13 then
   begin
      Perform(Wm_NextDlgCtl,0,0);
      key := #0;
   end;
end;

procedure Tfrm_Relatos.lbl_tpatendPrint(Sender: TObject);
begin
   case sql_acao.FieldByName('tipo_atendimento').AsInteger of
      0 : lbl_tpatend.Caption := 'AVULSO';
      1 : lbl_tpatend.Caption := 'CONTRATO';
      2 : lbl_tpatend.Caption := 'GARANTIA';
   end;
end;

procedure Tfrm_Relatos.btn_FlxVerClick(Sender: TObject);
begin
   SQL_Acao.Close;
   SQL_Acao.SQL.Clear;
   WS_SQL := 'select * from rel_os';
   ws_Sql := (ws_Sql + ' where id_os = :P_IDOS');
   SQL_Acao.SQL.Add(WS_SQL);
   SQL_Acao.ParamByName('P_IDOS').AsInteger    := Edit_N_OS.AsInteger;
   SQL_Acao.Open;
   if not sql_acao.IsEmpty then
   begin
      pp_relos.Print;
   end
   else
      showmessage('Nada a ser impresso sai fora mane');
end;

end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.56 unit uTAcesso;

```
interface
uses
  Windows, ComCtrls, StdCtrls, ExtCtrls, fcButton, fcImgBtn, fcShapeBtn,
  Controls, Classes, Forms, TAcesso,TMenu,TUsuario, dialogs;
type
  Tfrm_TAcesso = class(TForm)
    Panel1: TPanel;
    Combo_Usu: TComboBox;
    Label1: TLabel;
    Edit_Grupo: TEdit;
    Label2: TLabel;
    Panel2: TPanel;
    Panel3: TPanel;
    Splitter1: TSplitter;
    Label4: TLabel;
    Label5: TLabel;
    tv_Menu: TTreeView;
    rgp_Nivel: TRadioGroup;
    Panel4: TPanel;
    btn_Fechar: TfcShapeBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure btn_FecharClick(Sender: TObject);
    procedure Combo_UsuChange(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure grid_MenuEnter(Sender: TObject);
    procedure grid_MMenuEnter(Sender: TObject);
    procedure tv_MenuClick(Sender: TObject);
    procedure rgp_NivelClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure tv_MenuKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure Monta_Combo_Usu;
    procedure Mostra_Nivel;
    procedure Monta_tv_Menu;
  end;
var
   frm_TAcesso  : Tfrm_TAcesso;
   wb_alterar   : boolean;
   wi_IDUsu     : Integer;
   ws_idMenu    : String;
   wi_Nivel     : Integer;
   ws_ItemMenu  : String;
   ws_MenuDescr : String;
   wi_Seq       : Integer;
   wt_ANode     : TTreeNode;
implementation
{$R *.dfm}
procedure Tfrm_TAcesso.Monta_tv_Menu;
var
   wi_Modulo : Integer;
   menu      : TCMenu;
   TreeNode  : TTreeNode;
```

```
   TreeNode2 : TTreeNode;
   TreeNode3 : TTreeNode;
   TreeNode4 : TTreeNode;
   TreeNode5 : TTreeNode;
   TreeNode6 : TTreeNode;
   TreeNode7 : TTreeNode;
   TreeNode8 : TTreeNode;
   TreeNode9 : TTreeNode;
begin
   Menu := TCMenu.Create(nil);
   Menu.Seleciona;
   Menu.Primeiro;
   tv_Menu.ScrollBy(1,1);
   tv_Menu.Items.Clear;
   TreeNode           := tv_Menu.TopItem;
   wi_Modulo          := 0;
   repeat
      Menu.Coloca;
      if (wi_Modulo <> menu.P_id_modulo) then
      begin
         TreeNode     := tv_Menu.Items.Add(nil,menu.P_descricao);
      end
      else
      begin
         if (menu.P_tipo_menu = 2) then
            TreeNode2 := tv_Menu.Items.AddChild(TreeNode, menu.P_descricao)
         else if (menu.P_tipo_menu = 3) then
            TreeNode3 := tv_Menu.Items.AddChild(TreeNode2, menu.P_descricao)
         else if (menu.P_tipo_menu = 4) then
            TreeNode4 := tv_Menu.Items.AddChild(TreeNode3, menu.P_descricao)
         else if (menu.P_tipo_menu = 5) then
            TreeNode5 := tv_Menu.Items.AddChild(TreeNode4, menu.P_descricao)
         else if (menu.P_tipo_menu = 6) then
            TreeNode6 := tv_Menu.Items.AddChild(TreeNode5, menu.P_descricao)
         else if (menu.P_tipo_menu = 7) then
            TreeNode7 := tv_Menu.Items.AddChild(TreeNode6, menu.P_descricao)
         else if (menu.P_tipo_menu = 8) then
            TreeNode8 := tv_Menu.Items.AddChild(TreeNode7, menu.P_descricao)
         else if (menu.P_tipo_menu = 9) then
            TreeNode9 := tv_Menu.Items.AddChild(TreeNode8, menu.P_descricao);
      end;
      wi_Modulo       := menu.P_id_modulo;
      menu.Proximo;
   until menu.Final;
end;

procedure Tfrm_TAcesso.tv_MenuClick(Sender: TObject);
begin
   if tv_Menu.Selected.Text <> '' then
   begin
      ws_ItemMenu  := tv_Menu.Selected.Text;
      wb_alterar   := false;
      Mostra_Nivel;
      wb_alterar   := True;
   end
end;

procedure Tfrm_TAcesso.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
   frm_TAcesso.Release;
   frm_TAcesso := nil;
end;

procedure Tfrm_TAcesso.FormCreate(Sender: TObject);
begin
   wb_alterar := false;
   rgp_Nivel.ItemIndex := -1;
   Monta_Combo_Usu;
end;

procedure Tfrm_TAcesso.Monta_Combo_Usu;
var
   UsuarioCombo  : TCUsuario;
begin
   Combo_Usu.Items.Clear;
```

```
   UsuarioCombo  := TCUsuario.Create(nil);
   UsuarioCombo.SelecionaAtivos;
   UsuarioCombo.Primeiro;
   repeat
      UsuarioCombo.Coloca;

      Combo_Usu.Items.Add(UsuarioCombo.P_login);

      UsuarioCombo.Proximo
   until UsuarioCombo.Final;
end;

procedure Tfrm_TAcesso.btn_FecharClick(Sender: TObject);
begin
   Close;
end;

procedure Tfrm_TAcesso.Combo_UsuChange(Sender: TObject);
var
   UsuarioMenu  : TCUsuario;
begin
   UsuarioMenu  := TCUsuario.Create(nil);
   if Combo_Usu.Text <> '' then
   begin
      UsuarioMenu.Carregar(combo_Usu.Text);
      UsuarioMenu.Grupo.Selecionar(usuariomenu.P_id_grupo);
      Edit_Grupo.Text := UsuarioMenu.Grupo.P_descricao;
      monta_tv_Menu;
      ws_ItemMenu     := '';
      ws_MenuDescr    := '';
   end
   else
   begin
      rgp_Nivel.ItemIndex := -1;
   end;

end;

procedure Tfrm_TAcesso.FormKeyPress(Sender: TObject; var Key: Char);
begin
   if key = #13 then
   begin
      key := #0;
      monta_tv_Menu;
      ActiveControl := tv_Menu;
   end;
end;

procedure Tfrm_TAcesso.grid_MenuEnter(Sender: TObject);
begin
   if Combo_Usu.Text = '' then
   begin
      showmessage('Falta escolher o Usuario!!!');
      ActiveControl := Combo_Usu;
      Exit;
   end;
end;

procedure Tfrm_TAcesso.grid_MMenuEnter(Sender: TObject);
begin
   if Combo_Usu.Text = '' then
   begin
      showmessage('Falta escolher o Usuario!!!');
      ActiveControl := Combo_Usu;
      Exit;
   end;
end;

procedure Tfrm_TAcesso.Mostra_Nivel;
var
   MenuUsuario  : TCmenu;
   UsuarioMenu  : TCusuario;
   AcessoUsuario: TCAcesso;
begin
   MenuUsuario  := TCmenu.Create(nil);
   UsuarioMenu  := TCusuario.Create(nil);
   AcessoUsuario:= TCAcesso.Create(nil);
```

```
   MenuUsuario.Pesq_desc_Menu(ws_ItemMenu);
   UsuarioMenu.Carregar(combo_Usu.Text);

   if AcessoUsuario.SelecionarAcesso(UsuarioMenu.P_id_usuario,MenuUsuario.P_id_menu)
then
      rgp_Nivel.ItemIndex := 1
   else
      rgp_Nivel.ItemIndex := 0;

   MenuUsuario.FreeOnRelease;
   UsuarioMenu.FreeOnRelease;
   AcessoUsuario.FreeOnRelease;
end;

procedure Tfrm_TAcesso.rgp_NivelClick(Sender: TObject);
var
   MenuUsuario  : TCmenu;
   UsuarioMenu  : TCusuario;
   AcessoUsuario: TCAcesso;
begin
   if wb_alterar then
   if (rgp_nivel.ItemIndex >= 0 ) then
   begin
      MenuUsuario  := TCmenu.Create(nil);
      UsuarioMenu  := TCusuario.Create(nil);
      AcessoUsuario:= TCAcesso.Create(nil);
      if ws_ItemMenu = '' then
      begin
         showmessage('Selecione o Item do Menu a Alterar!!!');
      end
      else
      begin
         MenuUsuario.Pesq_desc_Menu(ws_ItemMenu);
         UsuarioMenu.Carregar(combo_Usu.Text);
         //---> cria ou altera o novo acesso no item do Menu
         if AcessoUsuario.PesquisaAcesso(UsuarioMenu.P_id_usuario,MenuUsuario.P_id_menu)
then
         begin

AcessoUsuario.Alterar(UsuarioMenu.P_id_usuario,rgp_nivel.ItemIndex+1,MenuUsuario.P_id_me
nu);
         end
         else
         begin
AcessoUsuario.Inserir(UsuarioMenu.P_id_usuario,rgp_nivel.ItemIndex+1,MenuUsuario.P_id_me
nu);
         end;
      end;
      MenuUsuario.FreeOnRelease;
      UsuarioMenu.FreeOnRelease;
      AcessoUsuario.FreeOnRelease;
   end;
end;

procedure Tfrm_TAcesso.FormActivate(Sender: TObject);
begin
   WindowState:= wsMaximized;
end;
procedure Tfrm_TAcesso.tv_MenuKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
   //tecla 1
   if ((key = VK_NumPad1) or (ORD(Key) = 49)) then
   begin
      key := 0;
      rgp_Nivel.ItemIndex:= 0;
      rgp_NivelClick(self);
   end;
   //tecla 2
   if ((key = VK_NumPad2) or (ORD(Key) = 50)) then
   begin
      key := 0;
      rgp_Nivel.ItemIndex:= 1;
      rgp_NivelClick(self);
   end;
end;
end.
```

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// PROJETO SCOS - CONTROLE DE ORDENS DE SERVIÇO                               //
// Tecnologia em Informática                                                  //
// Projeto de Conclusão de Curso                                              //
// 3° ano - 2005                                                              //
//                                                                            //
//      Desenvolvido por:                                                     //
//      - Marcelo Eduardo Hoepfner                                            //
//                                                                            //
//      - Helton Michel                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
```

2.2.57 unit uTela_ListaPadrao;

```
interface

uses
  Windows, Controls, ComCtrls, StdCtrls, ExtCtrls, fcButton, fcImgBtn,
  fcShapeBtn, Classes, forms;

type
  Tfrm_ListaPadrao = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btn_Incluir: TfcShapeBtn;
    btn_Altera: TfcShapeBtn;
    lv: TListView;
    Panel_BTN_SAIR: TPanel;
    btn_Sair: TfcShapeBtn;
    panel_listagem: TPanel;
    rgp_tipo: TRadioGroup;
    procedure btn_SairClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btn_IncluirClick(Sender: TObject);
    procedure lvMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frm_ListaPadrao    : Tfrm_ListaPadrao;
  wb_selecionoulinha : boolean;
  wl_Item            : TListItem;
  wi_limite          : Integer;

implementation

uses uMain, uDm_Base;

{$R *.dfm}

procedure Tfrm_ListaPadrao.btn_SairClick(Sender: TObject);
begin
   close;
end;

procedure Tfrm_ListaPadrao.FormActivate(Sender: TObject);
begin
   WindowState:= wsMaximized;
   wb_selecionoulinha:=false;
end;

procedure Tfrm_ListaPadrao.btn_IncluirClick(Sender: TObject);
begin
   frm_main.pnl_Menu.Enabled:= false;
```

```
end;

procedure Tfrm_ListaPadrao.lvMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  wb_selecionoulinha:=false;
  wl_Item := lv.GetItemAt(x, y);
  if (wl_Item <> nil) then
  begin
    if (wl_item.Selected) then
      wb_selecionoulinha:= true;
  end
  else
    wb_selecionoulinha:=false;
end;

procedure Tfrm_ListaPadrao.FormShow(Sender: TObject);
begin
   WindowState:= wsMaximized;
   ActiveControl := lv;
end;

end.
```