

**JOSÉ ALTAIR RIBEIRO DOS SANTOS
RAFAEL WINTER**

PROJETO SISTEMA PARA CONSULTÓRIO ODONTOLÓGICO

**CURITIBA
2004**

**JOSÉ ALTAIR RIBEIRO DOS SANTOS
RAFAEL WINTER**

**PROJETO SISTEMA PARA CONSULTÓRIO ODONTOLÓGICO
Volume II**

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.

Orientador: Prof.^º Mario de Paula Soares Filho

**CURITIBA
2004**

SUMÁRIO

| | |
|---------------------------------------|------------|
| 1 UNIT UMAIN | 1 |
| 2 UNIT ULOGIN | 5 |
| 3 UNIT USELPACIENTE | 7 |
| 4 UNIT UFICHA | 11 |
| 5 UNIT UDENTE | 36 |
| 6 UNIT UAGENDA | 44 |
| 7 UNIT UCADAGENDA | 50 |
| 8 UNIT ULOCAGENDA | 55 |
| 9 UNIT USELORC | 60 |
| 10 UNIT UCADORC | 64 |
| 11 UNIT UINSERIRSERVICO | 72 |
| 12 UNIT URELORCAMENTO | 76 |
| 13 UNIT USELSERVICOS | 79 |
| 14 UNIT UCADSERVICO | 83 |
| 15 UNIT UFELICITACOES | 85 |
| 16 UNIT URELCARTAO | 89 |
| 17 UNIT UATESTADO | 91 |
| 18 UNIT URELATESTADO | 93 |
| 19 UNIT UDADOSCLINICA | 95 |
| 20 UNIT UOPCOESUSUARIO | 97 |
| 21 UNIT UCADUSUARIO | 99 |
| 22 UNIT UPACIENTE | 102 |
| 23 UNIT UCOLPACIENTE | 106 |
| 24 UNIT UANAMNESE | 109 |
| 25 UNIT UEXAME | 111 |
| 26 UNIT UPLANOTRATAMENTO | 113 |
| 27 UNIT UCOLPLANO | 116 |
| 28 UNIT UODONTO | 118 |
| 29 UNIT UCOLODONTO | 121 |
| 30 UNIT UOPERACAO | 123 |

| | |
|--------------------------------------|------------|
| 31 UNIT UCOLOPERACAO..... | 127 |
| 32 UNIT URADIO | 129 |
| 33 UNIT UCOLRADIO | 132 |
| 34 UNIT UCOMPROMISSO | 134 |
| 35 UNIT UCOLCOMPROMISSO | 138 |
| 36 UNIT UORCAMENTO | 142 |
| 37 UNIT UCOLORCAMENTO..... | 147 |
| 38 UNIT USERVICOS..... | 149 |
| 39 UNIT UCOLSERVICOS | 151 |
| 40 UNIT UUSUARIO..... | 153 |

LISTA DE FIGURAS

| | |
|--------------------------------------|----|
| Figura 01 – FormPrincipal..... | 1 |
| Figura 02 – FormLogin | 5 |
| Figura 03 – FormSeleccionar | 7 |
| Figura 04 - FormFicha..... | 11 |
| Figura 05 – FormDente | 36 |
| Figura 06 - FormAgenda | 44 |
| Figura 07 - FormCadAgenda..... | 50 |
| Figura 08 – FormLocAgenda | 55 |
| Figura 09 - FormColOrcamento | 60 |
| Figura 10 – ForCadOrcamento | 64 |
| Figura 11 – FormInserirServico | 72 |
| Figura 12 - RelOrcamento | 76 |
| Figura 13 - FormServicos..... | 79 |
| Figura 14 - FormCadServico | 83 |
| Figura 15 - FormFelicitacoes..... | 85 |
| Figura 16 - RelCartao | 89 |
| Figura 17 - FormAtestado | 91 |
| Figura 18 - RelAtestado..... | 93 |
| Figura 19 – FormDadosClinica | 95 |
| Figura 20 - FormOpcoesUsuario | 97 |
| Figura 21 - FormCadUsuario..... | 99 |

1 UNIT UMAIN



FIGURA 01 – FORMPRINCIPAL

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, uAgenda, Buttons, Jpeg, uUsuario;

type
  TFormPrincipal = class(TForm)
    SBPacientes: TSpeedButton;
    LabPacientes: TLabel;
    SBAgenda: TSpeedButton;
    LabAgenda: TLabel;
    SBOrcamento: TSpeedButton;
    SBServicos: TSpeedButton;
    LabOrcamentos: TLabel;
    LabServicos: TLabel;
    SBUuarios: TSpeedButton;
    LabUsuarios: TLabel;
    Image1: TImage;
    SBSair: TSpeedButton;
    SBDadosClinica: TSpeedButton;
    LabDados: TLabel;
    LabAtestado: TLabel;
    SBAtestado: TSpeedButton;
    SBFelicitacoes: TSpeedButton;
    Label1: TLabel;
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```

procedure SBPacientesClick(Sender: TObject);
procedure SBAgendaClick(Sender: TObject);
procedure SBSairClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure SBUsuariosClick(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure SBServicosClick(Sender: TObject);
procedure SBorcamentoClick(Sender: TObject);
procedure SBDadosClinicaClick(Sender: TObject);
procedure SBAtestadoClick(Sender: TObject);
procedure SBFelicitacoesClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  Usuario: TUsuario;
end;

var
  FormPrincipal: TFormPrincipal;

implementation

uses uFicha, uSelPaciente, uDente, uCadUsuario, uOpcoesUsuario,
  uLogin, uSelServico, uRelOrcamento, uDadosClinica, uAtestado,
  uFelicitacoes, uSelOrc, uRelAtestado;

{$R *.dfm}

procedure TFormPrincipal.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FreeOnRelease;
end;

procedure TFormPrincipal.SBPacientesClick(Sender: TObject);
begin
  if not Usuario.PDadosPaciente Then
    MessageDlg('Você não tem permissão para acessar esta área!', mtError,
               [mbOk], 0)
  else
    begin
      if not assigned(FormSelecionar) then
        FormSelecionar := TFormSelecionar.Create(Application);

      FormSelecionar.ShowModal;
      FormSelecionar := nil;
    end;
end;

procedure TFormPrincipal.SBAgendaClick(Sender: TObject);
begin
  if Usuario.PAgenda Then
    begin
      if not assigned(FormAgenda) then
        FormAgenda := TFormAgenda.Create(nil);
      FormAgenda.ShowModal;
      FormAgenda := nil;
    end
  else
    MessageDlg('Você não tem permissão para acessar esta área!', mtError,
               [mbOk], 0);
end;

procedure TFormPrincipal.SBSairClick(Sender: TObject);
begin
  Usuario.Destroy;
  FormPrincipal.Close;
end;

procedure TFormPrincipal.FormCreate(Sender: TObject);
begin

```

```

SHORTDATEFORMAT := 'dd/mm/yyyy';

Usuario := TUuario.Create;
if not assigned(FormLogin) then
  FormLogin := TFormLogin.Create(nil);
FormLogin.ShowModal;
FormLogin := nil;
end;

procedure TFormPrincipal.SBUuariosClick(Sender: TObject);
begin
if not Usuario.PRoot then
  MessageDlg('Você não tem permissão para acessar esta área!', mterror,
             [mbOk], 0)
else
begin
  if not assigned(FormOpcoesUsuario) then
    FormOpcoesUsuario := TFormOpcoesUsuario.Create(nil);
  FormOpcoesUsuario.ShowModal;
  FormOpcoesUsuario := nil;
end;
end;

procedure TFormPrincipal.FormShow(Sender: TObject);
begin
  if Usuario.PLogin = '' Then
  begin
    Usuario.Destroy;
    Close;
  end;
end;

procedure TFormPrincipal.SBServicosClick(Sender: TObject);
begin
  if not Assigned(FormServicos) then
    FormServicos := TFormServicos.Create(nil);
  FormServicos.ShowModal;
  FormServicos := Nil;
end;

procedure TFormPrincipal.SBOrcamentoClick(Sender: TObject);
begin
  if not Usuario.POrcamento Then
    MessageDlg('Você não tem permissão para acessar esta área!', mterror,
               [mbOk], 0)
  else
begin
  if not Assigned(FormColOrcamento) then
    FormColOrcamento := TFormColOrcamento.Create(nil);
  FormColOrcamento.ShowModal;
  FormColOrcamento := Nil;
end;
end;

procedure TFormPrincipal.SBDadosClinicaClick(Sender: TObject);
begin
if not Usuario.PRoot then
  MessageDlg('Você não tem permissão para acessar esta área!', mterror,
             [mbOk], 0)
else
begin
  if not assigned(FormDadosClinica) then
    FormDadosClinica := TFormDadosClinica.Create(nil);
  FormDadosClinica.ShowModal;
  FormDadosClinica := nil;
end;
end;

procedure TFormPrincipal.SBAtestadoClick(Sender: TObject);
begin

```

```
if not Usuario.PAtestado then
  MessageDlg('Você não tem permissão para acessar esta área!', mterror,
             [mbOk], 0)
else
begin
  if not assigned(FormAtestado) then
    FormAtestado := TFormAtestado.Create(nil);
  FormAtestado.ShowModal;
  FormAtestado := nil;
end;
end;

procedure TFormPrincipal.SBFelicitacoesClick(Sender: TObject);
begin
if not Usuario.PFelicitacao then
  MessageDlg('Você não tem permissão para acessar esta área!', mterror,
             [mbOk], 0)
else
begin
  if not assigned(FormFelicitacoes) then
    FormFelicitacoes := TFormFelicitacoes.Create(nil);
  FormFelicitacoes.ShowModal;
  FormFelicitacoes := nil;
end;
end;
end.
```

2 UNIT ULOGIN



FIGURA 02 – FORMLOGIN

```

////////// ////////////////////////////////////////////////// ////////////////////////////////////////////////// //////////////////////////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA //////////////////////////////////////////////////
// Tecnologia em Informática //////////////////////////////////////////////////
// Projeto de Conclusão de Curso //////////////////////////////////////////////////
// 3º ano - 2004 //////////////////////////////////////////////////
// //////////////////////////////////////////////////
// Desenvolvido por: //////////////////////////////////////////////////
// - José Altair Ribeiro dos Santos //////////////////////////////////////////////////
// - Rafael Winter //////////////////////////////////////////////////
// //////////////////////////////////////////////////
unit uLogin;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;

type
  TFormLogin = class(TForm)
    LabLogin: TLabel;
    LabSenha: TLabel;
    EdLogin: TEdit;
    EdSenha: TEdit;
    BBOk: TBitBtn;
    BBCancelar: TBitBtn;
    procedure BBCancelarClick(Sender: TObject);
    procedure BBOkClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    procedure Fechar;
  public
    { Public declarations }
  end;

var
  FormLogin: TFormLogin;
  Contador: Integer;

implementation

uses uMain;

{$R *.dfm}

procedure TFormLogin.BBCancelarClick(Sender: TObject);

```

```
begin
  Fechar;
end;

procedure TFormLogin.BBOKClick(Sender: TObject);
begin
  if EdLogin.Text = '' Then
    begin
      MessageDlg('Digite um Login!', mtInformation, [mbOk], 0);
      Exit;
    end;
  FormPrincipal.Usuario.Carregar(EdLogin.Text);
  if FormPrincipal.Usuario.PLogin = EdLogin.Text Then
    begin
      if FormPrincipal.Usuario.PSenha = EdSenha.Text Then
        FormLogin.Close
      else
        begin
          MessageDlg('Senha Incorreta!', mtError, [mbok], 0);
          EdSenha.Text := '';
          EdSenha.SetFocus;
          Inc(Contador);
          if Contador >= 3 Then
            Fechar;
        end;
    end
  else
    begin
      MessageDlg('Usuário não encontrado!', mtError, [mbok], 0);
      Inc(Contador);
      if Contador >= 3 Then
        Fechar;
    end;
end;

procedure TFormLogin.Fechar;
begin
  FormPrincipal.Usuario.PLogin := '';
  Close;
end;

procedure TFormLogin.FormCreate(Sender: TObject);
begin
  Contador := 0;
end;

procedure TFormLogin.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FreeOnRelease;
end;

end.
```

3 UNIT USELPACIENTE

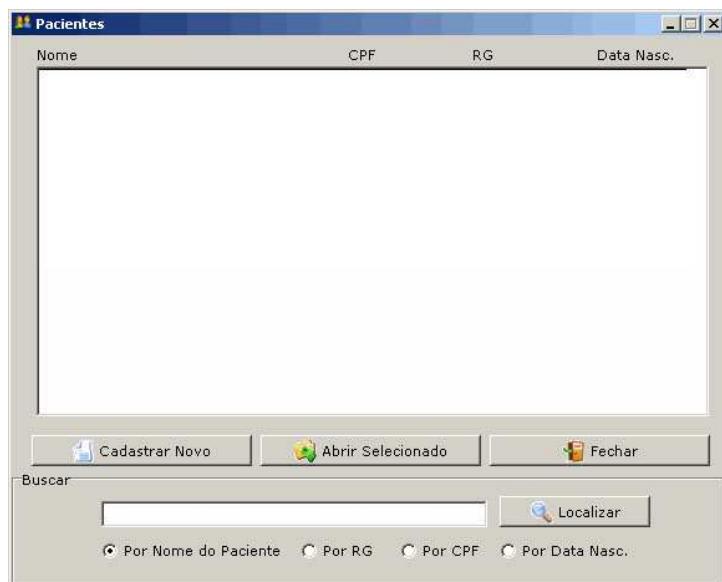


FIGURA 03 – FORMSELECCIONAR

```
///////////////////////////////
//  

// PROJETO CLÍNICA ODONTOLÓGICA  

// Tecnologia em Informática  

// Projeto de Conclusão de Curso  

// 3º ano - 2004  

//  

// Desenvolvido por:  

// - José Altair Ribeiro dos Santos  

// - Rafael Winter  

//  

/////////////////////////////
unit uSelPaciente;  

interface  

uses  

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  

  Dialogs, StdCtrls, Grids, Buttons, Mask, UColPaciente, ComCtrls;  

type  

  TFormSeleccionar = class(TForm)
    sgcolecao: TStringGrid;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    BBNovo: TBitBtn;
    BBAbrir: TBitBtn;
    BBLocalizar: TBitBtn;
    EdLocalizar: TMaskEdit;
    GBBuscar: TGroupBox;
    RBNome: TRadioButton;
    RBCPF: TRadioButton;
    RBRG: TRadioButton;
    RBDataNasc: TRadioButton;
    BBFechar: TBitBtn;
    DTData: TDateTimePicker;
    procedure FormCreate(Sender: TObject);
```

```

procedure BBAbrirClick(Sender: TObject);
procedure sgcolecaoSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure sgcolecaoDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure BBNovoClick(Sender: TObject);
procedure BBLocalizarClick(Sender: TObject);
procedure RBNomeClick(Sender: TObject);
procedure RBRGClick(Sender: TObject);
procedure RBCPFClick(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure RBDataNascClick(Sender: TObject);
procedure sgcolecaoDblClick(Sender: TObject);
procedure BBFecharClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  Cole: TColPaciente;
  procedure ColocaDados;
  procedure AbrirCliente;
end;

var
  FormSeleccionar: TFormSeleccionar;
  x: integer;

implementation

{$R *.dfm}

uses
  uFicha;

procedure TFormSeleccionar.FormCreate(Sender: TObject);
var
  a: integer;
begin
  SGColecao.ColWidths[0] := 0;
  SGColecao.ColWidths[1] := 240;
  SGColecao.ColWidths[2] := 100;
  SGColecao.ColWidths[3] := 100;
  SGColecao.ColWidths[4] := 80;
  Cole := TColPaciente.Create(nil);
  Cole.Seleciona;
  ColocaDados;
end;

procedure TFormSeleccionar.BBAbrirClick(Sender: TObject);
begin
  AbrirCliente;
end;

procedure TFormSeleccionar.sgcolecaoSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
begin
  x := arow;
end;

procedure TFormSeleccionar.sgcolecaoDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  if gdFocused in State then
    with (Sender as TStringGrid).Canvas do
  begin
    Brush.Color:=clGreen;
    FillRect(Rect);
    Font.Color := ClWhite;
    Font.Style := [fsUnderLine];
  end;
end;

```

```

    TextOut(Rect.Left + 3, Rect.Top + 3, SgColecao.Cells[Acol,ARow]);
end;
TStringGrid(Sender).Canvas.Pen.Color := clBlack;
TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormSeleccionar.BBNovoClick(Sender: TObject);
begin
  if not assigned(FormFicha) then
    FormFicha := TFormFicha.Create(Self);

  FormFicha.GBMenu.Enabled := False;
  FormFicha.EdData_Nasc.Date := Now;
  FormFicha.EdIniTratamento.Date := Now;
  FormFicha.EdCodigo.Text := '';
  FormFicha.ShowModal;
  FormFicha := nil;
end;

procedure TFormSeleccionar.ColocaDados;
var a :integer;
begin
  sgcolecao.Cells[0,0] := '';
  sgcolecao.Cells[1,0] := '';
  sgcolecao.Cells[2,0] := '';
  sgcolecao.Cells[3,0] := '';
  sgcolecao.Cells[4,0] := '';
  sgcolecao.RowCount := 1;
  a := 0;
  Cole.Primeiro;
  While Not Cole.Final do
    begin
      Cole.Coloa;
      sgcolecao.Cells[0,a] := InttoStr(Cole.CCodigo);
      sgcolecao.Cells[1,a] := Cole.CNome;
      sgcolecao.Cells[2,a] := ' ' + Cole.CCPF + ' ';
      sgcolecao.Cells[3,a] := ' ' + Cole.CRG + ' ';
      sgcolecao.Cells[4,a] := ' ' + Cole.CDataNasc + ' ';
      Cole.Proximo;
      sgcolecao.RowCount := a+1;
      inc(a);
    end;
end;

procedure TFormSeleccionar.BBLocalizarClick(Sender: TObject);
var
  campo: string;
begin
  If (RBRG.Checked) Then
    campo := 'pac_rg'
  Else
  If (RBCPF.Checked) Then
    campo := 'pac_cpf'
  Else
  If (RBNome.Checked) Then
    campo := 'pac_nome'
  Else
  If (RBDataNasc.Checked) Then
    campo := 'pac_datanasc';

  if campo = 'pac_datanasc' then
    Cole.Procura(campo, DateToStr(DTData.Date))
  else
    Cole.Procura(campo, EdLocalizar.Text);
  ColocaDados;
end;

procedure TFormSeleccionar.RBNomeClick(Sender: TObject);

```

```

begin
  EdLocalizar.EditMask := '';
  EdLocalizar.Text := '';
  DTData.Visible := False;
  EdLocalizar.Visible := True;
end;

procedure TFormSeleccionar.RBRGClick(Sender: TObject);
begin
  EdLocalizar.EditMask := '9.999.999-9;1;_';
  EdLocalizar.Text := '';
  DTData.Visible := False;
  EdLocalizar.Visible := True;
end;

procedure TFormSeleccionar.RBCPFCClick(Sender: TObject);
begin
  EdLocalizar.EditMask := '999.999.999-99;1;_';
  EdLocalizar.Text := '';
  DTData.Visible := False;
  EdLocalizar.Visible := True;
end;

procedure TFormSeleccionar.FormShow(Sender: TObject);
begin
  RBNome.Checked := true;
  RBNome.OnClick(Sender);
end;

procedure TFormSeleccionar.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Cole.FreeOnRelease;
  FormSeleccionar.FreeOnRelease;
end;

procedure TFormSeleccionar.RBDataNascClick(Sender: TObject);
begin
  DTData.Date := Now;
  DTData.Visible := True;
  EdLocalizar.Visible := False;
end;

procedure TFormSeleccionar.AbrirCliente;
begin
  if not assigned(FormFicha) then
    FormFicha := TFormFicha.Create(nil);

  FormFicha.CarregarDados(StrToInt(SgColecao.Cells[0, x]));
  FormFicha.ShowModal;
  FormFicha:=nil;
end;

procedure TFormSeleccionar.sgcolecaoDblClick(Sender: TObject);
begin
  AbrirCliente;
end;

procedure TFormSeleccionar.BBFecharClick(Sender: TObject);
begin
  Close;
end;

end.

```

4 UNIT UFICHA

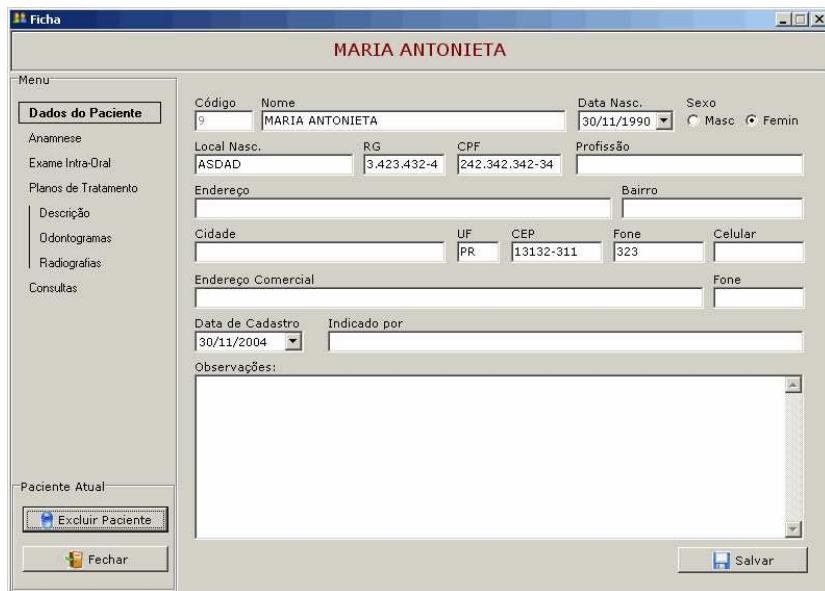


FIGURA 04 - FORMFICHA

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
//      Desenvolvido por:  
//      - José Altair Ribeiro dos Santos  
//      - Rafael Winter  
//  
//  
unit uFicha;  
  
interface  
  
uses  
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
  Dialogs, ComCtrls, StdCtrls, ExtCtrls, jpeg, Grids, Mask, Buttons, UPaciente,  
  ExtDlgs, uColCompromisso, uCompromisso, uAgenda;  
  
type  
  TFormFicha = class(TForm)  
    ednome: TEdit;  
    LabNome: TLabel;  
    edendereco: TEdit;  
    LabProfissao: TLabel;  
    edprofissao: TEdit;  
    LabEndereco: TLabel;  
    LabBairro: TLabel;  
    edbairro: TEdit;  
    LabCidade: TLabel;  
    edcidade: TEdit;  
    LabUF: TLabel;  
    eduf: TEdit;  
    LabCEP: TLabel;  
    LabFone: TLabel;  
    edfone1: TEdit;  
    LabEndCom: TLabel;  
    edendcom: TEdit;
```

```
LabFoneCom: TLabel;
edfonecom: TEdit;
LabDataNasc: TLabel;
LabLocalNasc: TLabel;
edlocal_nasc: TEdit;
Label14: TLabel;
Label15: TLabel;
edindicacao: TEdit;
PanP01: TPanel;
LabP01: TLabel;
RBP01Sim: TRadioButton;
RBP01Nao: TRadioButton;
PanP02: TPanel;
LabP02: TLabel;
PanP07: TPanel;
LabP07: TLabel;
RBP07Sim: TRadioButton;
RBP07Nao: TRadioButton;
PanP10: TPanel;
LabP10: TLabel;
RBP10Sim: TRadioButton;
RBP10Nao: TRadioButton;
LabP10Qual: TLabel;
EdP10: TEdit;
PanP11: TPanel;
LabP11: TLabel;
CBP1101: TCheckBox;
CBP1102: TCheckBox;
CBP1103: TCheckBox;
CBP1104: TCheckBox;
CBP1105: TCheckBox;
CBP1106: TCheckBox;
CBP1107: TCheckBox;
CBP1108: TCheckBox;
LabP11Outros: TLabel;
EdP11: TEdit;
LabObs: TLabel;
edobs: TMemo;
PanP03: TPanel;
LabP03: TLabel;
LabP03Quais: TLabel;
RBP03Sim: TRadioButton;
RBP03Nao: TRadioButton;
EdP03: TEdit;
PanP04: TPanel;
LabP04: TLabel;
LabP04Quais: TLabel;
RBP04Sim: TRadioButton;
RBP04Nao: TRadioButton;
EdP04: TEdit;
PanP05: TPanel;
LabP05: TLabel;
LabP05Quais: TLabel;
RBP05Nao: TRadioButton;
RBP05Sim: TRadioButton;
EdP05: TEdit;
PanP06: TPanel;
LabP06: TLabel;
RBP06Sim: TRadioButton;
RBP06Nao: TRadioButton;
PanP09: TPanel;
LabP09: TLabel;
EdP09: TEdit;
PanHigiene: TPanel;
LabHigiene: TLabel;
RBHigieneNormal: TRadioButton;
RBHigieneRegular: TRadioButton;
RBHigieneDeficiente: TRadioButton;
PanHalitose: TPanel;
LabHalitose: TLabel;
RBHalitoseAusente: TRadioButton;
```

```
RBHalitoseModerada: TRadioButton;
RBHalitoseForte: TRadioButton;
PanTartaro: TPanel;
LabTartaro: TLabel;
RBTartaroAusente: TRadioButton;
RBTartaroPouco: TRadioButton;
RBTartaroMuito: TRadioButton;
PanGengiva: TPanel;
LabGengiva: TLabel;
RBGengivaNormal: TRadioButton;
RBGengivaGengivite: TRadioButton;
RBGengivaPeriodontite: TRadioButton;
PanMucosa: TPanel;
LabMuscosa: TLabel;
RBMucosaNormal: TRadioButton;
RBMucosaAlterada: TRadioButton;
LabLingua: TLabel;
LabPalato: TLabel;
LabAssocalhoBucal: TLabel;
LabLabios: TLabel;
EdLingua: TEdit;
EdPalato: TEdit;
EdAssocalho: TEdit;
EdLabios: TEdit;
LabExameObs: TLabel;
MemoExameObs: TMemo;
ImgOdontograma: TImage;
LabObsRadio: TLabel;
MemoRadio: TMemo;
LabCodigo: TLabel;
edcodigo: TEdit;
LabRG: TLabel;
LabCPF: TLabel;
MemoOdontogramaObs: TMemo;
LabObsOdontograma: TLabel;
SGPlano: TStringGrid;
LabPlanoHistData: TLabel;
LabPlanoHistDescricao: TLabel;
edrg: TMaskEdit;
edcpf: TMaskEdit;
LabCelular: TLabel;
edfone2: TEdit;
CBP02Frio: TCheckBox;
CBP02Doces: TCheckBox;
CBP02Outros: TCheckBox;
EdP02: TEdit;
LabP02Aque: TLabel;
PanP08: TPanel;
LabP08: TLabel;
RBP08Sim: TRadioButton;
RBP08Nao: TRadioButton;
LabPlanoData: TLabel;
LabPlanoDescricao: TLabel;
EdPlanoDescricao: TEdit;
GBPlanoHistorico: TGroupBox;
PanImgOdontograma: TPanel;
LabOdontoData: TLabel;
SGOdontograma: TStringGrid;
LabOdontoDescricao: TLabel;
GBOdontoInserir: TGroupBox;
ImgRadio: TImage;
EdRadioDesc: TEdit;
SGConsultas: TStringGrid;
GBConsultas: TGroupBox;
GBMenu: TGroupBox;
LabMDados: TLabel;
LabMAnamnese: TLabel;
LabMExame: TLabel;
PanDados: TPanel;
LabSexo: TLabel;
RBMasculino: TRadioButton;
```

```
RBFeminino: TRadioButton;
PanAnamnese: TPanel;
PanRadio: TPanel;
PanOdontograma: TPanel;
BBInserirOdonto: TBitBtn;
BBExcluirOdonto: TBitBtn;
PanExameIntraOral: TPanel;
PanPlanoTratamento: TPanel;
BBApcionarConsulta: TBitBtn;
BBExcluirConsulta: TBitBtn;
PanConsultas: TPanel;
LabMPlanos: TLabel;
LabMDescricao: TLabel;
LabModontogramas: TLabel;
LabMRadiografias: TLabel;
LabMConsultas: TLabel;
ShMLinha: TShape;
ShSeleciona: TShape;
EdCep: TMaskEdit;
LabPlanoObs: TLabel;
MemoPlanoObs: TMemo;
LbRadioDesc: TLabel;
LbRadioData: TLabel;
SGRadio: TStringGrid;
PanImagen: TPanel;
LBColRadioData: TLabel;
LBColRadioDesc: TLabel;
BBRadioAdicionar: TBitBtn;
BBRadioSalvar: TBitBtn;
BBRadioExcluir: TBitBtn;
BBRadioAbrir: TBitBtn;
odradio: TOpenPictureDialog;
GBDados: TGroupBox;
lbnome: TLabel;
GBOdontoAtual: TGroupBox;
LabDataOdontograma: TLabel;
LabDescOdontograma: TLabel;
EdOdontogramaDescricao: TEdit;
GBOdontoOpcoes: TGroupBox;
BBSalvarOdonto: TBitBtn;
EdData_Nasc: TDateTimePicker;
EdIniTratamento: TDateTimePicker;
EdPlanoData: TDateTimePicker;
EdRadioData: TDateTimePicker;
BBSalvarDados: TBitBtn;
BBExcluir: TBitBtn;
BBSalvarExame: TBitBtn;
BBSalvarAnamnese: TBitBtn;
BBSalvarPlano: TBitBtn;
BBInserirPlano: TBitBtn;
PanSexo: TPanel;
BBFchar: TBitBtn;
EdOdontogramaData: TDateTimePicker;
BBAbrirPlano: TBitBtn;
BBSelecionarOdonto: TBitBtn;
GBPlanoOpcoes: TGroupBox;
GBPlanoAtual: TGroupBox;
GRadioAtual: TGroupBox;
GRadioOpcoes: TGroupBox;
GRadiografias: TGroupBox;
BBPlanoExcluir: TBitBtn;
LabConsultaData: TLabel;
LabConsultaHora: TLabel;
LabConsultaDesc: TLabel;
LabConsultaSituacao: TLabel;
LabConsultaDuracao: TLabel;
BBAalterarConsulta: TBitBtn;
GRetornoConsulta: TGroupBox;
PnNome: TPanel;
procedure FormCreate(Sender: TObject);
procedure GBMenuMouseMove(Sender: TObject; Shift: TShiftState; X,
```

```

    Y: Integer);
procedure LabMDadosMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure SGPlanoSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure SGPlanoDblClick(Sender: TObject);
procedure SGRadioSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure BBRadioAbrirClick(Sender: TObject);
procedure BBRadioAdicionarClick(Sender: TObject);
procedure edfoneKeyPress(Sender: TObject; var Key: Char);
procedure edufKeyPress(Sender: TObject; var Key: Char);
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure ImgOdontogramaMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
procedure BBSalvarDadosClick(Sender: TObject);
procedure BBSalvarExameClick(Sender: TObject);
procedure BBSalvarAnamneseClick(Sender: TObject);
procedure BBSalvarPlanoClick(Sender: TObject);
procedure BBInserirPlanoClick(Sender: TObject);
procedure BBAbrirPlanoClick(Sender: TObject);
procedure SGOdontogramaSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure SGOdontogramaDblClick(Sender: TObject);
procedure BBInserirOdontoClick(Sender: TObject);
procedure BBExcluirOdontoClick(Sender: TObject);
procedure BBSalvarOdontoClick(Sender: TObject);
procedure SGPlanoDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure BBSeleccionarOdontoClick(Sender: TObject);
procedure SGRadioDblClick(Sender: TObject);
procedure BBRadioSalvarClick(Sender: TObject);
procedure BBPlanoExcluirClick(Sender: TObject);
procedure BBRadioExcluirClick(Sender: TObject);
procedure BBExcluirClick(Sender: TObject);
procedure SGConsultasSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure BBExcluirConsultaClick(Sender: TObject);
procedure BBAgregarConsultaClick(Sender: TObject);
procedure BBAalterarConsultaClick(Sender: TObject);
procedure BBFeharClick(Sender: TObject);
procedure ednameChange(Sender: TObject);

private
  { Private declarations }

public
  procedure SalvarDados;
  procedure CarregarDados(codigo: integer);
  procedure ExcluirPaciente;

  procedure MenuSelecciona;
  procedure MudarSeleccionado(Lab: TLabel);

  procedure SalvarAnamnese;
  procedure LerAnamnese;

  procedure SalvarExame;
  procedure LerExame;

  procedure SalvarPlano(Est: Integer);
  procedure LerPlano(Codigo: Integer);
  procedure ColocaPlano;

  procedure SalvarOdontograma(Est: Integer);
  procedure LerOdontograma(Codigo: Integer);
  procedure LerOdontoOperacoes;
  procedure ColocaOdontograma;

  procedure SalvarRadio;

```

```

procedure GravarDadosRadio;
procedure LerRadio(Codigo: Integer);
procedure ColocaRadio;

procedure LerConsultas;

procedure SalvarTelaAtual(Tela: Integer);
end;

var
  FormFicha: TFormFicha;
  Paciente: TPaciente;
  Consultas: TColCompromisso;
  Compromisso: TCompromisso;
  Selecao, Selecionado, CodigoPlano: Integer;
  CodigoRadio, CodigoOdonto, CodigoConsulta: Integer;
  SalvarDadosTela, PnDados, PnAnamnese, PnExame: Boolean;
  PnPlano, PnOdonto, PnRadio, PnConsulta: Boolean;

implementation

uses uDente, uCadAgenda, uMain, uSelPaciente;

{$R *.dfm}

procedure TFormFicha.FormCreate(Sender: TObject);
var
  a: Integer;
  nome: String;
begin
  for a:= 0 to TFormFicha(Sender).ComponentCount -1 do
    begin
      nome := TFormFicha(Sender).Components[a].Name;
      if TFormFicha(Sender).Components[a] is TEdit Then
        TEdit(TFormFicha(Sender).Components[a]).OnChange := EdNome.OnChange;
      if TFormFicha(Sender).Components[a] is TMemo Then
        TMemo(TFormFicha(Sender).Components[a]).OnChange := EdNome.OnChange;
      if TFormFicha(Sender).Components[a] is TMaskEdit Then
        TMaskEdit(TFormFicha(Sender).Components[a]).OnChange :=
          EdNome.OnChange;
      if TFormFicha(Sender).Components[a] is TDateTimePicker Then
        TDateTimePicker(TFormFicha(Sender).Components[a]).OnChange :=
          EdNome.OnChange;
      if TFormFicha(Sender).Components[a] is TRadioButton Then
        TRadioButton(TFormFicha(Sender).Components[a]).OnClick :=
          EdNome.OnChange;
      if TFormFicha(Sender).Components[a] is TCheckBox Then
        TCheckBox(TFormFicha(Sender).Components[a]).OnClick :=
          EdNome.OnChange;
    end;
  Paciente := TPaciente.Create(FormFicha);
  Consultas := TColCompromisso.Create(Self);
  SalvarDadosTela := False;

  //PAINÉIS ACIONADOS
  PnDados := True;
  PnAnamnese := False;
  PnExame := False;
  PnPlano := False;
  PnOdonto := False;
  PnRadio := False;
  PnConsulta := False;
  Selecionado := 1;
end;

procedure TFormFicha.Salvardados;
begin
  if (EdNome.Text = '') Then
    begin
      MessageDlg('Nome do paciente vazio!', mtError, [mbOk], 0);
      Abort;
    end;
end;

```

```

        end
    else
    if Paciente.Verificar(EdCPF.Text, EdRG.Text, Paciente.Pcodigo) Then
        begin
            MessageDlg('CPF ou RG já cadastrado!', mtError, [mbOk],0);
            Abort;
        end
    else
    begin
        Paciente.Pnome := EdNome.Text;
        Paciente.PRG := edrg.text;
        Paciente.PCPF := edcpf.text;
        if RBMasculino.Checked then
            Paciente.Psexo := 'M'
        else
            Paciente.Psexo := 'F';
        Paciente.Pprofissao := EdProfissao.text;
        Paciente.PEndereco := EdEndereco.Text;
        Paciente.PBairro := EdBairro.Text;
        Paciente.PCidade := EdCidade.Text;
        Paciente.PUF := EdUf.text;
        Paciente.PCEP := EdCep.Text;
        Paciente.PFone1 := EdFone1.Text;
        Paciente.PFone2 := EdFone2.Text;
        Paciente.Pend_com := EdEndcom.Text;
        Paciente.Pfonecom := EdFonecom.Text;
        Paciente.PData_Nasc := EdData_nasc.Date;
        Paciente.PLocal_Nasc := EdLocal_nasc.Text;
        Paciente.Pini_tratamento := EdIniTratamento.Date;
        Paciente.PIndicacao := EdIndicacao.Text;
        Paciente.PObs := EdObs.Text;
        if Paciente.Pcodigo <> 0 Then
            Paciente.Salvar(1)
        else
            Paciente.Salvar(0);
        Edcodigo.text := IntToStr(Paciente.Pcodigo);
        LBNome.Caption := Paciente.PNome;
    end;
    SalvarDadosTela := False;
end;

procedure TFormFicha.CarregarDados(codigo: integer);
begin
    Paciente.Pcodigo := Codigo;
    Paciente.Carregar;
    EdCodigo.Text := IntToStr(Paciente.Pcodigo);
    EdNome.Text := Paciente.PNome;
    EdRG.Text := Paciente.PRG;
    EdCPF.Text := Paciente.PCPF;
    EdProfissao.Text := Paciente.PProfissao;
    EdEndereco.Text := Paciente.PEndereco;
    EdBairro.Text := Paciente.PBairro;
    EdCidade.Text := Paciente.PCidade;
    EdUF.Text := Paciente.Puf;
    EdCEP.Text := Paciente.Pcep;
    EdData_Nasc.Date := Paciente.Pdata_nasc;
    EdLocal_Nasc.Text := Paciente.Plocal_nasc;
    EdIniTratamento.Date := Paciente.Pini_tratamento;
    EdIndicacao.Text := Paciente.PIndicacao;
    EdObs.Text := Paciente.PObs;
    EdFone1.Text := Paciente.PFone1;
    EdFone2.Text := Paciente.PFone2;
    EdEndCom.Text := Paciente.Pend_com;
    EdFoneCom.Text := Paciente.Pfonecom;
    If Paciente.Psexo = 'M' Then
        rbmasculino.Checked := true;
    If Paciente.Psexo = 'F' Then
        rbfeminino.Checked := true;
    SalvarDadosTela := False;
end;

```

```

procedure TFormFicha.GBMenuMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
  if (y < LabMAnamnese.Top - 4) then
    begin
      ShSeleciona.Top := LabMDados.Top - 4;
      Selecao := 1;
    end
  else
    if (y < LabMExame.Top - 4) then
      begin
        ShSeleciona.Top := LabMAnamnese.Top - 4;
        Selecao := 2;
      end
    else
      if (y < LabMPlanos.Top - 4) then
        begin
          ShSeleciona.Top := LabMExame.Top - 4;
          Selecao := 3;
        end
      else
        if (y < LabMDescricao.Top - 4) then
          begin
            ShSeleciona.Top := LabMPlanos.Top - 4;
            Selecao := 4;
          end
        else
          if (y < LabMOdontogramas.Top - 4) then
            begin
              ShSeleciona.Top := LabMDescricao.Top - 4;
              Selecao := 5;
            end
          else
            if (y < LabMRadiografias.Top - 4) then
              begin
                ShSeleciona.Top := LabMOdontogramas.Top - 4;
                Selecao := 6;
              end
            else
              if (y < LabMConsultas.Top - 4) then
                begin
                  ShSeleciona.Top := LabMRadiografias.Top - 4;
                  Selecao := 7;
                end
              else
                begin
                  ShSeleciona.Top := LabMConsultas.Top - 4;
                  Selecao := 8;
                end;
            end;
        end;
  end;

procedure TFormFicha.MenuSeleciona;
var
  Permissao: Boolean;
begin
  Permissao := True;
  if SalvarDadosTela Then
    SalvarTelaAtual(Selecionado);
  SalvarDadosTela := False;

  case Selecao of
    2: if not FormPrincipal.Usuario.PAnamnese Then
        Permissao := False;
    3: if not FormPrincipal.Usuario.PExame Then
        Permissao := False;
    4, 5, 6, 7: if not FormPrincipal.Usuario.PPlano Then
        Permissao := False;
    8: if not FormPrincipal.Usuario.PAgenda Then
        Permissao := False;
  end;
  if not Permissao then

```

```

begin
  MessageDlg('Você não tem permissão para acessar esta área!', mterror,
             [mbOk], 0);
  Selecao := Seleccionado;
end;

case Selecao of
  1: begin
    PanDados.BringToFront;
    MudarSelecionaldo(LabMDados);
  end;
  2: begin
    PanAnamnese.BringToFront;
    MudarSelecionaldo(LabMANamnese);
    if not PnAnamnese Then
      begin
        PnAnamnese := true;
        LerAnamnese;
      end;
  end;
  3: begin
    PanExameIntraOral.BringToFront;
    MudarSelecionaldo(LabMExame);
    if not PnExame Then
      begin
        PnExame := True;
        LerExame;
      end;
  end;
  4, 5: begin
    FormFicha.PanPlanoTratamento.BringToFront;
    MudarSelecionaldo(LabMDescricao);
    LabMPlanos.Font.Style := [fsBold];
    if not PnPlano Then
      begin
        PnPlano := True;
        ColocaPlano;
      end;
  end;
  6: begin
    if (Paciente.PlanoAtual.PCodigo <> 0) Then
      begin
        PanOdontograma.BringToFront;
        MudarSelecionaldo(LabModontogramas);
        LabMPlanos.Font.Style := [fsBold];
        if not PnOdonto Then
          begin
            ImgOdontograma.Picture.
              LoadFromFile(ExtractFilePath(Application.ExeName) +
                           'dente\odonto.bmp');
            EdOdontogramaData.Date := Now;
            EdOdontogramaDescricao.Text := '';
            MemoOdontogramaObs.Text := '';
            GBODontoAtual.Enabled := False;
            ColocaOdontograma;
            PnOdonto := True;
          end;
      end
    else
      begin
        MessageDlg('Não há Plano de Tratamento Seleccionado', mterror, [mbok], 0);
        Selecao := 4;
        MenuSeleciona;
      end;
  end;
  7: begin
    if (Paciente.PlanoAtual.PCodigo <> 0) Then
      begin
        PanRadio.BringToFront;
        MudarSelecionaldo(LabMRadiografias);
        LabMPlanos.Font.Style := [fsBold];
      end;
  end;
end;

```

```

        if not PnRadio Then
            begin
                PnRadio := True;
                ColocaRadio;
                EdRadioData.Date := Now;
                EdRadioDesc.Text := '';
                MemoRadio.Text := '';
                ImgRadio.Picture.
                    LoadFromFile(ExtractFilePath(Application.ExeName) +
                        'imagens\padrao.jpg');
                GBRadioAtual.Enabled := False;
            end;
        end
    else
        begin
            MessageDlg('Não há Plano de Tratamento Selecionado', mtError, [mbok],0);
            Selecao := 4;
            MenuSeleciona;
        end;
    end;
8: begin
    PanConsultas.BringToFront;
    MudarSelecionado(LabMConsultas);
    if not PnConsulta Then
        begin
            LerConsultas;
            PnConsulta := True;
        end;
    end;
    Selecionado := Selecao;
end;

procedure TFormFicha.LabMDadosMouseDown(Sender: TObject;
    Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    MenuSeleciona;
end;

procedure TFormFicha.MudarSelecionado(Lab: TLabel);
begin
    LabMDados.Font.Style := [];
    LabMAnamnese.Font.Style := [];
    LabMExame.Font.Style := [];
    LabMPlanos.Font.Style := [];
    LabMDescricao.Font.Style := [];
    LabModontogramas.Font.Style := [];
    LabMRadiografias.Font.Style := [];
    LabMConsultas.Font.Style := [];
    Lab.Font.Style := [fsBold];
end;

procedure TFormFicha.SalvarAnamnese;
var
    P02: Integer;
    P11: String;
begin
    //Pergunta 01
    if RBP01Sim.Checked Then
        Paciente.Anamnese.PAnP01 := true
    Else
        Paciente.Anamnese.PAnP01 := false;
    //Pergunta 02
    P02 := 0;
    if CBP02Frio.Checked Then
        P02 := P02 + 1;
    if CBP02Doces.Checked Then
        P02 := P02 + 2;
    if CBP02Outros.Checked Then
        P02 := P02 + 4;
    Paciente.Anamnese.PAnP02 := IntToStr(P02);

```

```

Paciente.Anamnese.PAnP02q := EdP02.Text;
//Pergunta 03
if RBP03Sim.Checked Then
    Paciente.Anamnese.PAnP03 := true
Else
    Paciente.Anamnese.PAnP03 := false;
Paciente.Anamnese.PAnP03q := EdP03.Text;
//Pergunta 04
if RBP04Sim.Checked Then
    Paciente.Anamnese.PAnP04 := true
Else
    Paciente.Anamnese.PAnP04 := false;
Paciente.Anamnese.PAnP04q := EdP04.Text;
//Pergunta 05
if RBP05Sim.Checked Then
    Paciente.Anamnese.PAnP05 := true
Else
    Paciente.Anamnese.PAnP05 := false;
Paciente.Anamnese.PAnP05q := EdP05.Text;
//Pergunta 06
if RBP06Sim.Checked Then
    Paciente.Anamnese.PAnP06 := true
Else
    Paciente.Anamnese.PAnP06 := false;
//Pergunta 07
if RBP07Sim.Checked Then
    Paciente.Anamnese.PAnP07 := true
Else
    Paciente.Anamnese.PAnP07 := false;
//Pergunta 08
if RBP08Sim.Checked Then
    Paciente.Anamnese.PAnP08 := true
Else
    Paciente.Anamnese.PAnP08 := false;
//Pergunta 09
Paciente.Anamnese.PAnP09q := EdP09.Text;
//Pergunta 10
if RBP10Sim.Checked Then
    Paciente.Anamnese.PAnP10 := true
Else
    Paciente.Anamnese.PAnP10 := false;
P11 := '00000000';
if CBP1101.Checked Then
    P11[1] := '1';
if CBP1102.Checked Then
    P11[2] := '1';
if CBP1103.Checked Then
    P11[3] := '1';
if CBP1104.Checked Then
    P11[4] := '1';
if CBP1105.Checked Then
    P11[5] := '1';
if CBP1106.Checked Then
    P11[6] := '1';
if CBP1107.Checked Then
    P11[7] := '1';
if CBP1108.Checked Then
    P11[8] := '1';
Paciente.Anamnese.PAnP11 := P11;
Paciente.Anamnese.PAnP11q := EdP11.Text;
Paciente.Anamnese.Salvar(Paciente.Pcodigo);
SalvarDadosTela := False;
end;

procedure TFormFicha.LerAnamnese;
var
  P11: String;
  P02: Integer;
begin

```

```

Paciente.Anamnese.Buscar(Paciente.Pcodigo);
//Pergunta 01
if Paciente.Anamnese.PAnP01 = true Then
    RBP01Sim.Checked := true
Else
    RBP01Nao.Checked := true;
//Pergunta 02
CBP02Frio.Checked := false;
CBP02Doces.Checked := false;
CBP02Outros.Checked := false;
if Paciente.Anamnese.PAnP02 = '' Then
    P02 := 0
Else
    P02 := StrToInt(Paciente.Anamnese.PAnP02);
if (P02 >= 4) Then
    CBP02Outros.Checked := true;
if (P02 mod 2 <> 0) Then
    CBP02Frio.Checked := true;
if (P02 = 2) or (P02 = 3) or (P02 > 5) Then
    CBP02Doces.Checked := true;
EdP02.Text := Paciente.Anamnese.PAnP02q;
//Pergunta 03
if Paciente.Anamnese.PAnP03 = true Then
    RBP03Sim.Checked := true
Else
    RBP03Nao.Checked := true;
EdP03.Text := Paciente.Anamnese.PAnP03q;
//Pergunta 04
if Paciente.Anamnese.PAnP04 = true Then
    RBP04Sim.Checked := true
Else
    RBP04Nao.Checked := true;
EdP04.Text := Paciente.Anamnese.PAnP04q;
//Pergunta 05
if Paciente.Anamnese.PAnP05 = true Then
    RBP05Sim.Checked := true
Else
    RBP05Nao.Checked := true;
EdP05.Text := Paciente.Anamnese.PAnP05q;
//Pergunta 06
if Paciente.Anamnese.PAnP06 = true Then
    RBP06Sim.Checked := true
Else
    RBP06Nao.Checked := true;
//Pergunta 07
if Paciente.Anamnese.PAnP07 = true Then
    RBP07Sim.Checked := true
Else
    RBP07Nao.Checked := true;
//Pergunta 08
if Paciente.Anamnese.PAnP08 = true Then
    RBP08Sim.Checked := true
Else
    RBP08Nao.Checked := true;
//Pergunta 09
EdP09.Text := Paciente.Anamnese.PAnP09q;
//Pergunta 10
if Paciente.Anamnese.PAnP10 = true Then
    RBP10Sim.Checked := true
Else
    RBP10Nao.Checked := true;
EdP10.Text := Paciente.Anamnese.PAnP10q;

//Pergunta 11
P11 := Paciente.Anamnese.PAnP11;
if P11 = '' Then
    P11 := '00000000';
CBP1101.Checked := False;
CBP1102.Checked := False;
CBP1103.Checked := False;
CBP1104.Checked := False;

```

```

CBP1105.Checked := False;
CBP1106.Checked := False;
CBP1107.Checked := False;
CBP1108.Checked := False;
if P11[1] = '1' Then CBP1101.Checked := true;
if P11[2] = '1' Then CBP1102.Checked := true;
if P11[3] = '1' Then CBP1103.Checked := true;
if P11[4] = '1' Then CBP1104.Checked := true;
if P11[5] = '1' Then CBP1105.Checked := true;
if P11[6] = '1' Then CBP1106.Checked := true;
if P11[7] = '1' Then CBP1107.Checked := true;
if P11[8] = '1' Then CBP1108.Checked := true;
EdP11.Text := Paciente.Anamnese.PAnP11q;
SalvarDadosTela := False;
end;

procedure TFormFicha.LerExame;
var
  _exame: string;
begin
  Paciente.Exame.Buscar(Paciente.Pcodigo);
  _exame := Paciente.Exame.PExame;
  //Exame 01
  if _exame[1] = '2' Then
    RBHigieneDeficiente.Checked := true
  Else
    if _exame[1] = '1' Then
      RBHigieneRegular.Checked := true
    Else
      RBHigieneNormal.Checked := true;
  //Exame 02
  if _exame[2] = '2' Then
    RBHalitoseForte.Checked := true
  Else
    if _exame[2] = '1' Then
      RBHalitoseModerada.Checked := true
    Else
      RBHalitoseAusente.Checked := true;
  //Exame 03
  if _exame[3] = '2' Then
    RBTartaroMuito.Checked := true
  Else
    if _exame[3] = '1' Then
      RBTartaroPouco.Checked := true
    Else
      RBTartaroAusente.Checked := true;
  //Exame 04
  if _exame[4] = '2' Then
    RBGengivaPeriodontite.Checked := true
  Else
    if _exame[4] = '1' Then
      RBGengivaGengivite.Checked := true
    Else
      RBGengivaNormal.Checked := true;
  //Exame 05
  if _exame[5] = '1' Then
    RBMucosaAlterada.Checked := true
  Else
    RBMucosaNormal.Checked := true;
  EdLingua.Text := Paciente.Exame.PLingua;
  EdPalato.Text := Paciente.Exame.PPalato;
  EdAssoalho.Text := Paciente.Exame.PAssoalho;
  EdLabios.Text := Paciente.Exame.PLabios;
  MemoExameObs.Text := Paciente.Exame.PExObs;
  SalvarDadosTela := False;
end;

procedure TFormFicha.SalvarExame;
var
  _exame: string;
begin

```

```

_exame := '00000';
//Exame 01
if RBHigieneDeficiente.Checked Then
  _exame[1] := '2'
Else
if RBHigieneRegular.Checked Then
  _exame[1] := '1'
Else
  _exame[1] := '0';
//Exame 02
if RBHalitoseForte.Checked Then
  _exame[2] := '2'
Else
if RBHalitoseModerada.Checked Then
  _exame[2] := '1'
Else
  _exame[2] := '0';
//Exame 03
if RBTartaroMuito.Checked Then
  _exame[3] := '2'
Else
if RBTartaroPouco.Checked Then
  _exame[3] := '1'
Else
  _exame[3] := '0';
//Exame 04
if RGBengivaPeriodontite.Checked Then
  _exame[4] := '2'
Else
if RGBengivaGengivite.Checked Then
  _exame[4] := '1'
Else
  _exame[4] := '0';
//Exame 05
if RBMucosaAlterada.Checked Then
  _exame[5] := '1'
Else
  _exame[5] := '0';
Paciente.Exame.PExame := _exame;
Paciente.Exame.PLingua := EdLingua.Text;
Paciente.Exame.PPalato := EdPalato.Text;
Paciente.Exame.PAssoalho := EdAssoalho.Text;
Paciente.Exame.PLabios := EdLabios.Text;
Paciente.Exame.PExObs := MemoExameObs.Text;
Paciente.Exame.Salvar(Paciente.PCodigo);
SalvarDadosTela := False;
end;

procedure TFormFicha.LerPlano(Codigo: Integer);
begin
  Paciente.PlanoAtual.Carregar(Codigo);
  EdPlanoData.Date := Paciente.PlanoAtual.PData;
  EdPlanoDescricao.Text := Paciente.PlanoAtual.PDescricao;
  MemoPlanoObs.Text := Paciente.PlanoAtual.PObservacao;
  GBPPlanoAtual.Enabled := True;
  PnOdonto := False;
  PnRadio := False;
  SalvarDadosTela := False;
end;

procedure TFormFicha.SalvarPlano(Est: Integer);
begin
  Paciente.PlanoAtual.PCodPac := Paciente.Pcodigo;
  Paciente.PlanoAtual.PData := EdPlanoData.Date;
  Paciente.PlanoAtual.PDescricao := EdPlanoDescricao.Text;
  Paciente.PlanoAtual.PObservacao := MemoPlanoObs.Text;
  Paciente.PlanoAtual.Salvar(Est);
  ColocaPlano;
  SalvarDadosTela := False;
end;

```

```

procedure TFormFicha.ColocaPlano;
var
  a: integer;
begin
  SGPlano.ColCount := 3;
  SGPlano.RowCount := 1;
  SGPlano.ColWidths[0] := 0;
  SGPlano.ColWidths[1] := 90;
  SGPlano.ColWidths[2] := 280;
  SGPlano.Cells[0,0] := '';
  SGPlano.Cells[1,0] := '';
  SGPlano.Cells[2,0] := '';

  Paciente.ColPlano.CCodPaciente := Paciente.Pcodigo;
  Paciente.ColPlano.Seleciona;
  Paciente.ColPlano.Primeiro;
  a := 0;
  While Not Paciente.ColPlano.Final do
    begin
      Paciente.ColPlano.Coloca;
      SGPlano.Cells[0,a] := InttoStr(Paciente.ColPlano.CCodPlano);
      SGPlano.Cells[1,a] := Paciente.ColPlano.CDataPlano;
      SGPlano.Cells[2,a] := Paciente.ColPlano.CDescPlano;
      Paciente.ColPlano.Proximo;
      SGPlano.RowCount := a+1;
      inc(a);
    end;
  end;

procedure TFormFicha.SGPlanoSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if (SGPlano.Cells[0, Arow] <> '') then
    CodigoPlano := StrToInt(SGPlano.Cells[0, Arow])
  else
    CodigoPlano := 0;
end;

procedure TFormFicha.SGPlanoDblClick(Sender: TObject);
begin
  if CodigoPlano <> 0 Then
    LerPlano(CodigoPlano)
  else
    MessageDlg('Não há nenhum Plano de Tratamento selecionado', mtError, [mbok], 0);
end;

procedure TFormFicha.ColocaRadio;
var
  a: integer;
begin
  SGRadio.ColCount := 3;
  SGRadio.RowCount := 1;
  SGRadio.ColWidths[0] := 0;
  SGRadio.ColWidths[1] := 90;
  SGRadio.ColWidths[2] := 280;
  SGRadio.Cells[0,0] := '';
  SGRadio.Cells[1,0] := '';
  SGRadio.Cells[2,0] := '';

  Paciente.PlanoAtual.ColRadio.CCodPlano :=
    Paciente.PlanoAtual.PCódigo;
  Paciente.PlanoAtual.ColRadio.Seleciona;
  Paciente.PlanoAtual.ColRadio.Primeiro;
  a := 0;
  While Not Paciente.PlanoAtual.ColRadio.Final do
    begin
      Paciente.PlanoAtual.ColRadio.Coloca;
      SGRadio.Cells[0,a] :=
        InttoStr(Paciente.PlanoAtual.ColRadio.CCodRadio);
      SGRadio.Cells[1,a] :=
        DateToStr(Paciente.PlanoAtual.ColRadio.CDataRadio);
    end;
  end;

```

```

        SGRadio.Cells[2,a] :=
            Paciente.PlanoAtual.ColRadio.CDescRadio;
        Paciente.PlanoAtual.ColRadio.Proximo;
        SGRadio.RowCount := a+1;
        inc(a);
    end;
end;

procedure TFormFicha.LerRadio(Codigo: Integer);
begin
    Paciente.PlanoAtual.RadioAtual.Carregar(Codigo);
    EdRadioDesc.Text :=
        Paciente.PlanoAtual.RadioAtual.PDescricao;
    EdRadioData.Date := Paciente.PlanoAtual.RadioAtual.PData;
    MemoRadio.Text :=
        Paciente.PlanoAtual.RadioAtual.PObservacao;
    ImgRadio.Picture.LoadFromFile('C:\temp.jpg');
    GBRadioAtual.Enabled := True;
    EdRadioData.SetFocus;
    SalvarDadosTela := False;
end;

procedure TFormFicha.SalvarRadio;
begin
    if odradio.Execute then
    begin
        ImgRadio.Picture.LoadFromFile(odradio.FileName);
        ImgRadio.Picture.SaveToFile('C:\temp.jpg');
        Paciente.PlanoAtual.RadioAtual.PCodPlano :=
            Paciente.PlanoAtual.PCodigo;
        EdRadioData.Date := Now;
        EdRadioDesc.Text := 'NOVA RADIOGRAFIA';
        MemoRadio.Text := '';
        Paciente.PlanoAtual.RadioAtual.PDescricao :=
            EdRadioDesc.Text;
        Paciente.PlanoAtual.RadioAtual.PData :=
            EdRadioData.Date;
        Paciente.PlanoAtual.RadioAtual.Salvar(0);

        GBRadioAtual.Enabled := True;
        EdRadioData.SetFocus;
    end;
end;

procedure TFormFicha.SGRadioSelectCell(Sender: TObject; ACol,
    ARow: Integer; var CanSelect: Boolean);
begin
    if (SGRadio.Cells[0, ARow] <> '') Then
        CodigoRadio := StrToInt(SGRadio.Cells[0, ARow])
    else
        CodigoRadio := 0;
end;

procedure TFormFicha.BBRadioAbrirClick(Sender: TObject);
begin
    if CodigoRadio <> 0 Then
        LerRadio(CodigoRadio)
    else
        MessageDlg('Não há radiografia selecionada!', mtError, [mbok], 0);
end;

procedure TFormFicha.BBRadioAdicionarClick(Sender: TObject);
begin
    SalvarRadio;
    ColocaRadio;
end;

procedure TFormFicha.edfone1KeyPress(Sender: TObject; var Key: Char);
begin
    if not (key in['0'..'9', '(', ')', Chr(8)]) then
        key := #0;

```

```

end;

procedure TFormFicha.edufKeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in['A'..'Z', 'a'..'z', Chr(8)]) then
    key := #0;
end;

procedure TFormFicha.FormShow(Sender: TObject);
begin
  lbnome.Caption := ednome.Text;
end;

procedure TFormFicha.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FormSeleccionar.Cole.Seleciona;
  FormSeleccionar.ColocaDados;
  FreeOnRelease;
end;

procedure TFormFicha.ImgOdontogramaMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  idente, nomdente: string;
  idnum: integer;
  mm: boolean;
begin
  if (CodigoOdonto = 0) then
  begin
    MessageDlg('Não há nenhum odontograma selecionado!', mterror, [mbok], 0);
    exit;
  end;
  if (x < imgodontograma.Width div 2) then
    mm := false
  else
    mm := true;

  if (y < 60) then
  begin
    idente := 'a';
    if mm then
      nomdente := '2'
    else
      nomdente := '1';
  end
  else
  if (y < 90) then
  begin
    idente := 'b';
    if mm then
      nomdente := '2'
    else
      nomdente := '1';
  end
  else
  if (y < 120) then
  begin
    idente := 'c';
    if mm then
      nomdente := '6'
    else
      nomdente := '5';
  end
  else
  if (y < 150) then
  begin
    idente := 'd';
    if mm then
      nomdente := '7'
    else
  end;
end;

```

```

        nomdente := '8';
    end
else
if (y < 180) then
begin
    idente := 'e';
    if mm then
        nomdente := '3'
    else
        nomdente := '4';
end
else
begin
    idente := 'f';
    if mm then
        nomdente := '3'
    else
        nomdente := '4';
end;
if (idente = 'c') or (idente = 'd') then
begin
    idnum := (x div 30) - 2;
    if (idnum < 1) or (idnum > 10) then
        idnum := 0
    else
begin
    if x < (imgodontograma.Width div 2) then
        nomdente := nomdente + IntToStr((imgodontograma.Width div 2) - x) div 30 + 1)
    else
        nomdente := nomdente + IntToStr((x - (imgodontograma.Width div 2)) div 30 + 1);
end;
end
else
begin
    idnum := (x div 30) + 1;
    if x < (imgodontograma.Width div 2) then
        nomdente := nomdente + IntToStr(((imgodontograma.Width div 2) - x) div 30 + 1)
    else
        nomdente := nomdente + IntToStr((x - (imgodontograma.Width div 2)) div 30 + 1);
end;
if idnum <> 0 then
begin
    if idnum < 10 then
        idente := idente + '0' + IntToStr(idnum)
    else
        idente := idente + IntToStr(idnum);

    if not assigned(FormDente) then
        FormDente := TFormDente.Create(Self);
    FormDente.CodOdonto :=
        Paciente.PlanosAtuais.OdontoAtual.PCodigo;
    FormDente.Caption := 'Dente ' + nomdente;
    FormDente.neg_nom := idente;
    FormDente.ShowModal;
    FormDente := nil;
end;
end;

procedure TFormFicha.ColocaOdontograma;
var
  a: integer;
begin
  SGOdontograma.ColCount := 3;
  SGOdontograma.RowCount := 1;
  SGOdontograma.ColWidths[0] := 0;
  SGOdontograma.ColWidths[1] := 90;
  SGOdontograma.ColWidths[2] := 280;
  SGOdontograma.Cells[0,0] := '';
  SGOdontograma.Cells[1,0] := '';
  SGOdontograma.Cells[2,0] := '';

```

```

Paciente.PlanoAtual.ColOdonto.CCodPlano :=
    Paciente.PlanoAtual.PCodigo;
Paciente.PlanoAtual.ColOdonto.Seleciona;
Paciente.PlanoAtual.ColOdonto.Primeiro;
a := 0;
While Not Paciente.PlanoAtual.ColOdonto.Final do
begin
    Paciente.PlanoAtual.ColOdonto.Coloca;
    SGODontograma.Cells[0,a] :=
        InttoStr(Paciente.PlanoAtual.ColOdonto.CCodOdonto);
    SGODontograma.Cells[1,a] :=
        Paciente.PlanoAtual.ColOdonto.CDataOdonto;
    SGODontograma.Cells[2,a] :=
        Paciente.PlanoAtual.ColOdonto.CDescOdonto;
    Paciente.PlanoAtual.ColOdonto.Proximo;
    SGODontograma.RowCount := a+1;
    inc(a);
end;
end;

procedure TFormFicha.LerOdontograma(Codigo: Integer);
begin
    Paciente.PlanoAtual.OdontoAtual.Carregar(CodigoOdonto);
    Paciente.PlanoAtual.OdontoAtual.PCodigo := Codigo;
    EdOdontogramaData.Date :=
        Paciente.PlanoAtual.OdontoAtual.PData;
    EdOdontogramaDescricao.Text :=
        Paciente.PlanoAtual.OdontoAtual.PDescricao;
    MemoOdontogramaObs.Text :=
        Paciente.PlanoAtual.OdontoAtual.PObservacao;
    LerOdontoOperacoes;
    GBODontoAtual.Enabled := True;
    EdOdontogramaData.SetFocus;
    SalvarDadosTela := False;
end;

procedure TFormFicha.SalvarOdontograma(Est: Integer);
begin
    Paciente.PlanoAtual.OdontoAtual.PCodPlano :=
        Paciente.PlanoAtual.PCodigo;
    Paciente.PlanoAtual.OdontoAtual.PData :=
        EdOdontogramaData.Date;
    Paciente.PlanoAtual.OdontoAtual.PDescricao :=
        EdOdontogramaDescricao.Text;
    Paciente.PlanoAtual.OdontoAtual.PObservacao :=
        MemoOdontogramaObs.Text;
    Paciente.PlanoAtual.OdontoAtual.Salvar(Est);
    ColocaOdontograma;
    SalvarDadosTela := False;
end;

procedure TFormFicha.BBSalvarDadosClick(Sender: TObject);
begin
    if (EdNome.Text = '') Then
        MessageDlg('Nome do paciente vazio!', mtError, [mbOk], 0)
    else
        begin
            SalvarDados;
            GBMenu.Enabled := True;
        end;
end;

procedure TFormFicha.ExcluirPaciente;
var
    Exclui: Boolean;
begin
    Exclui := True;
    if MessageDlg('Deseja excluir o paciente?', mtConfirmation, [mbYes, mbNo], 0) = mbYes then
begin
    try
        Paciente.Salvar(2);

```

```

    Except
      Exclui := false;
    End;

    If Exclui Then
      begin
        MessageDlg('Dados Excluídos!', mtInformation, [mbOk], 0);
        FormFicha.Free;
      end
    else
      MessageDlg('Não foi possível a exclusão do paciente! Verifique a conexão com o
banco de dados.', mtError, [mbOk], 0);
    end;
  end;

procedure TFormFicha.BBSalvarExameClick(Sender: TObject);
begin
  SalvarExame;
end;

procedure TFormFicha.BBSalvarAnamneseClick(Sender: TObject);
begin
  SalvarAnamnese;
end;

procedure TFormFicha.BBSalvarPlanoClick(Sender: TObject);
begin
  SalvarPlano(1);
  BBInserirPlano.Enabled := True;
end;

procedure TFormFicha.BBInserirPlanoClick(Sender: TObject);
begin
  EdPlanoData.Date := Now;
  EdPlanoDescricao.Text := 'PLANO NOVO';
  MemoPlanoObs.Text := '';
  SalvarPlano(0);
  BBInserirPlano.Enabled := False;
  GBPPlanoAtual.Enabled := True;
  EdPlanoDescricao.SetFocus;
end;

procedure TFormFicha.BBAbrirPlanoClick(Sender: TObject);
begin
  if CodigoPlano <> 0 Then
    LerPlano(CodigoPlano)
  else
    MessageDlg('Não há nenhum Plano de Tratamento selecionado', mtError, [mbok], 0);
    BBInserirPlano.Enabled := True;
end;

procedure TFormFicha.SGOdontogramaSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if (SGOdontograma.Cells[0, ARow] <> '') then
    begin
      CodigoOdonto := StrToInt(SGOdontograma.Cells[0, ARow]);
    end
  else
    CodigoOdonto := 0;
end;

procedure TFormFicha.SGOdontogramaDblClick(Sender: TObject);
begin
  if CodigoOdonto <> 0 Then
    LerOdontograma(CodigoOdonto)
  else
    MessageDlg('Não há nenhum odontograma selecionado', mtError, [mbok], 0);
end;

procedure TFormFicha.LerOdontoOperacoes;

```

```

var
  ndente: string;
  dtop, dleft, dright, dbottom, px, py: integer;
  _cor: TColor;
  neg: TBitMap;
begin
  ImgOdontograma.Picture.
    LoadFromFile(ExtractFilePath(Application.ExeName)+'dente\odontobmp');

  neg := TBitMap.Create;
  neg.Width := imgodontograma.Width;
  neg.Height := imgodontograma.Height;
  neg.LoadFromFile(ExtractFilePath(Application.ExeName)+'dente\odonton.bmp');
  with Paciente.PlanoAtual.OdontoAtual.Operacoes do
    begin
      CCodOdonto := Paciente.PlanoAtual.OdontoAtual.PCodigo;
      Seleciona;
      Primeiro;
      end;
  while not
    Paciente.PlanoAtual.OdontoAtual.Operacoes.Final do
  begin
    Paciente.PlanoAtual.OdontoAtual.Operacoes.Coloca;
    ndente := Paciente.PlanoAtual.OdontoAtual.Operacoes.CNomeDente;
    case
      Paciente.PlanoAtual.OdontoAtual.Operacoes.CCor of
      0: _cor := clblue;
      1: _cor := clred;
      end;

      if (ndente[1] = 'a') then dtop := 0;
      if (ndente[1] = 'f') then dtop := 180;
      if (ndente[1] = 'b') then dtop := 60;
      if (ndente[1] = 'c') then dtop := 90;
      if (ndente[1] = 'd') then dtop := 120;
      if (ndente[1] = 'e') then dtop := 150;

      if (ndente[1] = 'a') or (ndente[1] = 'f') then
        dbottom := dtop + 60
      else dbottom := dtop + 30;

      if (ndente[1] = 'c') or (ndente[1] = 'd') then
        dleft := 60 + StrToInt(ndente[2]+ndente[3]) * 30
      else
        dleft := (StrToInt(ndente[2]+ndente[3]) - 1) * 30;
      dright := dleft + 30;

      Px :=
      Paciente.PlanoAtual.OdontoAtual.Operacoes.CPosX div 3;
      Py :=
      Paciente.PlanoAtual.OdontoAtual.Operacoes.CPosY div 3;
      case
        Paciente.PlanoAtual.OdontoAtual.Operacoes.COperacao
        of
        1: begin
          if (dbottom - dtop < 50) then
            FormDente.Obturacaomenor(imgodontograma, dtop, dleft, dbottom,
            dright, dleft+px, dtop+py, neg, _cor)
          else
            FormDente.Obturacao(imgodontograma, 2, dleft+px, dtop+py, neg, _cor);
          end;
        2: FormDente.Endodontia(imgodontograma, dtop, dleft, dbottom,
          dright, neg, _cor);
        3: FormDente.Exodontia(imgodontograma, dtop, dleft, dbottom,
          dright, _cor);
        end;
      Paciente.PlanoAtual.OdontoAtual.Operacoes.Proximo;
    end;
  SalvarDadosTela := False;
end;

```

```

procedure TFormFicha.BBInserirOdontoClick(Sender: TObject);
begin
  imgodontograma.Picture.
    LoadFromFile(ExtractFilePath(Application.ExeName) + 'dente\odonto.bmp');
  EdOdontogramaData.Date := Now;
  EdOdontogramaDescricao.Text := 'NOVO ODONTOGRAMA';
  SalvarOdontograma(0);
 CodigoOdonto := Paciente.PlanoAtual.OdontoAtual.PCodigo;
  BBInserirOdonto.Enabled := False;
  GBOdontoAtual.Enabled := True;
  EdOdontogramaDescricao.SetFocus;
end;

procedure TFormFicha.BBExcluirOdontoClick(Sender: TObject);
begin
  SalvarOdontograma(2);
  imgodontograma.Picture.
    LoadFromFile(ExtractFilePath(Application.ExeName) + 'dente\odonto.bmp');
  EdOdontogramaData.Date := '';
  EdOdontogramaDescricao.Text := '';
  MemoOdontogramaObs.Text := '';
  GBOdontoAtual.Enabled := False;
  BBInserirOdonto.Enabled := True;
end;

procedure TFormFicha.BBSalvarOdontoClick(Sender: TObject);
begin
  SalvarOdontograma(1);
  BBInserirOdonto.Enabled := True;
end;

procedure TFormFicha.SGPlanoDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormFicha.BBSelecionarOdontoClick(Sender: TObject);
begin
  if CodigoOdonto <> 0 Then
    LerOdontograma(CodigoOdonto)
  else
    MessageDlg('Não há nenhum odontograma selecionado', mtError, [mbok], 0);
    BBInserirOdonto.Enabled := True;
end;

procedure TFormFicha.SGRadioDblClick(Sender: TObject);
begin
  if CodigoRadio <> 0 Then
    LerRadio(CodigoRadio)
  else
    MessageDlg('Não há radiografia selecionada!', mtError, [mbok], 0);
end;

procedure TFormFicha.GravarDadosRadio;
begin
  with Paciente.PlanoAtual.RadioAtual do
  begin
    PDescricao := EdRadioDesc.Text;
    PData := EdRadioData.Date;
    PObservacao := MemoRadio.Text;
    Salvar(1);
  end;
  SalvarDadosTela := False;
end;

```

```

procedure TFormFicha.BBRadioSalvarClick(Sender: TObject);
begin
  GravarDadosRadio;
  ColocaRadio;
end;

procedure TFormFicha.BBPlanoExcluirClick(Sender: TObject);
begin
  if MessageDlg('Deseja excluir este Plano de Tratamento? (Isso excluirá ' +
    'todos os odontogramas e radiografias nele cadastrados)', mtConfirmation,
    [mbyes, mbno], 0) = mrYes Then
  begin
    SalvarPlano(2);
    GBPlanoAtual.Enabled := False;
    Paciente.PlanoAtual.PCodigo := 0;
    BBInserirPlano.Enabled := True;
    SalvarDadosTela := False;
  end;
  CodigoPlano := 0;
end;

procedure TFormFicha.BBRadioExcluirClick(Sender: TObject);
begin
  if (MessageDlg('Deseja excluir esta radiografia?', mtConfirmation,
    [mbyes, mbno], 0) = mrYes) Then
  begin
    Paciente.PlanoAtual.RadioAtual.Salvar(2);
    ColocaRadio;
    CodigoRadio := 0;

    EdRadioData.Date := Now;
    ImgRadio.Picture.
    LoadFromFile(ExtractFilePath(Application.ExeName)+'imagens\padrao.jpg');
    EdRadioDesc.Text := '';
    MemoRadio.Text := '';
    GBRadioAtual.Enabled := False;
    SalvarDadosTela := False;
  end;
end;

procedure TFormFicha.BBExcluirClick(Sender: TObject);
begin
  if MessageDlg('Tem certeza que deseja excluir o paciente? (Todos os dados '+
    'e tratamentos serão perdidos)', mtConfirmation,
    [mbYes, mbNo], 0) = mrYes Then
  begin
    Paciente.Salvar(2);
    FormFicha.Close;
  end;
end;

procedure TFormFicha.LerConsultas;
var
  a: integer;
begin
  SGConsultas.ColCount := 6;
  SGConsultas.RowCount := 1;
  SGConsultas.ColWidths[0] := 0;
  SGConsultas.ColWidths[1] := 80;
  SGConsultas.ColWidths[2] := 60;
  SGConsultas.ColWidths[3] := 60;
  SGConsultas.ColWidths[4] := 260;
  SGConsultas.ColWidths[5] := 100;

  Consultas.PesquisaPaciente(Paciente.Pcodigo, Now);
  Consultas.Primeiro;
  a := 0;
  while not (consultas.Final) do
  begin
    Consultas.ColoquePesquisa;
    SGConsultas.RowCount := a + 1;
  end;
end;

```

```

SGConsultas.Cells[0, a] := IntToStr(Consultas.CCodCompromisso);
SGConsultas.Cells[1, a] := DateToStr(Consultas.CDataCompromisso);
SGConsultas.Cells[2, a] := Copy(TimeToStr(Consultas.CHoraCompromisso), 1, 5);
SGConsultas.Cells[3, a] := Copy(TimeToStr(Consultas.CDuracaoCompromisso), 1, 5);
SGConsultas.Cells[4, a] := Consultas.CDescCompromisso;
if Consultas.CStatus = '1' Then
    SGConsultas.Cells[5, a] := 'CONFIRMADO'
else
    SGConsultas.Cells[5, a] := 'À CONFIRMAR';
Consultas.Proximo;
Inc(a);
end;
end;

procedure TFormFicha.SGConsultasSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if SGConsultas.Cells[0, Arow] <> '' Then
    CodigoConsulta := StrToInt(SGConsultas.Cells[0, Arow])
  else
    CodigoConsulta := 0;
end;

procedure TFormFicha.BBExcluirConsultaClick(Sender: TObject);
begin
  if CodigoConsulta = 0 Then
    MessageDlg('Não há consulta selecionada!', mtError, [mbok], 0)
  else
    begin
      if MessageDlg('Tem certeza que deseja excluir essa consulta?',
                     mtConfirmation, [mbyes, mbno], 0) = mrYes Then
        begin
          Compromisso := TCompromisso.Create(Self);
          Compromisso.PCodigo := CodigoConsulta;
          Compromisso.Salvar(2);
          Compromisso.Destroy;
          CodigoConsulta := 0;
          LerConsultas;
        end;
    end;
end;

procedure TFormFicha.BBAdicionarConsultaClick(Sender: TObject);
begin
  if not assigned(FormAgenda) then
    FormAgenda := TFormAgenda.Create(Self);
  FormAgenda.EdDataComp.Date := Now;
  FormAgenda.ArrumaAgenda;
  FormAgenda.ShowModal;
  FormAgenda := nil;
end;

procedure TFormFicha.BBAlterarConsultaClick(Sender: TObject);
begin
  if CodigoConsulta <> 0 Then
    begin
      if not assigned(FormAgenda) then
        FormAgenda := TFormAgenda.Create(Self);
      FormAgenda.EdDataComp.Date := StrToDate(SGConsultas.Cells[1, SGConsultas.Row]);
      FormAgenda.ArrumaAgenda;
      FormAgenda.ShowModal;
      FormAgenda := nil;
    end;
end;

procedure TFormFicha.BBFecharClick(Sender: TObject);
begin
  if SalvarDadosTela Then
    SalvarTelaAtual(Selecionado);
  Close;
end;

```

```
procedure TFormFicha.ednomeChange(Sender: TObject);
begin
  SalvarDadosTela := True;
end;

procedure TFormFicha.SalvarTelaAtual(Tela: Integer);
begin
  case Tela of
    1: if MessageDlg('Deseja salvar os dados do paciente?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes Then
        SalvarDados
      else
        CarregarDados(Paciente.PCodigo);
    2: if MessageDlg('Deseja salvar os dados da anamnese?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes Then
        SalvarAnamnese
      else
        LerAnamnese;
    3: if MessageDlg('Deseja salvar os dados do exame intra-oral?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes Then
        SalvarExame
      else
        LerExame;
    4, 5: if MessageDlg('Deseja salvar os dados do plano de tratamento?',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
        SalvarPlano(1)
      else
        LerPlano(Paciente.PlanoAtual.PCodigo);
    6: if MessageDlg('Deseja salvar os dados do odontograma?',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
        SalvarOdontograma(1)
      else
        LerOdontograma(Paciente.PlanoAtual.OdontoAtual.PCodigo);
    7: if MessageDlg('Deseja salvar os dados da radiografia?',
      mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
        GravarDadosRadio
      else
        LerRadio(Paciente.PlanoAtual.RadioAtual.PCodigo);
  end;
end;
end.
```

5 UNIT UDENTE



FIGURA 05 – FORMDENTE

```

/////////////////////////////////////////////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
///////////////////////////////////////////////////////////////////
unit uDente;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Buttons, uOperacao, uColOperacao, uFicha;

type
  TFormDente = class(TForm)
    imgdente: TImage;
    LBOperacoes: TListBox;
    GBOperacoes: TGroupBox;
    pndente: TPanel;
    rbrealizado: TRadioButton;
    rbrealizar: TRadioButton;
    SBEndodontia: TSpeedButton;
    SBExodontia: TSpeedButton;
    SBObturacao: TSpeedButton;
    BtDeletar: TBitBtn;
    EdDescricao: TEdit;
    LabDescricao: TLabel;
    BBDenteSalvar: TBitBtn;
    BBDenteSair: TBitBtn;
    PnDente: TPanel;
    PnOperacoes: TPanel;
    GBOpcoes: TGroupBox;
    GBOpRealizadas: TGroupBox;

procedure imgdenteMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure SBEndodontiaClick(Sender: TObject);
procedure rbrealizadoClick(Sender: TObject);
procedure rbrealizarClick(Sender: TObject);
procedure SBExodontiaClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);

```

```

procedure FormShow(Sender: TObject);
procedure SObturacaoClick(Sender: TObject);
procedure BtDeletarClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure BBDenteSalvarClick(Sender: TObject);
procedure BBDenteSairClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  neg_nom: string;
  CodOdonto: Integer;
  ColOperacao: TColOperacao;

    function Obturacao(imagem: TImage; tm, cima, esquerda: integer; negativo: TBitMap; cor: TColor): boolean;
    function Obturacaomenor(imagem: TImage; cima, esquerda, altura, largura, posx, posy: integer; negativo: TBitMap; cor: TColor): boolean;
    procedure Endodontia(imagem: TImage; cima, esquerda, altura, largura: integer; negativo: TBitMap; cor: TColor);
    procedure Exodontia(imagem: TImage; cima, esquerda, altura, largura: integer; cor: TColor);

  procedure bloqueiabotao;

  function pintaritem(opera, posx, posy, cont: integer; _cor: TColor): boolean;
  procedure ColocaLista(codopera, num, op, posx, posy, cor, bd: integer; descricao: string);
  procedure DeletaLista;
  procedure AtualizaDente;

  procedure SalvarOperacao;
  procedure LerOperacao;

end;

var
  FormDente: TFormDente;
  fneg: TBitMap;
  contador, ValOpera: integer;
  cor: TColor;
  Operacao: array of TOperacao;

implementation

{$R *.dfm}

function TFormDente.Obturacao(imagem: TImage; tm, cima, esquerda: integer; negativo: TBitMap; cor: TColor): boolean;
begin
  imagem.Canvas.Pen.Color := cor;
  imagem.Canvas.Brush.Color := cor;
  if negativo.Canvas.Pixels[cima, esquerda] = clwhite then
    begin
      imagem.Canvas.Ellipse(cima-tm, esquerda-tm, cima+tm, esquerda+tm);
      Result := true;
    end
  else
    begin
      Result := false;
    end;
end;

procedure TFormDente.imgdenteMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  confere: boolean;
begin
  confere := false;
  if pintaritem(ValOpera, x, y, contador, cor) then
    begin
      Inc(Contador);
    end;
end;

```

```

        confere := true;
    end;
if confere then
begin
    if cor = clblue then
        ColocaLista(0, Contador-1, ValOpera, x, y, 0, 1, EdDescricao.Text)
    else if cor = clred then
        ColocaLista(0, Contador-1, ValOpera, x, y, 1, 1, EdDescricao.Text);
    end;
if ValOpera = 2 then
begin
    SBEndodontia.Enabled := False;
    ValOpera := 0;
end
else
if ValOpera = 3 then
begin
    SBExodontia.Enabled := False;
    ValOpera := 0;
end;
end;

procedure TFormDente.Endodontia(imagem: TImage; cima, esquerda, altura, largura: integer;
negativo: TBitMap; cor: TColor);
var
    a,b: integer;
begin
    for a:= cima to altura do
        for b:= esquerda to largura do
        begin
            if negativo.Canvas.Pixels[b,a] = clred then
                imagem.Canvas.Pixels[b,a] := cor;
        end;
end;

procedure TFormDente.SBEndodontiaClick(Sender: TObject);
begin
bloqueiabotao;
if ValOpera <> 2 then
begin
    ValOpera := 2;
    SBEndodontia.Flat := true;
    EdDescricao.Text := 'Endodontia';
end
else
begin
    ValOpera := 0;
end;
end;

procedure TFormDente.rbrealizadoClick(Sender: TObject);
begin
    cor := clblue;
end;

procedure TFormDente.rbrealarClick(Sender: TObject);
begin
    cor := clred;
end;

procedure TFormDente.Exodontia(imagem: TImage; cima, esquerda, altura,
largura: integer; cor: TColor);
begin
    imagem.Canvas.Pen.Color := Cor;
    imagem.Canvas.Pen.Width := 3;
    imagem.Canvas.MoveTo(esquerda, cima);
    imagem.Canvas.LineTo(largura, altura);
    imagem.Canvas.MoveTo(largura, cima);
    imagem.Canvas.LineTo(esquerda, altura);
end;

```

```

procedure TFormDente.SBExodontiaClick(Sender: TObject);
begin
  bloqueiabotao;
  if ValOpera <> 3 then
    begin
      ValOpera := 3;
      SBExodontia.Flat := True;
      EdDescricao.Text := 'Exodontia';
    end
  else
    begin
      ValOpera := 0;
    end;
  SBExodontia.Enabled := true;
end;

function TFormDente.Obturacaomenor(imagem: TImage; cima, esquerda, altura,
  largura, posx, posy: integer; negativo: TBitMap; cor: TColor): boolean;
var
  a,b: integer;
  conf: TColor;
begin
  conf := negativo.Canvas.Pixels[posx, posy];
  if (conf=clred) or (conf=clblue) or (conf=clyellow) or (conf=clgreen) or
    (conf=clolive) or (conf=clfuchsia) or (conf=clqua) then
    begin
      for a:= cima to altura do
        for b:= esquerda to largura do
          begin
            if negativo.Canvas.Pixels[b,a] = conf then
              imagem.Canvas.Pixels[b,a] := cor;
          end;
      Result := true;
    end
  else
    begin
      Result := false;
    end;
end;

procedure TFormDente.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FormFicha.LerOdontoOperacoes;
  FreeOnRelease;
end;

procedure TFormDente.FormShow(Sender: TObject);
begin
  fneg := TBitMap.Create;
  FormDente.imgdente.Picture.LoadFromFile('dente\' + neg_nom + '.bmp');
  fneg.LoadFromFile('dente\' + neg_nom + 'n.bmp');
  fneg.Width := 90;
  if (neg_nom[1] = 'a') or (neg_nom[1] = 'f') then
    fneg.Height := 180
  else
    fneg.Height := 90;
  cor := clblue;
  ImgDente.Canvas.Font.Name := 'Arial';
  ImgDente.Canvas.Font.Size := 7;

  if (neg_nom[1] <> 'a') and (neg_nom[1] <> 'f') Then
    SBEndodontia.Visible := False
  else
    SBEndodontia.Visible := True;

  SetLength(Operacao, 1);
  Operacao[0] := TOperacao.Create(Self);
  ValOpera := 0;
  LerOperacao;
end;

```

```

procedure TFormDente.SBObturacaoClick(Sender: TObject);
begin
  bloqueiabotao;
  if ValOpera <> 1 then
    begin
      ValOpera := 1;
      SBObturacao.Flat := true;
      EdDescricao.Enabled := true;
      EdDescricao.Text := 'Obturação';
    end
  else
    begin
      ValOpera := 0;
      EdDescricao.Enabled := false;
    end;
end;

procedure TFormDente.bloqueiabotao;
var a: integer;
begin
  for a:= 0 to GBOperacoes.Owner.ComponentCount - 1 do
    begin
      if GBOperacoes.Owner.Components[a] is TSpeedButton then
        if TSpeedButton(GBOperacoes.Owner.Components[a]).Parent.Name = GBOperacoes.Name
      then
        TSpeedButton(GBOperacoes.Owner.Components[a]).Flat := false;
      end;
      EdDescricao.Enabled := False;
    end;
end;

procedure TFormDente.ColaLista(CodOpera, num, op, posx, posy, cor, bd: integer; descricao: string);
var
  item: string;
begin
  SetLength(Operacao, num+1);
  if not ASSIGNED(Operacao[num]) Then
    Operacao[num] := TOperacao.Create(Self);

  Operacao[num].PCodigo := CodOpera;
  Operacao[num].POperacao := Op;
  Operacao[num].PPosX := PosX;
  Operacao[num].PPosY := PosY;
  Operacao[num].PDescricao := Descricao;
  Operacao[num].PCor := Cor;
  Operacao[num].Situacao := bd;

  if num < 10 then
    item := '0'+IntToStr(num)+': '
  else
    item := IntToStr(num)+': ';

  if op = 1 then
    item := item + Descricao
  else if op = 2 then
    item := item + 'Endodontia'
  else if op = 3 then
    item := item + 'Exodontia';

  LBOperacoes.Items.Add(item);
end;

procedure TFormDente.DeletaLista;
var
  numitem: integer;
  valor: String;
begin
  bloqueiabotao;

```

```

valor := LBOperacoes.Items.Strings[LBOperacoes.ItemIndex];
numitem := StrToInt(valor[1] + valor[2]);

if (Operacao[numitem].POperacao = 2) then
    SBEndodontia.Enabled := True;

if (Operacao[numitem].POperacao = 3) then
    SBExodontia.Enabled := True;

if (Operacao[numitem].Situacao = 0) Then
    Operacao[numitem].Situacao := 3
else
if (Operacao[numitem].Situacao = 1) Then
    Operacao[numitem].Situacao := 2;
LBOperacoes.Items.Delete(LBOperacoes.ItemIndex);

end;

procedure TFormDente.BtDeletarClick(Sender: TObject);
begin
    if (LBOperacoes.ItemIndex >= 0) Then
    begin
        DeletaLista;
        AtualizaDente;
    end;
end;

procedure TFormDente.AtualizaDente;
var
    a, b: integer;
    confere: boolean;
    corx: TColor;
    posx, posy: string;
begin

    imgdente.Picture.LoadFromFile('dente\' + neg_nom + '.bmp');
    for a:= 1 to High(Operacao) do
        begin
            if Operacao[a].PCor = 0 then
                corx := clblue
            else
            if Operacao[a].PCor = 1 then
                corx := clred;
            if Operacao[a].Situacao < 2 then
                pintaritem(Operacao[a].POperacao, Operacao[a].PPosX, Operacao[a].PPosY, a, corx);

        end;
end;

function TFormDente.pintaritem(opera, posx, posy,
    cont: integer; _cor: TColor): boolean;
var
    confirma: boolean;
    valor: string;
begin

    if cont < 10 then
        valor := '0' + IntToStr(Cont)
    else
        valor := IntToStr(Cont);

    imgdente.Canvas.Font.Name := 'arial';
    imgdente.Canvas.Font.Size := 7;
    imgdente.Canvas.Font.Color := _cor;
    imgdente.Canvas.Brush.Color := clwhite;
    case opera of
    1: begin
            if imgdente.Height > 90 then
            begin
                if obturacao(imgdente, 7, posx, posy, fneg, _cor) then

```

```

begin
  imgdente.Canvas.Brush.Color := _cor;
  imgdente.Canvas.Font.Color := clwhite;
  imgdente.Canvas.TextOut(posx-5, posy-6, valor);
  confirma := true;
end;
end
else
begin
  if obturacaomenor(imgdente, 0, 0, imgdente.Height, imgdente.Width, posx, posy, fneg,
  _cor) then
    begin
      imgdente.Canvas.Brush.Color := _cor;
      imgdente.Canvas.Font.Color := clwhite;
      imgdente.Canvas.TextOut(posx, posy, valor);
      confirma := true;
    end;
  end;
end;
2: begin
  endodontia(imgdente, 0, 0, imgdente.Height, imgdente.Width, fneg, _cor);
  imgdente.Canvas.TextOut(15, 0, valor + ': Endodontia');
  confirma := true;
end;
3: begin
  Exodontia(imgdente, 0, 0, imgdente.Height, imgdente.Width, _cor);
  imgdente.Canvas.TextOut(15, imgdente.Height-10, valor + ': Exodontia');
  confirma := true;
end;
end;

if confirma then
  Result := true
else
  Result := false;
end;

procedure TFormDente.SalvarOperacao;
var
  a: integer;
  salva2: boolean;
  letra: char;
begin
for a := 1 to High(Operacao) do
begin
  begin
    Operacao[a].PDenteNome := neg_nom;
    Operacao[a].PCodOdonto := CodOdonto;
    if Operacao[a].Situacao = 1 Then
      Operacao[a].Salvar(0);
    if Operacao[a].Situacao = 3 Then
      Operacao[a].Salvar(2);
  end;
end;
end;

procedure TFormDente.FormCreate(Sender: TObject);
begin
  ColOperacao := TColOperacao.Create(Self);
end;

procedure TFormDente.LerOperacao;
var
  a, b: integer;
  numero: boolean;
begin
  ColOperacao.CCodOdonto := CodOdonto;
  ColOperacao.CNomeDente := neg_nom;
  ColOperacao.SelecionaDente;
  ColOperacao.Primeiro;
  a := 1;
  while not ColOperacao.Final do
    begin

```

```
ColOperacao.Coloca;
if ColOperacao.COperacao = 2 then
  SBEndodontia.Enabled := False;
if ColOperacao.COperacao = 3 then
  SBExodontia.Enabled := False;
ColocaLista(ColOperacao.CCodigo, a, ColOperacao.COperacao,
            ColOperacao.CPosX, ColOperacao.CPosY, ColOperacao.CCor, 0,
            ColOperacao.CDesc);
Inc(a);
ColOperacao.Proximo;
end;

AtualizaDente;
Contador := a;
end;

procedure TFormDente.BBDenteSalvarClick(Sender: TObject);
begin
  SalvarOperacao;
  Close;
end;

procedure TFormDente.BBDenteSairClick(Sender: TObject);
begin
  if MessageDlg('Deseja salvar as alterações?', mtConfirmation,
                [mbyes, mbno], 0) = mrYes Then
    begin
      SalvarOperacao;
    end;
  Close;
end;
end.
```

6 UNIT UAGENDA

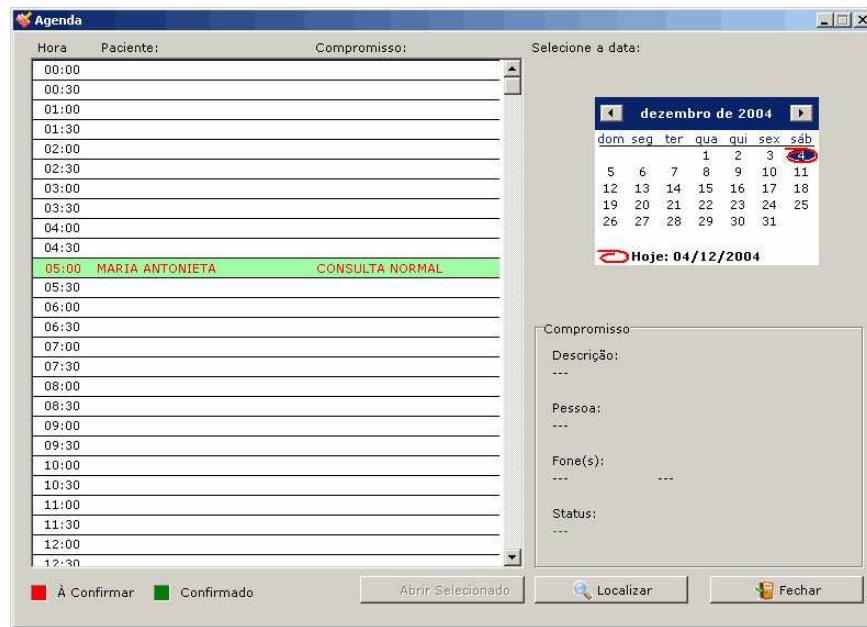


FIGURA 06 - FORMAGENDA

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uAgenda;  
  
interface  
  
uses  
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
  Dialogs, Grids, StdCtrls, ComCtrls, ExtCtrls, Buttons, uColCompromisso,  
  DateUtils;  
  
type  
  TFormAgenda = class(TForm)  
    LabSelData: TLabel;  
    SGDia: TStringGrid;  
    PanAgenda: TPanel;  
    LabHora: TLabel;  
    LabPaciente: TLabel;  
    PanAConfirmar: TPanel;  
    PanConfirmado: TPanel;  
    LabAConfirmar: TLabel;  
    LabConfirmado: TLabel;  
    BBBuscar: TBitBtn;  
    BBSair: TBitBtn;  
    EdDataComp: TMonthCalendar;  
    GBCompromisso: TGroupBox;  
    LabDesc: TLabel;  
    LabDescValor: TLabel;
```

```

LabNomePac: TLabel;
LabClienteValor: TLabel;
LabFones: TLabel;
LabFone1Valor: TLabel;
LabFone2Valor: TLabel;
LabConfirma: TLabel;
LabStatus: TLabel;
BBAAlterar: TBitBtn;
LabCompromisso: TLabel;

procedure Button3Click(Sender: TObject);

procedure ArrumaAgenda;
procedure FormShow(Sender: TObject);
procedure BBNovoClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure SGDiaDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure EdDataCompDblClick(Sender: TObject);
procedure BBAAlterarClick(Sender: TObject);
procedure SGDiaSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure SGDiaDblClick(Sender: TObject);
procedure BBBuscarClick(Sender: TObject);
procedure BBSairClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);

private
  { Private declarations }
public
  { Public declarations }
  procedure ColocaDados;
  procedure Atualizar;
end;

var
  FormAgenda: TFormAgenda;
  Compromissos: TColCompromisso;
  SomaCor, sel: Integer;

const
  cor: array[0..4] of String = ('$00A4E8FB', '$00A4A4FB', '$00A8FBA4',
    '$00FBD3A4', '$00A4BDFB');

//  cor: array[0..4] of String = ('$00A4A4FB', '$00FBD3A4', '$00A8FBA4',
//    '$00A4E8FB', '$00A4BDFB');
implementation

uses uCadAgenda, uLocAgenda;

{$R *.dfm}

procedure TFormAgenda.Button3Click(Sender: TObject);
begin
  if not assigned(FormLocAgenda) then
    FormLocAgenda := TFormLocAgenda.Create(Self);

  FormLocAgenda.ShowModal;
  FormLocAgenda := nil;
end;

procedure TFormAgenda.ArrumaAgenda;
var
  a, b, c: integer;
  hora: string;
begin

  SGDia.ColCount := 8;
  SGDia.ColWidths[0] := 0;
  SGDia.ColWidths[1] := 55;

```

```

SGDia.ColWidths[2] := 200;
SGDia.ColWidths[3] := 170;
SGDia.ColWidths[4] := 0;
SGDia.ColWidths[5] := 0;
SGDia.ColWidths[6] := 0;
SGDia.ColWidths[7] := 0;

SGDia.RowCount := 48;

SGDia.Cells[0,0] := '0';
SGDia.Cells[0,6] := '';

for a:= 0 to 23 do
begin
  b := a * 2;

  if a < 10 then
    hora := ' 0' + IntToStr(a)
  else
    hora := ' ' + IntToStr(a);

  for c:= 1 to 7 do
  begin
    SGDia.Cells[c, b] := '';
    SGDia.Cells[c, b+1] := '';
  end;

  SGDia.Cells[0,b] := '0';
  SGDia.Cells[0,b+1] := '0';

  SGDia.Cells[1, b] := hora + ':00';
  SGDia.Cells[1, b+1] := hora + ':30';
end;

Compromissos.CDataCompromisso := EdDataComp.Date;

ColocaDados;

end;

procedure TFormAgenda.FormShow(Sender: TObject);
begin
  ArrumaAgenda;
end;

procedure TFormAgenda.BBNovoClick(Sender: TObject);
begin
  if not assigned(FormCadAgenda) then
    FormCadAgenda := TFormCadAgenda.Create(Self);

  FormCadAgenda.CodigoCompromisso := 0;
  FormCadAgenda.ShowModal;
  FormCadAgenda := nil;
end;

procedure TFormAgenda.ColocaDados;
var a :integer;
  hora: string;
  duracao: TTime;
begin

  Compromissos.Seleciona;

  a := 0;
  Compromissos.Primeiro;

  While Not Compromissos.Final do
  begin

```

```

Compromisso.Coloca;
Hora := TimeToStr(Compromisso.CHoraCompromisso);
a := StrToInt(hora[1] + hora[2]) * 2;

Inc(SomaCor);
if (SomaCor>=5) then
  SomaCor :=0;

if (hora[4] = '3') then
  inc(a);

SGDia.Cells[0,a] := IntToStr(Compromisso.CCodCompromisso);
SGDia.Cells[2,a] := Compromisso.CNomeCompromisso;
SGDia.Cells[3,a] := Compromisso.CDescCompromisso;
SGDia.Cells[4,a] := Compromisso.CFone1Compromisso;
SGDia.Cells[5,a] := Compromisso.CFone2Compromisso;
SGDia.Cells[6,a] := Compromisso.CStatus;
SGDia.Cells[7,a] := cor[SomaCor];

Duracao := StrToTime('00:30');

while (duracao < Compromisso.CDuracaoCompromisso) do
begin
  inc(a);
  SGDia.Cells[6,a] := Compromisso.CStatus;
  SGDia.Cells[0,a] := IntToStr(Compromisso.CCodCompromisso);
  SGDia.Cells[7,a] := cor[SomaCor];
  SGDia.Cells[1,a] := '';
  duracao := duracao + StrToTime('00:30');
end;

Compromisso.Proximo;

end;

end;
procedure TFormAgenda.FormCreate(Sender: TObject);
begin
  Compromisso := TColCompromisso.Create(Self);
  EdDataComp.Date := Now;
end;

procedure TFormAgenda.SGDiaDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin

if (SGDia.Cells[6,ARow] = '1') then
  with (Sender as TStringGrid).Canvas do
begin
  Brush.Color:= StringToColor(SGDia.Cells[7, Arow]);
  FillRect(Rect);
  Font.Color := clGreen;

  TextOut(Rect.Left + 3, Rect.Top + 3, SgDia.Cells[Acol,ARow]);
end
else
if (SgDia.Cells[6,ARow] = '0') then
  with (Sender as TStringGrid).Canvas do
begin
  Brush.Color:= StringToColor(SGDia.Cells[7, Arow]);
  FillRect(Rect);
  Font.Color := clRed;
  TextOut(Rect.Left + 3, Rect.Top + 3, SgDia.Cells[Acol,ARow]);
end
else
begin
  SGDia.Canvas.Pen.Color := clBlack;
  SGDia.Canvas.MoveTo(Rect.Left, Rect.Top);
  SGDia.Canvas.LineTo(Rect.Right, Rect.Top);
  SGDia.Canvas.MoveTo(Rect.Left, Rect.Bottom);
end;
end;

```

```

    SGDia.Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

if (SGDia.Cells[1,Arow] <> '') then
begin
    SGDia.Canvas.Pen.Color := clBlack;
    SGDia.Canvas.MoveTo(Rect.Left, Rect.Top);
    SGDia.Canvas.LineTo(Rect.Right, Rect.Top);
end;

end;

procedure TFormAgenda.EdDataCompDblClick(Sender: TObject);
begin
    ArrumaAgenda;
    BBAlterar.Enabled := False;
end;

procedure TFormAgenda.BBAlterarClick(Sender: TObject);
begin
    {if not assigned(FormCadAgenda) then
        FormCadAgenda := TFormCadAgenda.Create(Self);

    FormCadAgenda.CodigoCompromisso := StrToInt(SGDia.Cells[0, sel]);
    FormCadAgenda.ShowModal;
    FormCadAgenda := nil; }

    Atualizar;
end;

procedure TFormAgenda.SGDiaSelectCell(Sender: TObject; ACol, ARow: Integer;
var CanSelect: Boolean);
var a: integer;
begin
    BBAlterar.Enabled := True;

    sel := arow;
    a := arow;

    if (SGDia.Cells[0, Arow] <> '0') then
begin
    while (SGDia.Cells[3, a] = '') do
        if (a>0) then
            dec(a)
        else
            break;
    LabDescValor.Caption := SGDia.Cells[3, a];
    LabClienteValor.Caption := SGDia.Cells[2, a];
    LabFone1Valor.Caption := SGDia.Cells[4, a];
    LabFone2Valor.Caption := SGDia.Cells[5, a];
    if (SGDia.Cells[6, a] = '0') then
begin
        LabConfirma.Caption := 'À CONFIRMAR';
        LabConfirma.Font.Color := clRed;
    end
    else
begin
        LabConfirma.Caption := 'CONFIRMADO';
        LabConfirma.Font.Color := clGreen;
    end;
end
else
begin
    LabDescValor.Caption := '---';
    LabClienteValor.Caption := '---';
    LabFone1Valor.Caption := '---';
    LabFone2Valor.Caption := '';
end;
end;
else
begin
    LabDescValor.Caption := '---';
    LabClienteValor.Caption := '---';
    LabFone1Valor.Caption := '---';
    LabFone2Valor.Caption := '';
end;
end;

```

```
    LabConfirma.Caption := '';
  end;
end;

procedure TFormAgenda.SGDiaDblClick(Sender: TObject);
begin
  Atualizar;
end;

procedure TFormAgenda.Atualizar;
begin
  if not assigned(FormCadAgenda) then
    FormCadAgenda := TFormCadAgenda.Create(Self);

  FormCadAgenda.CodigoCompromisso := StrToInt(SGDia.Cells[0, sel]);
  FormCadAgenda.HoraAtual := SGDia.Cells[1, sel];
  FormCadAgenda.DataAtual := EdDataComp.Date;

  FormCadAgenda.ShowModal;
  FormCadAgenda := nil;
end;

procedure TFormAgenda.BBBuscarClick(Sender: TObject);
begin
  if not assigned(FormLocAgenda) then
    FormLocAgenda := TFormLocAgenda.Create(nil);

  FormLocAgenda.ShowModal;
  FormLocAgenda := nil;
end;

procedure TFormAgenda.BBSairClick(Sender: TObject);
begin
  Close;
end;

procedure TFormAgenda.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FreeOnRelease;
end;

end.
```

7 UNIT UCADAGENDA



FIGURA 07 - FORMCADAGENDA

```
///////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uCadAgenda;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, Mask, Buttons, ExtCtrls, uCompromisso, uAgenda,
  uColPaciente;

type
  TFormCadAgenda = class(TForm)
    CBPaciente: TComboBox;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    DataComp: TDateTimePicker;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    UDHora: TUpDown;
    LabTipo: TLabel;
    CBDesc: TComboBox;
    LabAtend: TLabel;
    LabHora: TLabel;
    MemoAnotacoes: TMemo;
    MEAtend: TMaskEdit;
    EdFone1: TEdit;
    EdFone2: TEdit;
    BBSalvar: TBitBtn;
```

```

BBSair: TBitBtn;
PanConfirma: TPanel;
RBAConfirmar: TRadioButton;
RBConfirmado: TRadioButton;
UDDuracao: TUpDown;
LabObs: TLabel;
BBExcluir: TBitBtn;
MEHora: TEdit;
MEDuracao: TEdit;
SBAdicionaAssunto: TSpeedButton;
procedure Button2Click(Sender: TObject);

procedure UDHoraChangingEx(Sender: TObject; var AllowChange: Boolean;
  NewValue: Smallint; Direction: TUpDownDirection);
procedure UDDuracaoChangingEx(Sender: TObject; var AllowChange: Boolean;
  NewValue: Smallint; Direction: TUpDownDirection);
procedure BBSalvarClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure CBPacienteChange(Sender: TObject);
procedure BBExcluirClick(Sender: TObject);
procedure BBSairClick(Sender: TObject);
procedure SBAdicionaAssuntoClick(Sender: TObject);
private
  { Private declarations }
  function ArrumaHora(valor: string; aumenta: boolean):string;
public
  { Public declarations }
  Compromisso: TCompromisso;
  CodigoCompromisso: Integer;

  DataAtual: TDate;
  HoraAtual: String;

  procedure CarregarPacientes;
  procedure ColocaAssuntos;
  procedure CarregaDados(CodCompromisso: Integer);
  procedure SalvaDados(CodCompromisso: Integer);
end;

var
  FormCadAgenda: TFormCadAgenda;
  CodigoPaciente: Integer;
  Pacientes: TColPaciente;
  PrxComp: TTime;

implementation

uses uAssuntos;

{$R *.dfm}

function TFormCadAgenda.ArrumaHora(valor: string;
  aumenta: boolean): string;
var
  hora: integer;
  hr, min: string;
begin
  hora := StrToInt(valor[1]+valor[2]);
  min := valor[4]+valor[5];

  if aumenta then
    begin
      if min = '30' then
        Inc(hora);
    end
  else
    begin
      if min = '00' then

```

```

        Dec(hora);
    end;

    if (hora = 24) then
        hora := 0;

    if (hora = -1) then
        hora := 23;

    if min = '00' then
        min := '30';
    else
        min := '00';

    if hora < 10 then
        hr := '0'+ IntToStr(hora)
    else
        hr := IntToStr(hora);

    Result := hr + ':' + min;
end;

procedure TFormCadAgenda.Button2Click(Sender: TObject);
begin
close;
end;

procedure TFormCadAgenda.UDHoraChangingEx(Sender: TObject;
var AllowChange: Boolean; NewValue: Smallint;
Direction: TUpDownDirection);
begin
if direction = UpdUp then
    MEHora.Text := ArrumaHora(MEHora.Text, true)
else
    if direction = UpdDown then
        MEHora.Text := ArrumaHora(MEHora.Text, false);
end;

procedure TFormCadAgenda.UDDuracaoChangingEx(Sender: TObject;
var AllowChange: Boolean; NewValue: Smallint;
Direction: TUpDownDirection);
begin
if direction = UpdUp then
begin
    if Copy(TimeToStr(PrxComp), 1, 5) <> MEDuracao.Text Then
        MEDuracao.Text := ArrumaHora(MEDuracao.Text, true);
end;

if direction = UpdDown then
begin
    if (MEDuracao.Text <> '00:30') then
        MEDuracao.Text := ArrumaHora(MEDuracao.Text, false);
end;
end;

procedure TFormCadAgenda.CarregaDados(CodCompromisso: Integer);
begin
Compromisso.Carregar(CodCompromisso);
CBPaciente.Text := Compromisso.PCliente;
EdFone1.Text := Compromisso.PFone1;
EdFone2.Text := Compromisso.PFone2;
DataComp.Date := Compromisso.PData;
MEHora.Text := TimeToStr(Compromisso.PHora);
MEDuracao.Text := Copy(TimeToStr(Compromisso.PDuracao), 1, 5);
if (Compromisso.PConfirmada = True) Then
    RBCConfirmado.Checked := True
else
    RBAConfirmar.Checked := True;
CBDesc.Text := Compromisso.PDescricao;
MEAAtend.Text := TimeToStr(Compromisso.PHoraAtend);
MemoAnotacoes.Text := Compromisso.PObservacao;

```

```

    CodigoPaciente := Compromisso.PCodPaciente;
end;

procedure TFormCadAgenda.SalvaDados(CodCompromisso: Integer);
begin
    Compromisso.PCodigo := CodCompromisso;
    Compromisso.PCliente := CBPaciente.Text;
    Compromisso.PCodPaciente := CodigoPaciente;
    Compromisso.PFone1 := EdFone1.Text;
    Compromisso.PFone2 := EdFone2.Text;
    Compromisso.PData := DataComp.Date;
    Compromisso.PHora := StrToTime(MEHora.Text);
    Compromisso.PDuracao := StrToTime(MEDuracao.Text);
    if (MEAtend.Text = ' : ') then
        Compromisso.PHoraAtend := StrToTime('00:00')
    else
        Compromisso.PHoraAtend := StrToTime(MEAtend.Text);
    Compromisso.PDescricao := CBDesc.Text;
    Compromisso.PObservacao := MemoAnotacoes.Text;
    if RBAConfirmar.Checked Then
        Compromisso.PConfirma := False
    Else
        Compromisso.PConfirma := True;
    if (CodCompromisso = 0) Then
        Compromisso.Salvar(0)
    else
        Compromisso.Salvar(1);
end;

procedure TFormCadAgenda.BBSalvarClick(Sender: TObject);
begin
    SalvaDados(CodigoCompromisso);
    FormAgenda.ArrumaAgenda;
    FormCadAgenda.Close;
end;

procedure TFormCadAgenda.FormCreate(Sender: TObject);
begin
    Compromisso := TCompromisso.Create(Self);
    ColocaAssuntos;
    CarregarPacientes;
end;

procedure TFormCadAgenda.FormShow(Sender: TObject);
begin
    if CodigoCompromisso <> 0 Then
        CarregaDados(CodigoCompromisso)
    else
    begin
        DataComp.Date := DataAtual;
        MEHora.Text := Copy(HoraAtual, 3, 5);
    end;

    PrxComp := Compromisso.SelecionaMinimo(DataComp.Date, StrToTime(MEHora.Text)) -
    StrToTime(MEHora.Text);
    if (PrxComp <= StrToTime('00:00')) then
        PrxComp := StrToTime('23:30') - StrToTime(MEHora.Text) + StrToTime('00:30');

end;

procedure TFormCadAgenda.CarregarPacientes;
begin
    Pacientes := TColPaciente.Create(Self);
    Pacientes.SelecionaAgenda;
    Pacientes.Primeiro;
    while not Pacientes.Final do
    begin
        Pacientes.ColocaAgenda;

```

```

CBPaciente.Items.Add(Pacientes.CNome);
Pacientes.Proximo;
end;
end;

procedure TFormCadAgenda.CBPacienteChange(Sender: TObject);
var
  a, b: integer;
begin
  Pacientes.Primeiro;
  a := CBPaciente.ItemIndex;
  if (a <> -1) then
    begin
      for b := 0 to a-1 do
        Pacientes.Proximo;
      Pacientes.ColocaAgenda;
      EdFone1.Text := Pacientes.CFoneRes;
      EdFone2.Text := Pacientes.CFoneCom;
      CodigoPaciente := Pacientes.CCodigo;
    end
  else
    begin
      EdFone1.Text := '';
      EdFone2.Text := '';
      CodigoPaciente := 0;
    end;
end;

procedure TFormCadAgenda.BBExcluirClick(Sender: TObject);
begin
  if (CodigoCompromisso <> 0) then
    if MessageDlg('Deseja excluir o compromisso?', mtConfirmation, [mbyes,mbno],0) = mryes
  then
    begin
      Compromisso.Salvar(2);
    end;
  FormAgenda.ArrumaAgenda;
  Close;
end;

procedure TFormCadAgenda.BBSairClick(Sender: TObject);
begin
  Close;
end;

procedure TFormCadAgenda.ColocaAssuntos;
var
  Arquivo: TextFile;
  Valor: String;
begin
  AssignFile(Arquivo,
             ExtractFilePath(Application.ExeName) + 'textos\assuntos.txt');
  Reset(Arquivo);
  CBDesc.Items.Clear;
  while not EOF(Arquivo) do
    begin
      Readln(Arquivo, Valor);
      CBDesc.Items.Add(Valor);
    end;
  CloseFile(Arquivo);
end;

procedure TFormCadAgenda.SBADicionaAssuntoClick(Sender: TObject);
begin
  if not assigned(FormAssuntos) then
    FormAssuntos := TFormAssuntos.Create(Self);

  FormAssuntos.ShowModal;
  FormAssuntos := nil;
end;
end.

```

8 UNIT ULOCAGENDA

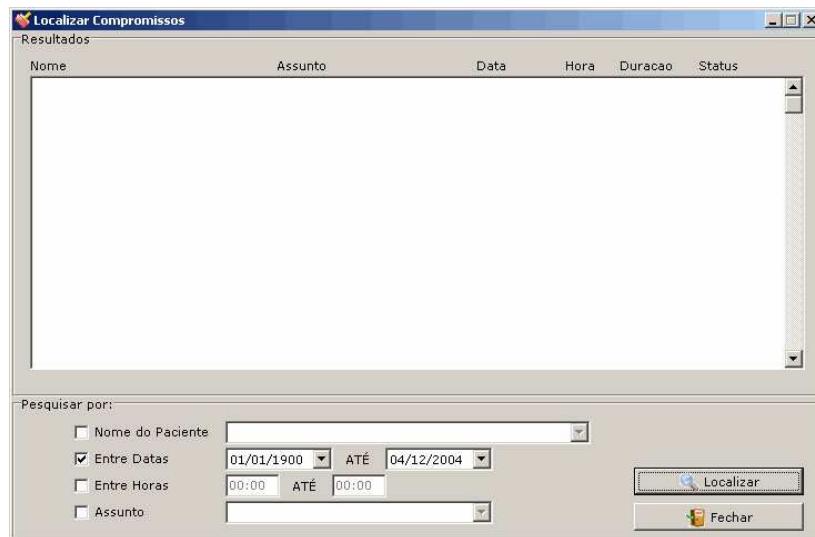


FIGURA 08 – FORMLOCAGENDA

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
unit uLocAgenda;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, StdCtrls, ComCtrls, Mask, Buttons, uColPaciente,
  uColCompromisso;

type
  TFormLocAgenda = class(TForm)
    Data2: TDateTimePicker;
    LabAtel: TLabel;
    Data1: TDateTimePicker;
    SGCompromissos: TStringGrid;
    CBNomes: TComboBox;
    GBOpcoes: TGroupBox;
    CBNome: TCheckBox;
    CBDData: TCheckBox;
    CBHora: TCheckBox;
    CBAssunto: TCheckBox;
    MEHoraIni: TMaskEdit;
    MEHoraFin: TMaskEdit;
    CBAssuntos: TComboBox;
    BBPesquisar: TBitBtn;
    BBSair: TBitBtn;
    LabNome: TLabel;
    LabAssunto: TLabel;
    LabHora: TLabel;
    LabDuracao: TLabel;
```

```

LabData: TLabel;
LabAte2: TLabel;
LabStatus: TLabel;
GBCompromisso: TGroupBox;
procedure FormCreate(Sender: TObject);
procedure CBNomeMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure CBDNomeMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure CBHoraMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure CBAssuntoMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure BBPesquisarClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure SGCompromissoDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure SGCompromissoSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
procedure SGCompromissoDblClick(Sender: TObject);
procedure BBSairClick(Sender: TObject);

private
  { Private declarations }
  procedure ArrumaGrade;
  procedure ColocaDados;
  procedure ColocaPacientes;
  procedure ColocaAssuntos;
  procedure Seleciona;
public
  { Public declarations }
  Pacientes: TColPaciente;
  Compromisso: TColCompromisso;
end;

var
  FormLocAgenda: TFormLocAgenda;
  ColocarPacientes, ColocarAssuntos: Boolean;
  Selecionado: TDate;

implementation

uses uAgenda;

{$R *.dfm}

procedure TFormLocAgenda.ArrumaGrade;
var
  a: integer;
begin
  SGCompromisso.RowCount := 1;
  SGCompromisso.ColWidths[0] := 0;
  SGCompromisso.ColWidths[1] := 218;
  SGCompromisso.ColWidths[2] := 180;
  SGCompromisso.ColWidths[3] := 80;
  SGCompromisso.ColWidths[4] := 50;
  SGCompromisso.ColWidths[5] := 55;
  SGCompromisso.ColWidths[6] := 90;
  for a := 0 to 7 do
    begin
      SGCompromisso.Cells[a,0] := '';
    end;
end;

procedure TFormLocAgenda.ColocaPacientes;
begin
  ColocarPacientes := True;
  Pacientes := TColPaciente.Create(Self);
  Pacientes.SelecionaAgenda;
  Pacientes.Primeiro;
  while not Pacientes.Final do

```

```

begin
  Pacientes.ColoaAgenda;
  CBNomes.Items.Add(Pacientes.CNome);
  Pacientes.Proximo;
end;
end;

procedure TFormLocAgenda.FormCreate(Sender: TObject);
begin
  ArrumaGrade;
  ColocarPacientes := False;
  ColocarAssuntos := False;
  Data1.DateTime := Now;
  Data2.DateTime := Now;
  Compromissos := TColCompromisso.Create(Self);
end;

procedure TFormLocAgenda.CBNomeMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if not ColocarPacientes then
    ColocaPacientes;
  if CBNome.Checked Then
    CBNomes.Enabled := True
  else
    CBNomes.Enabled := False;
end;

procedure TFormLocAgenda.CBDataMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if CBDData.Checked Then
    begin
      Data1.Enabled := True;
      Data2.Enabled := True;
    end
  else
    begin
      Data1.Enabled := False;
      Data2.Enabled := False;
    end;
end;

procedure TFormLocAgenda.CBHoraMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if CBHora.Checked Then
    begin
      MEHoraIni.Enabled := True;
      MEHoraFin.Enabled := True;
    end
  else
    begin
      MEHoraIni.Enabled := False;
      MEHoraFin.Enabled := False;
    end;
end;

procedure TFormLocAgenda.CBAssuntoMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if CBAssunto.Checked Then
    CBAssuntos.Enabled := True
  else
    CBAssuntos.Enabled := False;
  ColocaAssuntos;
end;

procedure TFormLocAgenda.ColocaAssuntos;
var
  assuntos: TextFile;

```

```

Item: String;
begin
  AssignFile(assuntos, ExtractFilePath(Application.ExeName)+ 'textos\assuntos.txt');
  Reset(assuntos);
  while not EOF(assuntos) do
    begin
      Readln(assuntos, item);
      CBAssuntos.Items.Add(Item);
    end;
  end;

procedure TFormLocAgenda.BBPesquisarClick(Sender: TObject);
begin
  ColocaDados;
end;

procedure TFormLocAgenda.ColocaDados;
var
  a: Integer;
begin
  ArrumaGrade;
  a := 0;
  if MEHoraIni.Text = ' : ' then
    MEHoraIni.Text := '00:00';
  if MEHoraFin.Text = ' : ' then
    MEHoraFin.Text := '00:00';
  Compromissos.Pesquisa(CBNomes.Text, CBAssuntos.Text, Data1.Date, Data2.Date,
  StrToTime(MEHoraIni.Text), StrToTime(MEHoraFin.Text), CBNome.Checked,
  CBAssunto.Checked, CBData.Checked, CBHora.Checked);
  Compromissos.Primeiro;
  while not Compromissos.Final do
    begin
      Compromissos.ColocaPesquisa;
      SGCompromissos.Cells[0, a] := IntToStr(Compromissos.CCodCompromisso);
      SGCompromissos.Cells[1, a] := Compromissos.CNomeCompromisso;
      SGCompromissos.Cells[2, a] := Compromissos.CDescCompromisso;
      SGCompromissos.Cells[3, a] := DateToStr(Compromissos.CDataCompromisso);
      SGCompromissos.Cells[4, a] := Copy(TimeToStr(Compromissos.CHoraCompromisso), 1, 5);
      SGCompromissos.Cells[5, a] := Copy(TimeToStr(Compromissos.CDuracaoCompromisso), 1 ,
      5);
      if Compromissos.CStatus = '1' Then
        SGCompromissos.Cells[6, a] := 'CONFIRMADO'
      else
        SGCompromissos.Cells[6, a] := 'À CONFIRMAR';
      Inc(a);
      SGCompromissos.RowCount := SGCompromissos.RowCount + 1;
      Compromissos.Proximo;
    end;
  SGCompromissos.RowCount := SGCompromissos.RowCount - 1;
end;

procedure TFormLocAgenda.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

procedure TFormLocAgenda.SGCompromissosDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormLocAgenda.SGCompromissosSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if SGCompromissos.Cells[3, ARow] <> '' Then

```

```
    Seleccionado := StrToDate(SGCompromissos.Cells[3, ARow])
else
    Seleccionado := StrToDate('30/12/1899');
end;

procedure TFormLocAgenda.Seleciona;
begin
    FormAgenda.EdDataComp.Date := Seleccionado;
    FormAgenda.ArrumaAgenda;
    Close;
end;

procedure TFormLocAgenda.SGCompromissosDblClick(Sender: TObject);
begin
    if Seleccionado <> StrToDate('30/12/1899') Then
        Seleciona;
end;

procedure TFormLocAgenda.BBSairClick(Sender: TObject);
begin
    Close;
end;

end.
```

9 UNIT USELORC

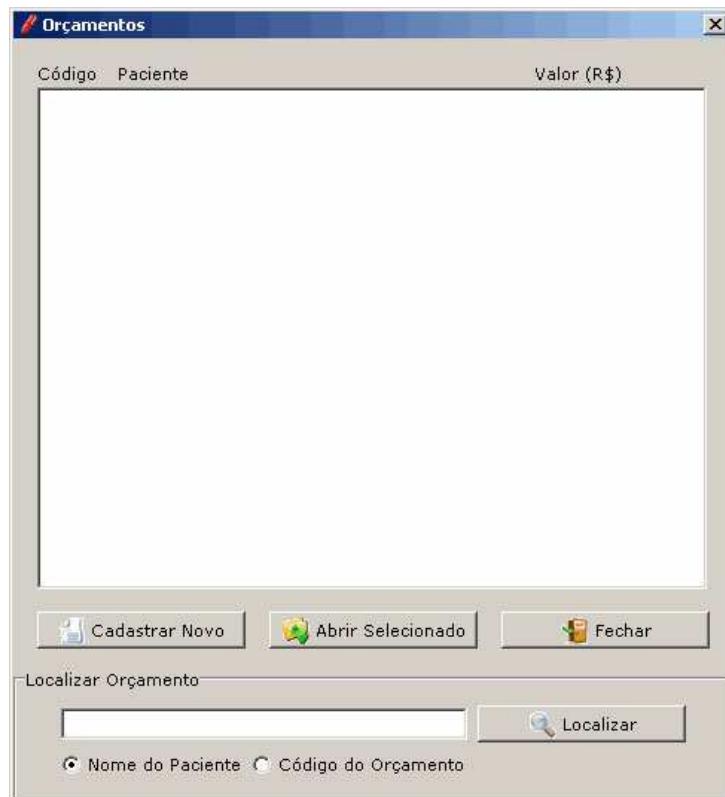


FIGURA 09 - FORMCOLORCAMENTO

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uSelOrc;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, Buttons, uColOrcamento, Math;

type
  TFormColOrcamento = class(TForm)
    GBLocalizar: TGroupBox;
    SGOrçamentos: TStringGrid;
    BBInserir: TBitBtn;
    BBAAlterar: TBitBtn;
    BBSair: TBitBtn;
    LabCodigo: TLabel;
    LabPaciente: TLabel;
    LabTotal: TLabel;
    EdLocalizar: TEdit;
  end;
```

```

RBNome: TRadioButton;
RBCodigo: TRadioButton;
BBLocalizar: TBitBtn;
procedure FormCreate(Sender: TObject);
procedure SGOrcamentosDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure BBInserirClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure SGOrcamentosSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure BBLocalizarClick(Sender: TObject);
procedure RBCodigoClick(Sender: TObject);
procedure RBNomeClick(Sender: TObject);
procedure EdLocalizarKeyPress(Sender: TObject; var Key: Char);
procedure BBAalterarClick(Sender: TObject);
procedure SGOrcamentosDblClick(Sender: TObject);
procedure BBSairClick(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
  ColOrcamento: TColOrcamento;
  procedure ArrumarGrade;
  procedure ColocarDados(Campo, Valor: String);
  procedure AbrirSelecionado;
end;

var
  FormColOrcamento: TFormColOrcamento;
  Linha: Integer;

implementation

uses uCadOrc;

{$R *.dfm}

procedure TFormColOrcamento.ArrumarGrade;
begin
  SGOrcamentos.RowCount := 1;
  SGOrcamentos.ColWidths[0] := 50;
  SGOrcamentos.ColWidths[1] := 280;
  SGOrcamentos.ColWidths[2] := 100;
  SGOrcamentos.Cells[0,0] := '';
  SGOrcamentos.Cells[1,0] := '';
  SGOrcamentos.Cells[2,0] := '';
end;

procedure TFormColOrcamento.ColarDados(Campo, Valor: String);
var
  a: integer;
begin
  ArrumarGrade;
  ColOrcamento.Seleciona(Campo, Valor);
  ColOrcamento.Primeiro;
  a := 0;
  while not ColOrcamento.Final do
    begin
      ColOrcamento.Cola;
      SGOrcamentos.Cells[0, a] := IntToStr(ColOrcamento.CCodOrcamento);
      SGOrcamentos.Cells[1, a] := ColOrcamento.CNomePaciente;
      SGOrcamentos.Cells[2, a] :=
        FloatToStr(RoundTo(ColOrcamento.CValor, -2));
      Inc(a);
      SGOrcamentos.RowCount := a+1;
      ColOrcamento.Proximo;
    end;
  SGOrcamentos.RowCount := a;
  Linha := -1;
end;

```

```

procedure TFormColOrcamento.FormCreate(Sender: TObject);
begin
  ColOrcamento := TColOrcamento.Create(Self);
  ColocarDados('','');
  Linha := -1;
end;

procedure TFormColOrcamento.SGOrcamentosDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormColOrcamento.BBInserirClick(Sender: TObject);
begin
  if not assigned(FormCadOrcamento) Then
    FormCadOrcamento := TFormCadOrcamento.Create(nil);
  FormCadOrcamento.Orcamento.PCodigo := 0;
  FormCadOrcamento.DTData.Date := Now;
  FormCadOrcamento.DTDataPriParcela.Date := Now;
  FormCadOrcamento.Orcamento.PDataPriParc := Now;
  FormCadOrcamento.ShowModal;
  FormCadOrcamento := nil;
end;

procedure TFormColOrcamento.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

procedure TFormColOrcamento.SGOrcamentosSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  Linha := ARow;
end;

procedure TFormColOrcamento.AbrirSelecionado;
begin
  if Linha <> -1 Then
    begin
      if not assigned(FormCadOrcamento) Then
        FormCadOrcamento := TFormCadOrcamento.Create(nil);
      FormCadOrcamento.Orcamento.PCodigo :=
        StrToInt(SGOrcamentos.Cells[0, Linha]);
      FormCadOrcamento.GBOrcamento.Enabled := False;
      FormCadOrcamento.ShowModal;
      FormCadOrcamento := nil;
    end
  else
    MessageDlg('Não há orçamento selecionado!', mtError, [mbOk], 0);
end;

procedure TFormColOrcamento.BBLocalizarClick(Sender: TObject);
var
  Campo: String;
begin
  if RBCodigo.Checked Then
    Campo := 'a.orc_cod'
  else
    Campo := 'c.pac_nome';
  ColocarDados(Campo, EdLocalizar.Text);
end;

procedure TFormColOrcamento.RBCodigoClick(Sender: TObject);
begin
  EdLocalizar.Text := '';

```

```
end;

procedure TFormColOrcamento.RBNomeClick(Sender: TObject);
begin
  EdLocalizar.Text := '';
end;

procedure TFormColOrcamento.EdLocalizarKeyPress(Sender: TObject;
  var Key: Char);
begin
  if RBCodigo.Checked Then
    if not (key in['0'..'9', Chr(8)]) then
      key := #0;
end;

procedure TFormColOrcamento.BBAAlterarClick(Sender: TObject);
begin
  AbrirSeleccionado;
end;

procedure TFormColOrcamento.SGOrcamentosDblClick(Sender: TObject);
begin
  AbrirSeleccionado;
end;

procedure TFormColOrcamento.BBSairClick(Sender: TObject);
begin
  Close;
end;

end.
```

10 UNIT UCADORC

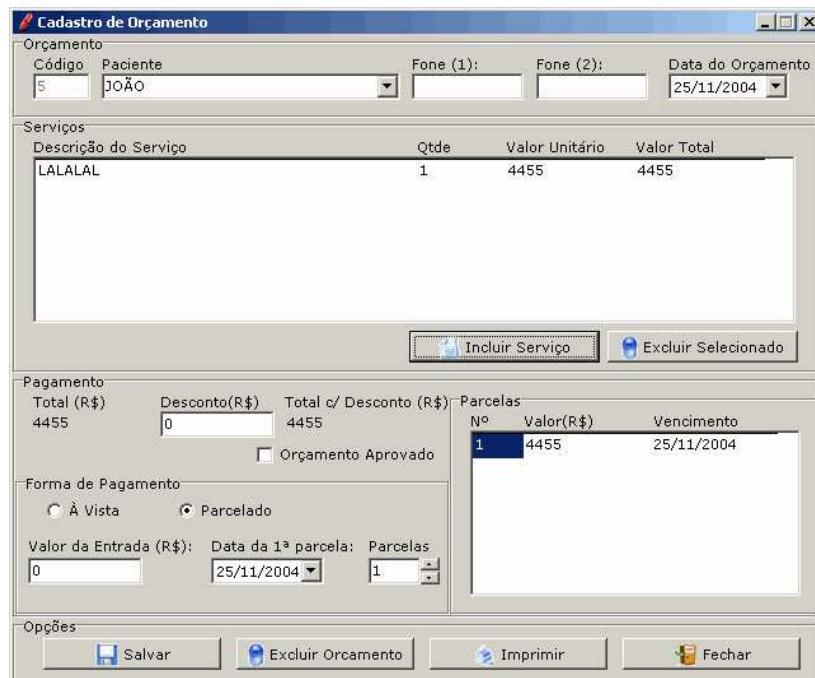


FIGURA 10 – FORCADORCAMENTO

```
///////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uCadOrc;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, ExtCtrls, jpeg, Grids, Mask, Buttons,
  ExtDlgs, uOrcamento, math, uColPaciente, uRelOrcamento;

type
  TFormCadOrcamento = class(TForm)
    SGCadOrc: TStringGrid;
    LabPaciente: TLabel;
    CBAprovado: TCheckBox;
    LabTotal: TLabel;
    RBAVista: TRadioButton;
    RBParc: TRadioButton;
    EdNumParcelas: TEdit;
    UDPArq: TUpDown;
    LabParcela: TLabel;
    LabPrimeira: TLabel;
    EdValEnt: TEdit;
    LbValEnt: TLabel;
  end;
```

```

LabDesc: TLabel;
EdDesc: TEdit;
LabTotalDesc: TLabel;
LabData: TLabel;
DTData: TDateTimePicker;
SGParcelas: TStringGrid;
DTDatapriparcela: TDateTimePicker;
BBSalvar: TBitBtn;
BBImprimir: TBitBtn;
BBIncluir: TBitBtn;
BBExcluir: TBitBtn;
BBSair: TBitBtn;
BBExcluirOrc: TBitBtn;
CBPaciente: TComboBox;
LabParNum: TLabel;
LabVenc: TLabel;
LabValor: TLabel;
LbQuantidade: TLabel;
LbDescServ: TLabel;
LbValUnit: TLabel;
LbValTot: TLabel;
GOrcamento: TGroupBox;
GServicos: TGroupBox;
GPagamento: TGroupBox;
EdCodigo: TEdit;
LabCodigo: TLabel;
LabFone1: TLabel;
LabFone2: TLabel;
EdFone1: TEdit;
EdFone2: TEdit;
GBCParcelas: TGroupBox;
GBForma: TGroupBox;
PnParcelado: TPanel;
GOpcoes: TGroupBox;
LabTotProc: TLabel;
LabTotDesc: TLabel;
procedure EdFone1KeyPress(Sender: TObject; var Key: Char);
procedure SGCadOrcDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure FormCreate(Sender: TObject);
procedure BBIncluirClick(Sender: TObject);
procedure EdDescChange(Sender: TObject);
procedure EdDescKeyPress(Sender: TObject; var Key: Char);
procedure EdDescExit(Sender: TObject);
procedure RBParcClick(Sender: TObject);
procedure RBAVistaClick(Sender: TObject);
procedure BBExcluirClick(Sender: TObject);
procedure BBSalvarClick(Sender: TObject);
procedure CBPacienteChange(Sender: TObject);
procedure EdValEntChange(Sender: TObject);
procedure DTDatapriparcelaChange(Sender: TObject);
procedure EdNumParcelasChange(Sender: TObject);
procedure SGCadOrcSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BBExcluirOrcClick(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure BBSairClick(Sender: TObject);
procedure BBImprimirClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
Orcamento: TOrcamento;
Pacientes: TColPaciente;
procedure ArrumaGrades;
procedure LerDados;
procedure Recalcular;
procedure SalvarDados;
procedure CarregarPacientes;

```

```

procedure ColocarParcelas(Total, Entrada: Real; NumParc: Integer;
                           DataPrimeira: TDate);
begin

var
  FormCadOrcamento: TFormCadOrcamento;
  Serv, CodigoPaciente: Integer;
  Fechar: Boolean;

implementation

uses uInserirServico, uSelOrc;

{$R *.dfm}

procedure TFormCadOrcamento.EdFone1KeyPress(Sender: TObject;
  var Key: Char);
begin
  if not (key in['0'..'9', '(', ')', Chr(8)]) then
    key := #0;
end;

procedure TFormCadOrcamento.SGCadOrcDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormCadOrcamento.ArrumaGrades;
begin
  SGCadOrc.RowCount := 1;
  SGCadOrc.ColWidths[0] := 0;
  SGCadOrc.ColWidths[1] := 300;
  SGCadOrc.ColWidths[2] := 70;
  SGCadOrc.ColWidths[3] := 102;
  SGCadOrc.ColWidths[4] := 102;
  SGParcelas.RowCount := 1;
  SGParcelas.ColWidths[0] := 40;
  SGParcelas.ColWidths[1] := 100;
  SGParcelas.ColWidths[2] := 100;
end;

procedure TFormCadOrcamento.LerDados;
begin
  Orcamento.Carregar;
  EdCodigo.Text := IntToStr(Orcamento.PCodigo);
  CBPaciente.Text := Orcamento.PNomePaciente;
  EdFone1.Text := Orcamento.PFone1;
  EdFone2.Text := Orcamento.PFone2;
  CodigoPaciente := Orcamento.PCodPac;
  DTData.Date := Orcamento.PDataOrcamento;
  Recalcular;
end;

procedure TFormCadOrcamento.SalvarDados;
begin
  if CBPaciente.Text = '' Then
    begin
      MessageDlg('Não há paciente selecionado!', mtError, [mbOk], 0);
    end
  else
    if CodigoPaciente = 0 Then
      begin
        MessageDlg('Paciente não cadastrado!', mtError, [mbOk], 0);
      end
  else
    if SGCadOrc.Cells[0,0] = '' Then

```

```

begin
  MessageDlg('Não há serviços no orçamento!', mtError, [mbOk], 0);
end
else
begin
  Orcamento.PNomePaciente := CBPaciente.Text;
  Orcamento.PFone1 := EdFone1.Text;
  Orcamento.PFone2 := EdFone2.Text;
  Orcamento.PCodPac := CódigoPaciente;
  Orcamento.PDataOrcamento := DTData.Date;
  Orcamento.PAprovado := CBAprovado.Checked;
  Orcamento.PFormaPg := RBAVista.Checked;
  Orcamento.PValorEntrada := StrToFloat(EdValEnt.Text);
  Orcamento.PDataPriParc := DTDataPriParcela.Date;
  Orcamento.PDesconto := StrToFloat(EdDesc.Text);
  Orcamento.PNumParc := StrToInt(EdNumParcelas.Text);
  if Orcamento.PCodigo = 0 Then
    Orcamento.Salvar(0)
  else
    Orcamento.Salvar(1);
end;
end;

procedure TFormCadOrcamento.FormCreate(Sender: TObject);
begin
  Orcamento := TOrcamento.Create(Self);
  ArrumaGrades;
end;

procedure TFormCadOrcamento.BBIncluirClick(Sender: TObject);
begin
  if NOT ASSIGNED(FormInserirServico) Then
    FormInserirServico := TFormInserirServico.Create(nil);
  FormInserirServico.ShowModal;
  FormInserirServico := Nil;
end;

procedure TFormCadOrcamento.EdDescChange(Sender: TObject);
begin
  if EdDesc.Text = '' Then
    Orcamento.PDesconto := 0
  Else
    Orcamento.PDesconto := StrToFloat(EdDesc.Text);
  LabTotDesc.Caption := FloatToStr(RoundTo(Orcamento.LerDesc, -2));
  if RBParc.Checked Then
    ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada,
                    Orcamento.PNumParc,Orcamento.PDataPriParc);
end;

procedure TFormCadOrcamento.EdDescKeyPress(Sender: TObject; var Key: Char);
var
  a: integer;
  comvir: boolean;
begin
  comvir := False;
  if not (key in['0'..'9', ',', Chr(8)]) then
    key := #0;

  if key = ',' then
    for a:= 1 to Length(TEdit(Sender).Text) do
      if TEdit(Sender).Text[a] = ',' Then
        comvir := True;
  if comvir Then
    Key := #0;
end;

procedure TFormCadOrcamento.EdDescExit(Sender: TObject);
begin
  TEdit(Sender).Text :=
    FloatToStr(RoundTo(StrToFloat(TEdit(Sender).Text), -2));
  if TEdit(Sender).Text = '' Then

```

```

        TEdit(Sender).Text := '0';
end;

procedure TFormCadOrcamento.RBParcClick(Sender: TObject);
begin
    RBParc.Checked := True;
    PnParcelado.Visible := True;
    ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada,
                    Orcamento.PNumParc, Orcamento.PDataPriParc);

end;

procedure TFormCadOrcamento.RBAVistaClick(Sender: TObject);
begin
    RBAVista.Checked := True;
    PnParcelado.Visible := False;
    SGParcelas.RowCount := 1;
    SGParcelas.Cells[0,0] := '';
    SGParcelas.Cells[1,0] := '';
    SGParcelas.Cells[2,0] := '';
end;

procedure TFormCadOrcamento.BBExcluirClick(Sender: TObject);
begin
    if (Serv <> -1) then
    begin
        if (Orcamento.Servico[Serv].Situacao = 0) or
            (Orcamento.Servico[Serv].Situacao = 2) Then
            Orcamento.Servico[Serv].Situacao := 3
        else
            Orcamento.Servico[Serv].Situacao := 4;
        Recalcular;
    end;
end;

procedure TFormCadOrcamento.Recalcular;
var
    a, b: Integer;
begin
    SGCadOrc.Cells[0,0] := '';
    SGCadOrc.Cells[1,0] := '';
    SGCadOrc.Cells[2,0] := '';
    SGCadOrc.Cells[3,0] := '';
    SGCadOrc.Cells[4,0] := '';
    LabTotProc.Caption := FloatToStr(RoundTo(Orcamento.LerTotal, - 2));
    LabTotDesc.Caption := FloatToStr(RoundTo(Orcamento.LerDesc, - 2));
    EdDesc.Text := FloatToStr(Orcamento.PDesconto);
    if Orcamento.PFormaPg Then
        begin
            RBAVista.Checked := True;
            PnParcelado.Visible := False;
        end
    Else
        begin
            RBParc.Checked := True;
            PnParcelado.Visible := True;
            ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada,
                            Orcamento.PNumParc, Orcamento.PDataPriParc);
        end;
    EdValEnt.Text := FloatToStr(Orcamento.PValorEntrada);
    DTDataPriParcela.Date := Orcamento.PDataPriParc;
    EdNumParcelas.Text := IntToStr(Orcamento.PNumParc);
    UDParc.Position := Orcamento.PNumParc;
    CBAprovado.Checked := Orcamento.PAprovado;
    b := 0;
    for a:= 0 to High(Orcamento.Servico) do
        begin
            if Orcamento.Servico[a].Situacao < 3 Then
                begin
                    SGCadOrc.RowCount := b + 1;
                    SGCadOrc.Cells[0, b] := IntToStr(a);
                end;
        end;
end;

```

```

    SG_CadOrc.Cells[1, b] := Orcamento.Servico[a].PServNome;
    SG_CadOrc.Cells[2, b] := IntToStr(Orcamento.Servico[a].PQtde);
    SG_CadOrc.Cells[3, b] := FloatToStr(Orcamento.Servico[a].PValSer);
    SG_CadOrc.Cells[4, b] := FloatToStr(Orcamento.Servico[a].Total);
    Inc(b);
end;
end;
Serv := -1;
end;

procedure TFormCadOrcamento.BBSalvarClick(Sender: TObject);
begin
  SalvarDados;
end;

procedure TFormCadOrcamento.CarregarPacientes;
begin
  Pacientes := TColPaciente.Create(Self);
  Pacientes.SelecionaAgenda;
  Pacientes.Primeiro;
  while not Pacientes.Final do
  begin
    Pacientes.ColocaAgenda;
    CBPaciente.Items.Add(Pacientes.CNome);
    Pacientes.Proximo;
  end;
end;

procedure TFormCadOrcamento.CBPacienteChange(Sender: TObject);
var
  a, b: Integer;
begin
  Pacientes.Primeiro;
  a := CBPaciente.ItemIndex;

  if (a <> -1) then
  begin
    for b := 0 to a-1 do
      Pacientes.Proximo;
    Pacientes.ColocaAgenda;
    EdFone1.Text := Pacientes.CFoneRes;
    EdFone2.Text := Pacientes.CFoneCom;
    CodigoPaciente := Pacientes.CCodigo;
  end
  else
  begin
    EdFone1.Text := '';
    EdFone2.Text := '';
    CodigoPaciente := 0;
  end;
end;

procedure TFormCadOrcamento.ColocarParcelas(Total, Entrada: Real;
  NumParc: Integer; DataPrimeira: TDate);
var
  a: Integer;
  divisao, parcela: Real;
  Data: TDate;
begin
  SG_Parcelas.RowCount := NumParc;
  SG_Parcelas.Cells[0,0] := '';
  SG_Parcelas.Cells[0,1] := '';
  SG_Parcelas.Cells[0,2] := '';
  divisao := Total - Entrada;
  if NumParc = 0 Then
    NumParc := 1;
  parcela := divisao/NumParc;
  parcela := RoundTo(parcela, -2);
  Data := DataPrimeira;
  for a:= 0 to NumParc - 1 do
  begin
    begin
      SG_Parcelas.Cells[1, a] := IntToStr(parcela);
      SG_Parcelas.Cells[2, a] := FloatToStr(divisao);
      SG_Parcelas.Cells[3, a] := Data.ToString('dd/MM/yyyy');
    end;
  end;
end;

```

```

    SGParcelas.Cells[0, a] := IntToStr(a+1);
    SGParcelas.Cells[1, a] := FloatToStr(parcela);
    SGParcelas.Cells[2, a] := DateToStr(Data);
    Data := IncMonth(Data, 1);
  end;
end;

procedure TFormCadOrcamento.EdValEntChange(Sender: TObject);
begin
  if (EdValEnt.Text <> '') Then
    Orcamento.PValorEntrada := StrToFloat(EdValEnt.Text)
  Else
    Orcamento.PValorEntrada := 0;
  ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada, Orcamento.PNumParc,
                  Orcamento.PDataPriParc);
end;

procedure TFormCadOrcamento.DTDatapriparcelaChange(Sender: TObject);
begin
  Orcamento.PDataPriParc := DTDataPriParcela.Date;
  ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada,
                  Orcamento.PNumParc, Orcamento.PDataPriParc);
end;

procedure TFormCadOrcamento.EdNumParcelasChange(Sender: TObject);
begin
  Orcamento.PNumParc := StrToInt(EdNumParcelas.Text);
  ColocarParcelas(Orcamento.LerDesc, Orcamento.PValorEntrada,
                  Orcamento.PNumParc, Orcamento.PDataPriParc);
end;

procedure TFormCadOrcamento.SGCadOrcSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if SGCadOrc.Cells[0, ARow] <> '' Then
    Serv := StrToInt(SGCadOrc.Cells[0, ARow])
  else
    Serv := -1;
end;

procedure TFormCadOrcamento.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if not Fechar then
  begin
    if MessageDlg('Deseja salvar orçamento?', mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
      begin
        SalvarDados;
        if CBPaciente.Text = '' Then
          Abort;
      end;
    end;
    FormColOrcamento.ColocarDados('','');
    FreeOnRelease;
  end;
end;

procedure TFormCadOrcamento.BBExcluirOrcClick(Sender: TObject);
begin
  if MessageDlg('Deseja excluir o orçamento?',
                mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
    begin
      Orcamento.Salvar(2);
      Fechar := True;
      Close;
    end;
end;

procedure TFormCadOrcamento.FormShow(Sender: TObject);
begin
  Fechar := False;

```

```

if Orcamento.PCodigo = 0 Then
  CarregarPacientes
Else
  LerDados;
Serv := -1;
end;

procedure TFormCadOrcamento.BBSairClick(Sender: TObject);
begin
  Close;
end;

procedure TFormCadOrcamento.BBImprimirClick(Sender: TObject);
begin
  if MessageDlg('Você precisa salvar o orçamento antes de imprimir! Deseja '
    +'salvar?', mtConfirmation, [mbYes, mbNo], 0) = mrYes Then
  begin
    SalvarDados;
    if NOT ASSIGNED(ReOrcamento)Then
      ReOrcamento := TReOrcamento.Create(Self);
    with ReOrcamento do
      begin
        LabPacValor.Caption := CBPaciente.Text;
        LabNumeroValor.Caption := EdCodigo.Text;
        LabDataValor.Caption := DateToStr(DTData.Date);
        LabTotalValor.Caption := LabTotProc.Caption;
        LabDescontoValor.Caption := EdDesc.Text;
        LabTotDescValor.Caption := LabTotDesc.Caption;
        if RBAVista.Checked Then
          begin
            LabFormaValor.Caption := 'À VISTA';
            LabNumParcValor.Caption := '';
            LabDataPriValor.Caption := '';
            LabDataUltValor.Caption := '';
            LabValorParcValor.Caption := '';
            LabEntradaValor.Caption := '';
          end
        else
          begin
            LabFormaValor.Caption := 'PARCELADO';
            LabNumParcValor.Caption := EdNumParcelas.Text;
            LabDataPriValor.Caption := DateToStr(DTDataPriParcela.Date);
            LabDataUltValor.Caption :=
              SGParcelas.Cells[2, SGParcelas.RowCount-1];
            LabValorParcValor.Caption :=
              SGParcelas.Cells[1, SGParcelas.RowCount-1];
            LabEntradaValor.Caption := EdValEnt.Text;
          end;
        QServicos.Close;
        QServicos.ParamByName('codorc').AsInteger := Orcamento.PCodigo;
        QServicos.ExecSQL;
        QServicos.Open;
      end;
    ReOrcamento.QROrcamento.Preview;
  end;
end;

```

11 UNIT UINSERIRSERVICO



FIGURA 11 – FORMINSERIRSERVICO

```

///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
/////////////////////////////
unit uInserirServico;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, uColServicos, ComCtrls, StdCtrls, Buttons, uOrcamento;

type
  TFormInserirServico = class(TForm)
    SGServicos: TStringGrid;
    LabValor: TLabel;
    LabNome: TLabel;
    BBInserir: TBitBtn;
    LabQtde: TLabel;
    EdQtde: TEdit;
    UDQtde: TUpDown;
    BBCancelar: TBitBtn;
    GBPesquisar: TGroupBox;
    EdLocalizar: TEdit;
    Localizar: TBitBtn;
  procedure FormCreate(Sender: TObject);
  procedure SGServicosDrawCell(Sender: TObject; ACol, ARow: Integer;
    Rect: TRect; State: TGridDrawState);
  procedure LocalizarClick(Sender: TObject);
  procedure BBCancelarClick(Sender: TObject);
  procedure BBInserirClick(Sender: TObject);
  procedure SGServicosSelectCell(Sender: TObject; ACol, ARow: Integer;
    var CanSelect: Boolean);
private
  { Private declarations }
public
  { Public declarations }
end;

```

```

ColServicos: TColServicos;
procedure ArrumaGrade;
procedure LerServicos(Nome: String);
end;

var
FormInserirServico: TFormInserirServico;
CodigoServico: Integer;
Valor: Real;
Nome: String;

implementation

uses UCadOrc;

{$R *.dfm}

procedure TFormInserirServico.ARRUMAGRADE;
begin
SGServicos.RowCount := 1;
SGServicos.ColCount := 3;
SGServicos.ColWidths[0] := 0;
SGServicos.ColWidths[1] := 310;
SGServicos.ColWidths[2] := 70;
SGServicos.Cells[0,0] := '';
SGServicos.Cells[1,0] := '';
SGServicos.Cells[2,0] := '';
end;

procedure TFormInserirServico.LerServicos(Nome: String);
var
a: Integer;
begin
ArrumaGrade;
a := 0;
ColServicos.Seleciona(Nome);
ColServicos.Primeiro;
SGServicos.RowCount := 1;
while not ColServicos.Final do
begin
ColServicos.Coloa;
SGServicos.Cells[0, a] := IntToStr(ColServicos.CCódigo);
SGServicos.Cells[1, a] := ColServicos.CNome;
SGServicos.Cells[2, a] := FloatToStr(ColServicos.CValor);
Inc(a);
SGServicos.RowCount := a+1;
ColServicos.Proximo;
end;
SGServicos.RowCount := SGServicos.RowCount - 1;
end;

procedure TFormInserirServico.FormCreate(Sender: TObject);
begin
ColServicos := TColServicos.Create(Self);
CodigoServico := 0;
LerServicos('');
end;

procedure TFormInserirServico.SGServicosDrawCell(Sender: TObject; ACol,
ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
TStringGrid(Sender).Canvas.Pen.Color := clBlack;
TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormInserirServico.LocalizarClick(Sender: TObject);
begin
LerServicos(EdLocalizar.Text);

```

```

end;

procedure TFormInserirServico.BBCancelarClick(Sender: TObject);
begin
  Close;
end;

procedure TFormInserirServico.BBInserirClick(Sender: TObject);
var
  a, b, c: integer;
  confere: boolean;
begin
  ifCodigoServico = 0 Then
    MessageDlg('Não há serviço selecionado!', mtError, [mbOk], 0)
  else
    begin
      b := High(FormCadOrcamento.Orcamento.Servico);
      confere := False;
      c:=0;
      for a:= 0 to b do
        begin
          if FormCadOrcamento.Orcamento.Servico[a].Situacao < 3 Then
            begin
              Inc(c);
              if FormCadOrcamento.Orcamento.Servico[a].PCodSer =
                CodigoServico Then
                  begin
                    confere := True;
                    break;
                  end;
            end;
        end;
      if confere Then
        begin
          if MessageDlg('Serviço já adicionado! Deseja adicionar mais '+
            EdQtde.Text + ' ao orçamento?', mtConfirmation,
            [mbYes, mbNo], 0) = mrYes Then
            begin
              with FormCadOrcamento.Orcamento.Servico[a] do
                begin
                  PQtde := PQtde + StrToInt(EdQtde.Text);
                  Total := PQtde * PValSer;
                  if Situacao = 0 Then
                    Situacao := 2
                  else
                    Situacao := 1;
                end;
              FormCadOrcamento.Recalcular;
            end;
        end;
      end;
    else
      begin
        if (c >= 10) Then
          MessageDlg('Não é possível mais adicionar serviços! O '+
            'máximo permitido é 10 (dez)', mtError, [mbOk], 0)
        else
          begin
            SetLength(FormCadOrcamento.Orcamento.Servico, b+2);
            FormCadOrcamento.Orcamento.Servico[b+1] :=
              TORCServ.Create;
            with FormCadOrcamento.Orcamento.Servico[b+1] do
              begin
                PCodigo := 0;
                PCodOrc := FormCadOrcamento.Orcamento.PCodigo;
                PCodSer := CodigoServico;
                PValSer := Valor;
                PServNome := Nome;
                PQtde := StrToInt(EdQtde.Text);
                Total := PQtde * PValSer;
                Situacao := 1;
                FormCadOrcamento.Recalcular;
              end;
          end;
      end;
    end;
  end;
end;

```

```
        end;
    end;
end;
Close;
end;
begin
  if SGServicos.Cells[0, Arow] <> '' Then
    begin
      CodigoServico := StrToInt(SGServicos.Cells[0, Arow]);
      Valor := StrToFloat(SGServicos.Cells[2, Arow]);
      Nome := SGServicos.Cells[1, Arow];
    end;
end;
end.
```

12 UNIT URELOR CAMENTO

FIGURA 12 - RELORCAMENTO

```
//
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uRelOrcamento;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, QuickRpt, ExtCtrls, QRCtrls, DB, DBTables, IniFiles;

type
  TRelOrcamento = class(TForm)
    QROrçamento: TQuickRep;
    PageHeaderBand1: TQRBand;
    TitleBand1: TQRBand;
    LabNomeClinica: TQRLabel;
    LabEndereco: TQRLabel;
    LabPaciente: TQRLabel;
    LabPacValor: TQRLabel;
    LabData: TQRLabel;
    LabDataValor: TQRLabel;
    ImgTitulo: TQRImage;
    ImgNome: TQRImage;
    LabOrcamento: TQRLabel;
    ImgServicos: TQRImage;
    LabServiço: TQRLabel;
    LabQtde: TQRLabel;
    LabValor: TQRLabel;
```

```

DetailBand1: TQRBand;
DBServico: TQRDBText;
DBQtde: TQRDBText;
DBValor: TQRDBText;
QServicos: TQuery;
LabValTotal: TQRLabel;
DBValTotal: TQRDBText;
ShLinhaGeral: TQRShape;
QRShape1: TQRShape;
QRShape2: TQRShape;
QRShape3: TQRShape;
ImgDown: TQRImage;
QRShape4: TQRShape;
QRShape5: TQRShape;
QRShape6: TQRShape;
QRShape7: TQRShape;
QRShape8: TQRShape;
QRShape9: TQRShape;
QRShape10: TQRShape;
QRShape11: TQRShape;
QRShape12: TQRShape;
QRShape13: TQRShape;
QRShape14: TQRShape;
LabTotal: TQRLabel;
LabTotalValor: TQRLabel;
QRShape15: TQRShape;
QRShape16: TQRShape;
LabForma: TQRLabel;
LabFormaValor: TQRLabel;
LabDescontoValor: TQRLabel;
LabDesconto: TQRLabel;
LabNumParc: TQRLabel;
LabNumParcValor: TQRLabel;
LabEntrada: TQRLabel;
LabEntradaValor: TQRLabel;
LabDataPri: TQRLabel;
LabDataPriValor: TQRLabel;
LabTotalDesc: TQRLabel;
LabTotDescValor: TQRLabel;
LabAssPacinte: TQRLabel;
LabAssDoutor: TQRLabel;
QRShape17: TQRShape;
QRShape18: TQRShape;
QRShape19: TQRShape;
LabNumero: TQRLabel;
LabNumeroValor: TQRLabel;
LabDataUlt: TQRLabel;
LabDataUltValor: TQRLabel;
LabValorParc: TQRLabel;
LabValorParcValor: TQRLabel;
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  RelOrcamento: TRelOrcamento;

implementation

{$R *.dfm}

procedure TRelOrcamento.FormCreate(Sender: TObject);
var
  Ini: TIniFile;
begin
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName) + 'dadosdr.ini');
  ImgTitulo.Canvas.RoundRect(0, 0, 483, 60, 15, 15);
  ImgNome.Canvas.RoundRect(0, 5, 430, 35, 10, 10);

```

```
ImgServicos.Canvas.RoundRect(0,0, 483, 50, 15, 15);
ImgDown.Canvas.RoundRect(0, -20, 483, 15, 15, 15);
LabNomeClinica.Caption := Ini.ReadString('Doutor', 'Clinica', '');
LabEndereco.Caption := Ini.ReadString('Doutor', 'Endereco', '');
Ini.Free;
end;
```

13 UNIT USELSERVICOS

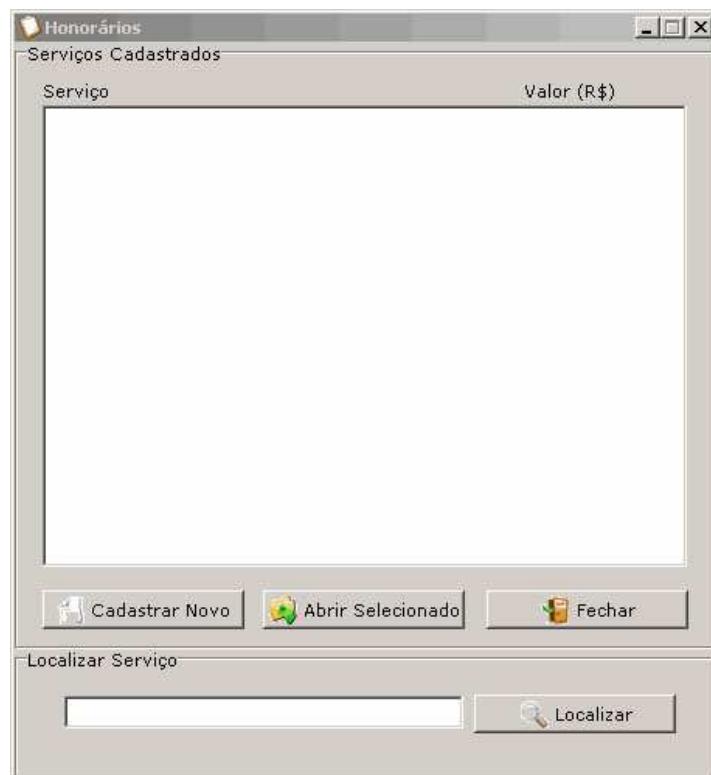


FIGURA 13 - FORMSERVICOS

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
/////////////////////////////
unit uSelServico;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, StdCtrls, Buttons, uColServicos, uServicos;

type
  TFormServicos = class(TForm)
    GBServicos: TGroupBox;
    GBLocalizar: TGroupBox;
    BBLocalizar: TBitBtn;
    SGServicos: TStringGrid;
    BBNovoServico: TBitBtn;
    BBAAlterarSelecionado: TBitBtn;
    BBSair: TBitBtn;
    EdLocalizar: TEdit;
    LabNome: TLabel;
    LabValor: TLabel;
  end;

```

```

procedure SGservicosDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure FormCreate(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BBLocalizarClick(Sender: TObject);
procedure BBNovoServicoClick(Sender: TObject);
procedure SGservicosSelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure BBAalterarSelecionadoClick(Sender: TObject);
procedure SGservicosDblClick(Sender: TObject);
procedure BBSairClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  ServicoAtual: TServicos;
  ColServicos: TColServicos;
  procedure ArrumaGrade;
  procedure LerServicos(Nome: String);
  procedure SelecionarServico;
end;

var
  FormServicos: TFormServicos;
  Linha: Integer;

implementation

uses uCadServico;

{$R *.dfm}

procedure TFormServicos.ArrumaGrade;
begin
  SGservicos.RowCount := 1;
  SGservicos.ColCount := 3;
  SGservicos.ColWidths[0] := 0;
  SGservicos.ColWidths[1] := 310;
  SGservicos.ColWidths[2] := 70;

  SGservicos.Cells[0,0] := '';
  SGservicos.Cells[1,0] := '';
  SGservicos.Cells[2,0] := '';

end;

procedure TFormServicos.LerServicos(Nome: String);
var
  a: Integer;
begin
  ArrumaGrade;
  a := 0;
  ColServicos.Seleciona(Nome);
  ColServicos.Primeiro;

  while not ColServicos.Final do
  begin
    ColServicos.Coloa;
    SGservicos.Cells[0, a] := IntToStr(ColServicos.CCodigo);
    SGservicos.Cells[1, a] := ColServicos.CNome;
    SGservicos.Cells[2, a] := FloatToStr(ColServicos.CValor);
    Inc(a);
    SGservicos.RowCount := a+1;
    ColServicos.Proximo;
  end;
  SGservicos.RowCount := SGservicos.RowCount - 1;
end;

procedure TFormServicos.SGservicosDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);

```

```

begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormServicos.FormCreate(Sender: TObject);
begin
  ColServicos := TColServicos.Create(Self);
  ServicoAtual := TServicos.Create(Self);
end;

procedure TFormServicos.FormShow(Sender: TObject);
begin
  LerServicos('');
end;

procedure TFormServicos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FormServicos.FreeOnRelease;
end;

procedure TFormServicos.BBLocalizarClick(Sender: TObject);
begin
  LerServicos(EdLocalizar.Text);
end;

procedure TFormServicos.SelecionarServico;
begin
  ServicoAtual.PCodigo := StrToInt(SGServicos.Cells[0, Linha]);
  ServicoAtual.PNome := SGServicos.Cells[1, Linha];
  ServicoAtual.PValor := StrToFloat(SGServicos.Cells[2, Linha]);
end;

procedure TFormServicos.BBNovoServiçoClick(Sender: TObject);
begin
  if not assigned(FormCadServiço) then
    FormCadServiço := TFormCadServiço.Create(nil);
  FormCadServiço.Servico.PCodigo := 0;
  FormCadServiço.ShowModal;
  FormCadServiço := nil;
end;

procedure TFormServicos.SGServicosSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if (SGServicos.Cells[0, ARow] <> '') Then
    Linha := ARow
  else Linha := -1;
end;

procedure TFormServicos.BBAlterarSelecionadoClick(Sender: TObject);
begin
  if SGServicos.Cells[0,0] = '' Then
    begin
      MessageDlg('Não há serviço selecionado!', mtInformation, [mbOk], 0);
    end
  else
    begin
      if not assigned(FormCadServiço) then
        FormCadServiço := TFormCadServiço.Create(nil);
      SeleccionarServiço;
      FormCadServiço.Servico := ServicoAtual;
      FormCadServiço.ColocaDados;
      FormCadServiço.EdServiço.Enabled := False;
      FormCadServiço.ShowModal;
      FormCadServiço := nil;
    end;
end;

```

```
        end;
end;

procedure TFormServicos.SGServicosDblClick(Sender: TObject);
begin
  BBAlterarSeleccionadoClick(Sender);
end;

procedure TFormServicos.BBSairClick(Sender: TObject);
begin
  Close;
end;

end.
```

14 UNIT UCADSERVICO



FIGURA 14 - FORMCADSERVICO

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uCadServico;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, uServicos;

type
  TFormCadServico = class(TForm)
    EdServico: TEdit;
    LabNome: TLabel;
    EdValor: TEdit;
    LabValor: TLabel;
    BBSalvar: TBitBtn;
    BBCancelar: TBitBtn;
    procedure EdValorKeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
    procedure BBSalvarClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure BBCancelarClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Servico: TServicos;
    procedure SalvarServico;
    procedure ColocaDados;
  end;

var
  FormCadServico: TFormCadServico;

implementation

uses uSelServico;

{$R *.dfm}

procedure TFormCadServico.EdValorKeyPress(Sender: TObject; var Key: Char);
var
  a: integer;
  comvir: boolean;
```

```

begin
  comvir := False;
  if not (key in['0'..'9', ',', Chr(8)]) then
    key := #0;
  if key = ',' then
    for a:= 1 to Length(EdValor.Text) do
      if EdValor.Text[a] = ',' Then
        comvir := True;
  if comvir Then
    Key := #0;
end;

procedure TFormCadServico.SalvarServico;
var
  Estado: Integer;
begin
  Estado := -1;
  if EdServiço.Text = '' Then
    MessageDlg('Digite o nome do serviço', mtInformation, [mbOk], 0)
  else
    if Servico.PCodigo = 0 Then
      begin
        Servico.Carregar('serv_nome', EdServiço.Text);
        if Servico.PNome <> '' Then
          MessageDlg('Serviço já cadastrado!', mtError, [mbOk], 0)
        else
          Estado := 0;
      end
    else
      Estado := 1;
    if (estado > -1) then
      begin
        Servico.PNome := EdServiço.Text;
        if EdValor.Text = '' Then
          EdValor.Text := '0,0';
        Servico.PValor := StrToFloat(EdValor.Text);
        Servico.Salvar(Estado);
      end;
  end;
end;

procedure TFormCadServico.FormCreate(Sender: TObject);
begin
  Servico := TServicos.Create(FormCadServico);
end;

procedure TFormCadServico.BBSalvarClick(Sender: TObject);
begin
  SalvarServiço;
  Close;
end;

procedure TFormCadServico.ColocaDados;
begin
  EdServiço.Text := Servico.PNome;
  EdValor.Text := FloatToStr(Servico.PValor);
end;

procedure TFormCadServico.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FormServicos.LerServicos('');
  FreeOnRelease;
end;

procedure TFormCadServico.BBCancelarClick(Sender: TObject);
begin
  Close;
end;

end.

```

15 UNIT UFELICITACOES



FIGURA 15 - FORMFELICITACOES

```
///////////
//  

// PROJETO CLÍNICA ODONTOLÓGICA  

// Tecnologia em Informática  

// Projeto de Conclusão de Curso  

// 3º ano - 2004  

//  

// Desenvolvido por:  

// - José Altair Ribeiro dos Santos  

// - Rafael Winter  

//  

//  

unit uFelicitacoes;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, Buttons, ComCtrls, DB, DBTables;

type
  TFormFelicitacoes = class(TForm)
    SGAniversariantes: TStringGrid;
    LabNome: TLabel;
    LabData: TLabel;
    GBPacientes: TGroupBox;
    LabNomeAtual: TLabel;
    LabDataAtual: TLabel;
    GBAAtual: TGroupBox;
    MemoMsg: TMemo;
    BBVisualisar: TBitBtn;
    BBSalvarTexto: TBitBtn;
    BBFechar: TBitBtn;
    GBLocalizar: TGroupBox;
  end;
```

```

CBNome: TCheckBox;
CBData: TCheckBox;
EdNome: TEdit;
DTData1: TDateTimePicker;
DTData2: TDateTimePicker;
LabE: TLabel;
BBLocalizar: TBitBtn;
QAniversarios: TQuery;
LabMsn: TLabel;
procedure SGAniversariantesDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
procedure CBNomeClick(Sender: TObject);
procedure CBDataClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure BBLocalizarClick(Sender: TObject);
procedure SGAniversariantesSelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
procedure BBFecharClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BBSalvarTextoClick(Sender: TObject);
procedure BBVisualisarClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  procedure ArrumaGrade;
  procedure ColocaDados;
end;

const Mes: array[1..12] of String =
  ('Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho',
  'Julho', 'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro');
var
  FormFelicitacoes: TFormFelicitacoes;
implementation
uses uRelCartao;

{$R *.dfm}

procedure TFormFelicitacoes.SGAniversariantesDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  TStringGrid(Sender).Canvas.Pen.Color := clBlack;
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Top);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Top);
  TStringGrid(Sender).Canvas.MoveTo(Rect.Left, Rect.Bottom);
  TStringGrid(Sender).Canvas.LineTo(Rect.Right, Rect.Bottom);
end;

procedure TFormFelicitacoes.CBNomeClick(Sender: TObject);
begin
  if (CBNome.Checked) Then
    EdNome.Enabled := True
  else
    EdNome.Enabled := False;
end;

procedure TFormFelicitacoes.CBDataClick(Sender: TObject);
begin
  if (CBData.Checked) Then
    begin
      DTData1.Enabled := True;
      DTData2.Enabled := True;
    end
  else
    begin
      DTData1.Enabled := False;
      DTData2.Enabled := False;
    end;
end;

```

```

end;

procedure TFormFelicitacoes.ArrumaGrade;
begin
  SGAniversariantes.ColWidths[0] := 300;
  SGAniversariantes.ColWidths[1] := 65;

  SGAniversariantes.Cells[0, 0] := '';
  SGAniversariantes.Cells[1, 0] := '';

  SGAniversariantes.RowCount := 1;
end;

procedure TFormFelicitacoes.ColocaDados;
var
  a: Integer;
begin
  ArrumaGrade;
  QAniversarios.ExecSQL;
  QAniversarios.Open;
  QAniversarios.First;
  a := 0;
  while not QAniversarios.Eof do
  begin
    SGAniversariantes.RowCount := a+1;
    SGAniversariantes.Cells[0, a] :=
      QAniversarios.FieldByName('nome').AsString;
    SGAniversariantes.Cells[1, a] :=
      Copy(QAniversarios.FieldByName('data').AsString, 1, 5);
    Inc(a);
    QAniversarios.Next;
  end;

  QAniversarios.Close;
end;

procedure TFormFelicitacoes.FormCreate(Sender: TObject);
begin
  MemoMsg.Lines.LoadFromFile(ExtractFilePath(Application.ExeName) +
    'textos\felicita.txt');
  ColocaDados;
end;

procedure TFormFelicitacoes.BBLocalizarClick(Sender: TObject);
var
  nome, data, qSQL: string;
begin
  if not (CBNome.Checked) AND not (CBDData.Checked) Then
    MessageDlg('Não há parâmetros selecionados', mtError, [mbOk], 0)
  else
    begin
      nome := 'pac_nome like ' + QuotedStr('%' + EdNome.Text + '%');
      data := 'date_part('+ QuotedStr('month')+ ', pac_datanasc) BETWEEN ' +
        COPY(DateToStr(DTData1.Date), 4, 2) + ' AND ' +
        COPY(DateToStr(DTData2.Date), 4, 2) + ' AND ' +
        'date_part('+ QuotedStr('day')+ ', pac_datanasc) BETWEEN ' +
        COPY(DateToStr(DTData1.Date), 1, 2) + ' AND ' +
        COPY(DateToStr(DTData2.Date), 1, 2);
      qSQL := 'SELECT pac_nome AS nome, pac_datanasc AS data ' +
        'FROM paciente WHERE ';
      if CBNome.Checked Then
        qSQL := qSQL + nome;
      if CBDData.Checked Then
        begin
          if CBNome.Checked Then
            qSQL := qSQL + ' AND ';
          qSQL := qSQL + data;
        end;
      QAniversarios.SQL.Text := qSQL;
      ColocaDados;
    end;
end;

```

```

end;

procedure TFormFelicitacoes.SGANiversariantesSelectCell(Sender: TObject;
  ACol, ARow: Integer; var CanSelect: Boolean);
begin
  LabNomeAtual.Caption := SGANiversariantes.Cells[0, Arow];
  LabDataAtual.Caption := SGANiversariantes.Cells[1, Arow];
end;

procedure TFormFelicitacoes.BBFecharClick(Sender: TObject);
begin
  Close;
end;

procedure TFormFelicitacoes.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

procedure TFormFelicitacoes.BBSalvarTextoClick(Sender: TObject);
begin
  MemoMsg.Lines.SaveToFile(ExtractFilePath(Application.ExeName)+ 
    'textos\felicita.txt');
  MessageDlg('Mensagem Salva!', mtInformation, [mbOk], 0);
end;

procedure TFormFelicitacoes.BBVisualisarClick(Sender: TObject);
var
  a: integer;
begin
  if (LabNomeAtual.Caption <> '-x-') AND (LabNomeAtual.Caption <> '') AND
    (LabDataAtual.Caption <> '-x-') AND (LabDataAtual.Caption <> '') Then
  begin
    if not assigned(RelCartao) then
      RelCartao := TRelCartao.Create(nil);

    RelCartao.LabNomeAniversariante.Caption := LabNomeAtual.Caption;
    RelCartao.MemoMsn.Lines.Text := MemoMsg.Lines.Text;
    a := StrToInt(COPY(LabDataAtual.Caption, 4, 2));
    RelCartao.LabData.Caption :=
      RelCartao.LabData.Caption + COPY(LabDataAtual.Caption, 1, 2) +
      ' de ' + Mes[a] + ' de ' + IntToStr(CurrentYear);
    a := RelCartao.MemoMsn.Lines.Count div 2;
    RelCartao.MemoMsn.Top := (RelCartao.QRCartao.Height div 2) - (a * 15);
    RelCartao.QRCartao.PReviewModal;
    RelCartao.Close;
  end
  else
    MessageDlg('Não há pessoa selecionada!', mtInformation, [mbOk], 0);
end;
end.

```

16 UNIT URELCARTAO

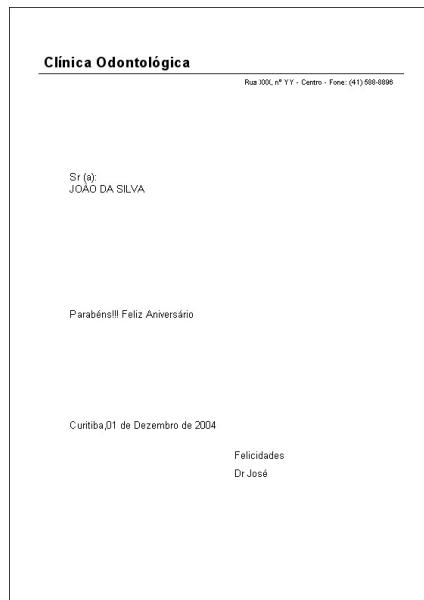


FIGURA 16 - RELCARTAO

```
///////////////////////////////
//  

// PROJETO CLÍNICA ODONTOLÓGICA  

// Tecnologia em Informática  

// Projeto de Conclusão de Curso  

// 3º ano - 2004  

//  

// Desenvolvido por:  

// - José Altair Ribeiro dos Santos  

// - Rafael Winter  

//  

/////////////////////////////
unit uRelCartao;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, QuickRpt, QCtrls, IniFiles, StdCtrls;

type
  TRelCartao = class(TForm)
    QRCartao: TQuickRep;
    TitleBand1: TQRBand;
    ShLinha: TQRShape;
    LabNomeClinica: TQRLabel;
    LabEndereco: TQRLabel;
    MemoMsn: TQRMemo;
    LabNomeAniversariante: TQRLabel;
    LabSr: TQRLabel;
    LabDr: TQRLabel;
    LabData: TQRLabel;
    LabFelicidades: TQRLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
end;

var
  RelCartao: TRelCartao;

implementation

{$R *.dfm}

procedure TRelCartao.FormCreate(Sender: TObject);
var
  Ini: TIniFile;
begin
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName)+ 'dadosdr.ini');
  LabNomeClinica.Caption := Ini.ReadString('Doutor', 'Clinica', '');
  LabEndereco.Caption := Ini.ReadString('Doutor', 'Endereco', '');
  LabDr.Caption := Ini.ReadString('Doutor', 'Nome', '');
  LabData.Caption := Ini.ReadString('Doutor', 'Cidade', '') + ',';
  Ini.Free;
end;

procedure TRelCartao.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Destroy;
end;

end.
```

17 UNIT UATESTADO

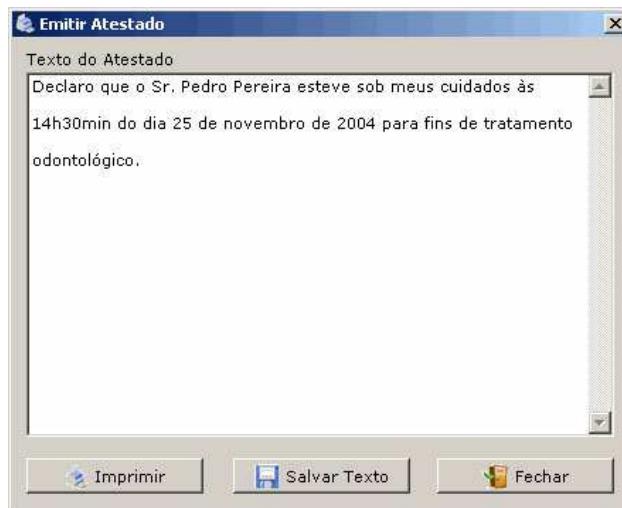


FIGURA 17 - FORMATESTADO

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
/////////////////////////////
unit uAtestado;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;

type
  TFormAtestado = class(TForm)
    MemoTexto: TMemo;
    LabTexto: TLabel;
    BBVisualisar: TBitBtn;
    BBSalvarTexto: TBitBtn;
    BBFechar: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BBVisualisarClick(Sender: TObject);
    procedure BBFecharClick(Sender: TObject);
    procedure BBSalvarTextoClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormAtestado: TFormAtestado;

implementation

uses uRelAtestado;
```

```
{$R *.dfm}

procedure TFormAtestado.FormCreate(Sender: TObject);
begin
  MemoTexto.Lines.LoadFromFile(ExtractFilePath(Application.ExeName) +
    'textos\atestado.txt');
end;

procedure TFormAtestado.BBVisualisarClick(Sender: TObject);
var
  a: Integer;
begin
  RelAtestado := TRelAtestado.Create(nil);

  RelAtestado.Texto.Lines.Text := MemoTexto.Text;
  a := RelAtestado.Texto.Lines.Count div 2;
  RelAtestado.Texto.Top := (RelAtestado.QRAtestado.Height div 2) - (a * 15);
  RelAtestado.QRAtestado.PreviewModal;
  RelAtestado.Close;
end;

procedure TFormAtestado.BBFecharClick(Sender: TObject);
begin
  RelAtestado.Free;
  Close;
end;

procedure TFormAtestado.BBSalvarTextoClick(Sender: TObject);
begin
  MemoTexto.Lines.SaveToFile(ExtractFilePath(Application.ExeName) +
    'textos\atestado.txt');
  MessageDlg('Texto Salvo', mtInformation, [mbOk], 0);
end;

end.
```

18 UNIT URELATESTADO



FIGURA 18 - RELATESTADO

```
///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
/////////////////////////////
unit uRelAtestado;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, QuickRpt, QRCtrls, IniFiles;

type
  TRelAtestado = class(TForm)
    QRAtestado: TQuickRep;
    PageHeaderBand1: TQRBand;
    TitleBand1: TQRBand;
    LabNomeClinica: TQRLabel;
    ShTitulo: TQRShape;
    ShAssinatura: TQRShape;
    LabDr: TQRLabel;
    LabCRO: TQRLabel;
    Texto: TQRMemo;
    LabEndereco: TQRLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  RelAtestado: TRelAtestado;
```

```
implementation

{$R *.dfm}

procedure TRelAtestado.FormCreate(Sender: TObject);
var
  Ini: TIniFile;
begin
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName) +'dadosdr.ini');
  LabNomeClinica.Caption := Ini.ReadString('Doutor', 'Clinica', '');
  LabEndereco.Caption := Ini.ReadString('Doutor', 'Endereco', '');
  LabDr.Caption := Ini.ReadString('Doutor', 'Nome', '');
  LabCRO.Caption := 'CRO' + Ini.ReadString('Doutor', 'CRO', '');
  Ini.Free;
end;

procedure TRelAtestado.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

end.
```

19 UNIT UDADOSCLINICA

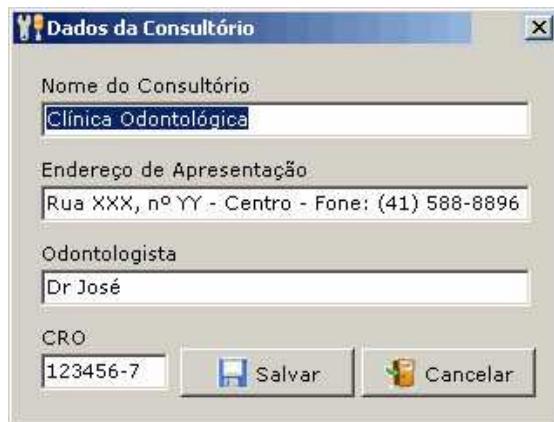


FIGURA 19 – FORMDADOSCLINICA

```
///////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uDadosClinica;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, IniFiles;

type
  TFormDadosClinica = class(TForm)
    LabNome: TLabel;
    EdNome: TEdit;
    LabEndereco: TLabel;
    EdEndereco: TEdit;
    LabNomeDr: TLabel;
    EdDr: TEdit;
    LabCRO: TLabel;
    EdCRO: TEdit;
    BBSalvar: TBitBtn;
    BBCancelar: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BBSalvarClick(Sender: TObject);
    procedure BBCancelarClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormDadosClinica: TFormDadosClinica;
  Ini: TIniFile;
```

```
implementation

{$R *.dfm}

procedure TFormDadosClinica.FormCreate(Sender: TObject);
begin
  Ini := TIniFile.Create(ExtractFilePath(Application.ExeName)+'dadosdr.ini');
  EdNome.Text :=
    Ini.ReadString('Doutor', 'Clinica', 'Consultório Odontológico');
  EdEndereco.Text := Ini.ReadString('Doutor', 'Endereco', '');
  EdDR.Text := Ini.ReadString('Doutor', 'Nome', '');
  EdCRO.Text := Ini.ReadString('Doutor', 'CRO', '');
end;

procedure TFormDadosClinica.BBSalvarClick(Sender: TObject);
begin
  Ini.WriteString('Doutor', 'Clinica', EdNome.Text);
  Ini.WriteString('Doutor', 'Endereco', EdEndereco.Text);
  Ini.WriteString('Doutor', 'Nome', EdDR.Text);
  Ini.ReadString('Doutor', 'CRO', EdCRO.Text);
  Ini.Free;
  Close;
end;

procedure TFormDadosClinica.BBCancelarClick(Sender: TObject);
begin
  Ini.Free;
  Close;
end;

procedure TFormDadosClinica.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

end.
```

20 UNIT UOPCOESUSUARIO



FIGURA 20 - FORMOPCOESUSUARIO

```

////////// PROJETO CLÍNICA ODONTOLÓGICA //////////
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uOpcoesUsuario;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, Grids, uUsuario;

type
  TFormOpcoesUsuario = class(TForm)
    BBCadastrarUsuario: TBitBtn;
    BBExcluirUsuario: TBitBtn;
    BBSair: TBitBtn;
    procedure BBCadastrarUsuarioClick(Sender: TObject);
    procedure BBSairClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure BBExcluirUsuarioClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormOpcoesUsuario: TFormOpcoesUsuario;
  Usuario: TUusuario;
  Login: String;

implementation

uses uCadUsuario;

{$R *.dfm}

procedure TFormOpcoesUsuario.BBCadastrarUsuarioClick(Sender: TObject);
begin
  if not assigned(FormCadUsuario) then
    FormCadUsuario := TFormCadUsuario.Create(Self);
  FormCadUsuario.ShowModal;
  FormCadUsuario := nil;
end;

```

```
procedure TFormOpcoesUsuario.BBSairClick(Sender: TObject);
begin
  Close;
end;

procedure TFormOpcoesUsuario.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  FreeOnRelease;
end;

procedure TFormOpcoesUsuario.BBExcluirUsuarioClick(Sender: TObject);
begin
  Usuario := TUsuario.Create;
  Login := InputBox('Excluir Usuário', 'Digite o usuário a ser excluído', '');
  Usuario.Carregar(Login);
  if Usuario.PLogin = '' Then
    begin
      MessageDlg('Usuário não encontrado!', mtError, [mbOk], 0);
    end
  else
    begin
      if MessageDlg('Tem certeza que deseja excluir usuário?', mtConfirmation,
                     [mbYes, mbNo], 0) = mrYes Then
        begin
          Usuario.Salvar(2);
        end;
    end;
  Usuario.Destroy;
end;

end.
```

21 UNIT UCADUSUARIO

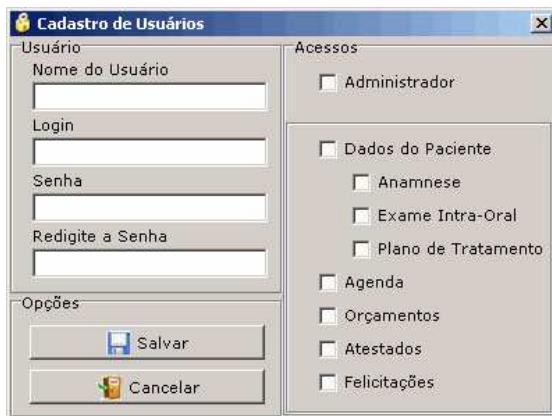


FIGURA 21 - FORMCADUSUARIO

```
///////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
unit uCadUsuario;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, uUsuario;

type
  TFormCadUsuario = class(TForm)
    GBUsuarioAcessos: TGroupBox;
    CBRoot: TCheckBox;
    GBAcessos: TGroupBox;
    CBDadosPaciente: TCheckBox;
    CBAmnese: TCheckBox;
    CBExame: TCheckBox;
    CBAgenda: TCheckBox;
    COrcamento: TCheckBox;
    GBDados: TGroupBox;
    LabLogin: TLabel;
    EdLogin: TEdit;
    EdSenha: TEdit;
    LabSenha: TLabel;
    LabConfirmaSenha: TLabel;
    EdConfirmaSenha: TEdit;
    BBSalvar: TBitBtn;
    BBCancelar: TBitBtn;
    CBAtestados: TCheckBox;
    CBFelicitacoes: TCheckBox;
    LabNome: TLabel;
    EdNome: TEdit;
    CBPlano: TCheckBox;
    GBOpcoes: TGroupBox;
    procedure CBRootMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  end;
```

```

procedure TFormCreate(Sender: TObject);
procedure BBSalvarClick(Sender: TObject);
procedure BBCancelarClick(Sender: TObject);
procedure CBDadosPacienteClick(Sender: TObject);
procedure CBAnamneseClick(Sender: TObject);
private
  { Private declarations }
  procedure SalvarUsuario;
public
  { Public declarations }
  NovoUsuario: TUsuario;
end;

var
  FormCadUsuario: TFormCadUsuario;

implementation

{$R *.dfm}

procedure TFormCadUsuario.CBRootMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if TCheckBox(sender).Checked Then
    begin
      CBDadosPaciente.Checked := True;
      CBAnamnese.Checked := True;
      CBExame.Checked := True;
      CBPlano.Checked := True;
      CBAgenda.Checked := True;
      COrcamento.Checked := True;
      CBAtestados.Checked := True;
      CBFelicitacoes.Checked := True;
      GBAcessos.Enabled := False;
    end
  else
    GBAcessos.Enabled := True;
end;

procedure TFormCadUsuario.FormCreate(Sender: TObject);
begin
  NovoUsuario := TUsuario.Create;
end;

procedure TFormCadUsuario.SalvarUsuario;
begin
  NovoUsuario.PNome := EdNome.Text;
  NovoUsuario.PLogin := EdLogin.Text;
  NovoUsuario.PSenha := EdSenha.Text;
  NovoUsuario.PRoot := CBRoot.Checked;
  NovoUsuario.PDadosPaciente := CBDadosPaciente.Checked;
  NovoUsuario.PAnamnese := CBAnamnese.Checked;
  NovoUsuario.PExame := CBExame.Checked;
  NovoUsuario.PPlano := CBPlano.Checked;
  NovoUsuario.PAgenda := CBAgenda.Checked;
  NovoUsuario.POrcamento := COrcamento.Checked;
  NovoUsuario.PAtestado := CBAtestados.Checked;
  NovoUsuario.PFelicitacao := CBFelicitacoes.Checked;
  NovoUsuario.Salvar(0);
  Close;
end;

procedure TFormCadUsuario.BBSalvarClick(Sender: TObject);
begin
  if (EdSenha.Text <> EdConfirmSenha.Text) Then
    MessageDlg('Senha e confirmação de senha não conferem!', mtError, [mbOk], 0)
  else
    if EdLogin.Text <> '' Then
      begin

```

```
NovoUsuario.Carregar(EdLogin.Text);
if NovoUsuario.PLogin = EdLogin.Text Then
    MessageDlg('Login já cadastrado! Escolha outro Login',
               mtError, [mbOk], 0)
else
    SalvarUsuario;
end
else
    MessageDlg('Campo Login vazio', mtError, [mbOk], 0);
end;

procedure TFormCadUsuario.BBCancelarClick(Sender: TObject);
begin
    Close;
end;

procedure TFormCadUsuario.CBDadosPacienteClick(Sender: TObject);
begin
    if not CBDadosPaciente.Checked Then
        begin
            CBAmnese.Checked := False;
            CBExame.Checked := False;
            CBPlano.Checked := False;
        end;
end;

procedure TFormCadUsuario.CBAnamneseClick(Sender: TObject);
begin
    if TCheckBox(Sender).Checked then
        CBDadosPaciente.Checked := True;
end;
end.
```

22 UNIT UPACIENTE

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit UPaciente;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, UAnamnese, UExame,  
  UPlanoTratamento, uColPlano, dialogs;  
  
type  
  TPaciente = class(TComponent)  
  
private  
  //Fields  
  Fcodigo: integer;  
  Fnome: string;  
  Frg: string;  
  Fcpf: string;  
  Fsexo: String;  
  Fprofissao: string;  
  Fendereco: string;  
  Fbairro: string;  
  Fcidade: string;  
  Fuf: string;  
  Fcep: string;  
  Ffone1: string;  
  Ffone2: string;  
  Fend_com: string;  
  Ffone_com: string;  
  Fdata_nasc: TDate;  
  Flocal_nasc: string;  
  Fini_tratamento: TDate;  
  Findicacao: string;  
  Fobs: string;  
  
public  
  Anamnese: TAnamnese;  
  Exame: TExame;  
  ColPlano: TColPlano;  
  PlanoAtual: TPlano;  
  
  constructor Create(Dono: TComponent);  
  destructor Destroy; override;  
  
  //Property's  
  property Pcodigo: integer read FCodigo write FCodigo;  
  property Pnome: string read FNome write FNome;  
  property PRG: string read FRG write FRG;  
  property PCPF: string read FCPF write FCPF;  
  property Psexo: string read FSexo write FSexo;  
  property Pprofissao: string read Fprofissao write Fprofissao;  
  property Pendereco: string read Fendereco write Fendereco;  
  property Pbairro: string read Fbairro write Fbairro;  
  property Pcidade: string read Fcidade write Fcidade;  
  property Puf: string read Fuf write Fuf;
```

```

property Pcep: string read Fcep write Fcep;
property PFone1: string read Ffone1 write Ffone1;
property PFone2: string read Ffone2 write Ffone2;
property Pend_com: string read Fend_com write Fend_com;
property Pfonecom: string read Ffone_com write Ffone_com;
property Pdata_nasc: Tdate read Fdata_nasc write Fdata_nasc;
property Plocal_nasc: string read Flocal_nasc write Flocal_nasc;
property Pini_tratamento: Tdate read Fini_tratamento write Fini_tratamento;
property Pindicacao: string read Findicacao write Findicacao;
property Pobs: string read Fobs write Fobs;

//Operações
procedure Salvar(estado: integer);
procedure Carregar;
function Verificar(cpf, rg: string; cod: integer): boolean;
end;

implementation

{ TPaciente }

constructor TPaciente.Create(Dono: TComponent);
begin
    inherited Create(Dono);
    Anamnese := TAnamnese.Create;
    Exame := TExame.Create;
    ColPlano := TColPlano.Create(Self);
    PlanoAtual := TPlano.Create(Self);
end;

procedure TPaciente.Salvar(estado: integer);
var
    Salvar: TQuery;
begin
//Estado = 0 >> Inserir
//Estado = 1 >> Atualizar
//Estado = 2 >> Excluir
    try
        Salvar := TQuery.Create(Self);
        Salvar.DatabaseName := 'PGODONTO';

        if (estado = 0) then
            begin
                Salvar.Close;

                //INSERINDO DADOS NA TABELA
                Salvar.SQL.Text :=
                    'INSERT INTO paciente (pac_nome, pac_rg, pac_cpf, pac_sexo, ' +
                    'pac_prof, pac_end, pac_bairro, pac_cidade, pac_uf, pac_cep, ' +
                    'pac_fone1, pac_fone2, pac_endcom, pac_fonecom, pac_datanasc, ' +
                    'pac_localnasc, pac_initratamento, pac_indicacao, pac_obs) VALUES ' +
                    '(:pmnome, :pmrg, :pmcpf, :pmsexo, :pmprofissao, :pmendereco, ' +
                    ':pmbairro, :pmcidade, :pmuf, :pmcep, :pmfone1, :pmfone2, ' +
                    ':pmend_com, :pmfone_com, :pmdata_nasc, :pmlocal_nasc, ' +
                    ':pmindicacao, :pmobs)';

                end
            end
        else
            if (estado = 1) then
                begin

                    //ATUALIZANDO DADOS NA TABELA
                    Salvar.SQL.Text :=
                        'UPDATE paciente SET pac_nome = :pmnome, pac_rg = :pmrg, pac_cpf = :pmcpf, ' +
                        'pac_sexo = :pmsexo, pac_prof = :pmprofissao, pac_end = :pmendereco, ' +
                        'pac_bairro = :pmbairro, pac_cidade = :pmcidade, pac_uf = :pmuf, pac_cep = :pmcep, ' +
                        '+
                        'pac_fone1 = :pmfone1, pac_fone2 = :pmfone2, pac_endcom = :pmend_com, pac_fonecom = :pmfone_com, ' +
                        '+
                        'pac_datanasc = :pmdata_nasc, pac_localnasc = :pmlocal_nasc, pac_initratamento = :pmindicacao, ' +
                        ':pmobs)';

                end;
            end;
    end;
end;

```

```

'pac_indicacao = :pmindicacao, pac_obs = :pmobs WHERE pac_cod = :pmcod';
Salvar.ParamByName('pmcod').AsInteger := PCodigo;
end
else
if (estado = 2) then
begin
  Salvar.SQL.Text :=
    'DELETE FROM paciente WHERE pac_cod = :pmcod';
  Salvar.ParamByName('pmcod').AsInteger := PCodigo;
end;

//ATRIBUI PARÂMETROS A CONSULTA
if (estado < 2) then
begin
  Salvar.ParamByName('pmnome').AsString := FNome;
  Salvar.ParamByName('pmrg').AsString := FRG;
  Salvar.ParamByName('pmcpf').AsString := FCPF;
  Salvar.ParamByName('pmsexo').AsString := FSexo;
  Salvar.ParamByName('pmprofissao').AsString := FProfissao;
  Salvar.ParamByName('pmendereco').AsString := FEndereco;
  Salvar.ParamByName('pmbairro').AsString := FBairro;
  Salvar.ParamByName('pmcidade').AsString := FCidade;
  Salvar.ParamByName('pmuf').AsString := FUF;
  Salvar.ParamByName('pmcep').AsString := Fcep;
  Salvar.ParamByName('pmfone1').AsString := Ffone1;
  Salvar.ParamByName('pmfone2').AsString := Ffone2;
  Salvar.ParamByName('pmend_com').AsString := Fend_com;
  Salvar.ParamByName('pmfone_com').AsString := Ffone_com;
  Salvar.ParamByName('pmdata_nasc').AsDate := Fdata_nasc;
  Salvar.ParamByName('pmlocal_nasc').AsString := Flocal_nasc;
  Salvar.ParamByName('pminitramento').AsDate := Fini_tratamento;
  Salvar.ParamByName('pmindicacao').AsString := Findicacao;
  Salvar.ParamByName('pmobs').AsString := Fobs;
end;

try
  Salvar.ExecSQL;
  if Estado = 0 Then
  begin
    Salvar.Close;
    Salvar.SQL.Text :=
      'SELECT MAX(pac_cod) as qtde FROM paciente';
    Salvar.ExecSQL;
    Salvar.Open;
    FCodigo := Salvar.FieldByName('qtde').AsInteger;
  end;
  if Estado < 2 Then
    MessageDlg('Dados salvos com sucesso', mtInformation, [mbOk], 0)
  else
    MessageDlg('Dados excluídos com sucesso', mtInformation, [mbOk], 0);
except
  MessageDlg('Não foi possível salvar! Verifique a conexão ' +
             'com o Banco de Dados', mtError, [mbOk], 0);
end;

Finally
  FreeAndNil(Salvar);
End;

end;

procedure TPaciente.Carregar;
var
  Ler: TQuery;
begin
  Ler := TQuery.Create(Application);

  try
    Ler.DatabaseName := 'PGODONTO';
    Ler.SQL.Text := 'SELECT pac_nome, pac_rg, pac_cpf, pac_sexo, pac_prof, ' +

```

```

'pac_end, pac_bairro, pac_cidade, pac_uf, pac_cep, pac_fone1, pac_fone2, ' +
'pac_datanasc, pac_localnasc, pac_iniltratamento, pac_indicacao, pac_obs, ' +
'pac_endcom, pac_fonecom FROM paciente WHERE pac_cod = :pmcod ';
Ler.ParamByName('pmcod').AsInteger := PCodigo;
Ler.Open;
FNome := Ler.fieldbyname('pac_nome').AsString;
FRG := Ler.fieldbyname('pac_rg').AsString;
FCPF := Ler.fieldbyname('pac_cpf').AsString;
FSexo := Ler.fieldbyname('pac_sexo').AsString;
FProfissao := Ler.fieldbyname('pac_prof').AsString;
FEndereco := Ler.fieldbyname('pac_end').AsString;
FBairro := Ler.fieldbyname('pac_bairro').AsString;
FCidade := Ler.fieldbyname('pac_cidade').AsString;
FUF := Ler.fieldbyname('pac_uf').AsString;
FCEP := Ler.fieldbyname('pac_cep').AsString;
FFone1 := Ler.fieldbyname('pac_fone1').AsString;
FFone2 := Ler.fieldbyname('pac_fone2').AsString;
Fdata_nasc := Ler.fieldbyname('pac_datanasc').AsDateTime;
FLocal_nasc := Ler.fieldbyname('pac_localnasc').AsString;
FIni_tratamento := Ler.fieldbyname('pac_iniltratamento').AsDateTime;
FIndicacao := Ler.fieldbyname('pac_indicacao').AsString;
FObs := Ler.fieldbyname('pac_obs').AsString;
FEnd_Com := Ler.fieldbyname('pac_endcom').AsString;
FFone_Com := Ler.fieldbyname('pac_fonecom').AsString;
finally
  FreeAndNil(Ler);
end;
end;

function TPaciente.Verificar(cpf, rg: string; cod: integer): boolean;
var
  Consulta: TQuery;
begin
  begin
    try
      Consulta := TQuery.Create(Application);
      Consulta.DatabaseName := 'PGODONTO';
      Consulta.Close;
      Consulta.SQL.Text :=
        'SELECT pac_cod, pac_rg, pac_cpf FROM paciente WHERE pac_rg = :prg or ' +
        'pac_cpf = :pcpf';
      Consulta.ParamByName('prg').AsString := rg;
      Consulta.ParamByName('pcpf').AsString := cpf;
      Consulta.ExecSQL;
      Consulta.Open;

      if (((Consulta.FieldByName('pac_rg').AsString = '') AND
          (Consulta.FieldByName('pac_cpf').AsString = '')) OR
          (cod = Consulta.FieldByName('pac_cod').AsInteger)) then
        Result := false
      else
        Result := true;
    Finally
      FreeAndNil(Consulta);
    End;
  end;
  destructor TPaciente.Destroy;
  begin
    inherited;
  end;
end.

```

23 UNIT UCOLPACIENTE

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uColPaciente;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs;  
  
type  
  TColPaciente = class(TComponent)  
  private  
    OColecao : TQuery;  
  public  
    CCodigo: Integer;  
    CNome: String;  
    CCPF: String;  
    CRG: String;  
    CDataNasc: String;  
    CFoneRes: String;  
    CFoneCom: String;  
  
    constructor Create(Dono: TComponent);  
    destructor Destroy; override;  
  
    procedure Seleciona;  
    procedure SelecionaAgenda;  
    procedure Primeiro;  
    procedure Anterior;  
    procedure Proximo;  
    procedure Ultimo;  
    procedure Coloca;  
    procedure ColocaAgenda;  
    procedure Procura(campo, valor: String);  
    function Comeco: Boolean;  
    function Final: Boolean;  
  end;  
  
implementation  
  
{ TcolPaciente }  
  
constructor TColPaciente.Create(Dono: TComponent);  
begin  
  inherited Create(Dono);  
  OColecao := TQuery.Create(Self);  
  OColecao.DatabaseName := 'PGODONTO';  
end;  
  
destructor TColPaciente.Destroy;  
begin  
  inherited;  
  OColecao.Free;  
end;  
  
procedure TColPaciente.Coloca;
```

```

begin
  CCodigo := OColecao.FieldByName('pac_cod').AsInteger;
  CNome := OColecao.FieldByName('pac_nome').AsString;
  CRG := OColecao.FieldByName('pac_rg').AsString;
  CCPF := OColecao.FieldByName('pac_cpf').AsString;
  CDataNasc := OColecao.FieldByName('pac_datanasc').AsString;
end;

procedure TColPaciente.Primeiro;
begin
  OColecao.First;
end;

procedure TColPaciente.Seleciona;
begin
  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT pac_cod, pac_nome, pac_rg, pac_cpf, pac_datanasc FROM paciente ' +
    'ORDER BY pac_nome';
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColPaciente.Ultimo;
begin
  OColecao.Last;
end;

procedure TColPaciente.Anterior;
begin
  OColecao.Prior;
end;

procedure TColPaciente.Proximo;
begin
  OColecao.Next;
end;

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

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

procedure TColPaciente.Procura(campo, valor: String);
var
  data: string;
begin
  OColecao.Close;
  if campo = 'pac_datanasc' Then
    begin
      data := copy(valor, 7, 4) + copy(valor, 4, 2) + copy(valor, 1, 2);
      OColecao.SQL.Text :=
        'SELECT pac_cod, pac_nome, pac_rg, pac_cpf, pac_datanasc FROM paciente WHERE ' +
        campo + ' = :pmvalor ORDER BY pac_nome';
      OColecao.ParamByName('pmvalor').AsString := data;
    end
  Else
    begin
      OColecao.SQL.Text :=
        'SELECT pac_cod, pac_nome, pac_rg, pac_cpf, pac_datanasc FROM ' +
        'paciente WHERE ' + campo + ' like :pmvalor ORDER BY pac_nome';
      OColecao.ParamByName('pmvalor').AsString := '%' + valor + '%';
    end;
  OColecao.ExecSQL;
  OColecao.Open;
end;

```

```
procedure TColPaciente.SelecionaAgenda;
begin
  Ocolecao.Close;
  Ocolecao.SQL.Text :=
    'SELECT pac_cod, pac_nome, pac_fone1, pac_fonecom FROM paciente ' +
    'ORDER BY pac_nome';
  Ocolecao.ExecSQL;
  Ocolecao.Open;
end;

procedure TColPaciente.ColocaAgenda;
begin
  CCodigo := OColecao.FieldByName('pac_cod').AsInteger;
  CNome := OColecao.FieldByName('pac_nome').AsString;
  CFoneRes := OColecao.FieldByName('pac_fone1').AsString;
  CFoneCom := OColecao.FieldByName('pac_fonecom').AsString;
end;

end.
```

24 UNIT UANAMNESE

```

// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
unit uExame;

interface

uses
  Controls, DBTables, SysUtils, Forms, Dialogs;

type
  TExame = class(TObject)

private
  //Fields
  FExame: String;
  FLingua: String;
  FAssoalho: String;
  FLabios: String;
  FPalato: String;
  FExObs: String;

public
  //Property's
  property PExame: String read FExame write FExame;
  property PLingua: String read FLingua write FLingua;
  property PAasoalho: String read FAssoalho write FAssoalho;
  property PLabios: String read FLabios write FLabios;
  property PPalato: String read FPalato write FPalato;
  property PExObs: String read FExObs write FExObs;

  //Operações
  procedure Buscar(Codigo: Integer);
  procedure Salvar(Codigo: Integer);

end;

implementation

{ TExame }

procedure TExame.Buscar(Codigo: Integer);
var
  Busca: TQuery;
begin
  try
    Busca := TQuery.Create(Application);
    Busca.DatabaseName := 'PGODONTO';
    Busca.SQL.Text :=
      'SELECT pac_exame, pac_lingua, pac_labios, pac_assoalho, pac_palato, ' +
      'pac_exobs FROM paciente WHERE pac_cod = :pmcod';
    Busca.ParamByName('pmcod').AsInteger := Codigo;
    Busca.ExecSQL;
    Busca.Open;
    FExame := Busca.FieldByName('pac_exame').AsString;
    FLingua := Busca.FieldByName('pac_lingua').AsString;
  except
  end;
end;

```

```

FLabios := Busca.FieldByName('pac_labios').AsString;
FPalato := Busca.FieldByName('pac_palato').AsString;
FAssoalho := Busca.FieldByName('pac_assoalho').AsString;
FExObs := Busca.FieldByName('pac_exobs').AsString;
Busca.Close;
Finally
  FreeAndNil(Busca);
end;
end;

procedure TExame.Salvar(Codigo: Integer);
var
  Salva: TQuery;
begin
  try
    Salva := TQuery.Create(Application);
    Salva.DatabaseName := 'PGODONTO';
    Salva.SQL.Text :=
      'UPDATE paciente SET pac_exame = :pmexame, pac_lingua = :pmlingua, ' +
      'pac_labios = :pmlabios, pac_assoalho = :pmassoalho, pac_palato = :pmpalato, ' +
      'pac_exobs = :pmexobs WHERE pac_cod = :pmcod';
    Salva.ParamByName('pmcod').AsInteger := Codigo;
    Salva.ParamByName('pmexame').AsString := FExame;
    Salva.ParamByName('pmlabios').AsString := FLabios;
    Salva.ParamByName('pmlingua').AsString := FLingua;
    Salva.ParamByName('pmpalato').AsString := FPalato;
    Salva.ParamByName('pmassoalho').AsString := FAssoalho;
    Salva.ParamByName('pmexobs').AsString := FExObs;
    try
      Salva.ExecSQL;
      MessageDlg('Dados do exame Intra-Oral salvo com sucesso',
                 mtInformation, [mbOk], 0);
    Except
      MessageDlg('Não foi possível cadastrar usuário! Verifique a conexão ' +
                 'com o Banco de Dados', mtError, [mbOk], 0);
    End;
  Finally
    FreeAndNil(Salva);
  end;
end;
end.

```

25 UNIT UEXAME

```

// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
//   - José Altair Ribeiro dos Santos
//   - Rafael Winter
//
unit uExame;

interface

uses
  Controls, DBTables, SysUtils, Forms, Dialogs;

type
  TExame = class(TObject)

private
  //Fields
  FExame: String;
  FLingua: String;
  FAssoalho: String;
  FLabios: String;
  FPalato: String;
  FExObs: String;

public
  //Property's
  property PExame: String read FExame write FExame;
  property PLingua: String read FLingua write FLingua;
  property PAasoalho: String read FAssoalho write FAssoalho;
  property PLabios: String read FLabios write FLabios;
  property PPalato: String read FPalato write FPalato;
  property PExObs: String read FExObs write FExObs;

  //Operações
  procedure Buscar(Codigo: Integer);
  procedure Salvar(Codigo: Integer);

end;

implementation

{ TExame }

procedure TExame.Buscar(Codigo: Integer);
var
  Busca: TQuery;
begin
  try
    Busca := TQuery.Create(Application);
    Busca.DatabaseName := 'PGODONTO';
    Busca.SQL.Text :=
      'SELECT pac_exame, pac_lingua, pac_labios, pac_assoalho, pac_palato, ' +
      'pac_exobs FROM paciente WHERE pac_cod = :pmcod';
    Busca.ParamByName('pmcod').AsInteger := Codigo;
    Busca.ExecSQL;
    Busca.Open;
    FExame := Busca.FieldByName('pac_exame').AsString;
    FLingua := Busca.FieldByName('pac_lingua').AsString;
  except
  end;
end;

```

```

FLabios := Busca.FieldByName('pac_labios').AsString;
FPalato := Busca.FieldByName('pac_palato').AsString;
FAssoalho := Busca.FieldByName('pac_assoalho').AsString;
FExObs := Busca.FieldByName('pac_exobs').AsString;
Busca.Close;
Finally
  FreeAndNil(Busca);
end;
end;

procedure TExame.Salvar(Codigo: Integer);
var
  Salva: TQuery;
begin
  try
    Salva := TQuery.Create(Application);
    Salva.DatabaseName := 'PGODONTO';
    Salva.SQL.Text :=
      'UPDATE paciente SET pac_exame = :pmexame, pac_lingua = :pmlingua, ' +
      'pac_labios = :pmlabios, pac_assoalho = :pmassoalho, pac_palato = :pppalato, ' +
      'pac_exobs = :pmexobs WHERE pac_cod = :pmcod';
    Salva.ParamByName('pmcod').AsInteger := Codigo;
    Salva.ParamByName('pmexame').AsString := FExame;
    Salva.ParamByName('pmlabios').AsString := FLabios;
    Salva.ParamByName('pmlingua').AsString := FLingua;
    Salva.ParamByName('pppalato').AsString := FPalato;
    Salva.ParamByName('pmassoalho').AsString := FAssoalho;
    Salva.ParamByName('pmexobs').AsString := FExObs;

    try
      Salva.ExecSQL;
      MessageDlg('Dados do exame Intra-Oral salvo com sucesso',
                 mtInformation, [mbOk], 0);
    Except
      MessageDlg('Não foi possível cadastrar usuário! Verifique a conexão ' +
                 'com o Banco de Dados', mtError, [mbOk], 0);
    End;
  Finally
    FreeAndNil(Salva);
  end;
end;
end.

```

26 UNIT UPLANOTRATAMENTO

```

//////////  

//  

// PROJETO CLÍNICA ODONTOLÓGICA  

// Tecnologia em Informática  

// Projeto de Conclusão de Curso  

// 3º ano - 2004  

//  

// Desenvolvido por:  

// - José Altair Ribeiro dos Santos  

// - Rafael Winter  

//  

/////////  

unit uPlanoTratamento;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, uColRadio, uColOdonto, Dialogs,
  uOdonto, uRadio;

type
  TPlano = class(TComponent)

private
  //Fields
  FCodigo: Integer;
  FCodPac: Integer;
  FData: TDate;
  FDescricao: String;
  FObservacao: String;

public
  ColOdonto: TColOdonto;
  ColRadio: TColRadio;
  OdontoAtual: TOdontograma;
  RadioAtual: TRadio;
  constructor Create(Dono: TComponent);
  destructor Destroy; override;

  //Property's
  property PCodigo: Integer read FCodigo write FCodigo;
  property PCodPac: Integer read FCodPac write FCodPac;
  property PData: TDate read FData write FData;
  property PDescricao: String read FDescricao write FDescricao;
  property PObservacao: String read FObservacao write FObservacao;

  //Operações
  procedure Carregar(Codigo: Integer);
  procedure Salvar(Estado: Integer);

end;

implementation

{ TPlano }

constructor TPlano.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  ColOdonto := TColOdonto.Create(Self);
  ColRadio := TColRadio.Create(Self);
  OdontoAtual := TOdontograma.Create(Self);
  RadioAtual := TRadio.Create(Self);
end;

procedure TPlano.Carregar(Codigo: Integer);

```

```

var
  Ler: TQuery;
begin
try
  Ler := TQuery.Create(Application);
  Ler.DatabaseName := 'PGODONTO';
  Ler.SQL.Text :=
    'SELECT plano_data, plano_desc, plano_obs FROM plano WHERE ' +
    'plano_cod = :pmcod';
  Ler.ParamByName('pmcod').AsInteger := FCodigo;
  Ler.ExecSQL;

  Ler.Open;

  FCodigo := FCodigo;
  FData := Ler.FieldByName('plano_data').AsDateTime;
  FDescricao := Ler.FieldByName('plano_desc').AsString;
  FObservacao := Ler.FieldByName('plano_obs').AsString;

  Ler.Close;

Finally
  FreeAndNil(Ler);
end;
end;

procedure TPlano.Salvar(Estado: Integer);
var
  Salvar: TQuery;
begin
//Estado = 0 => Inserir
//Estado = 1 => Atualizar
//Estado = 2 => Excluir

try
  Salvar := TQuery.Create(Self);
  Salvar.DatabaseName := 'PGODONTO';

  if (Estado = 0) Then
    begin
      Salvar.SQL.Text :=
        'INSERT INTO plano (pac_cod, plano_data, plano_desc, plano_obs) ' +
        'VALUES (:pmcodpaciente, :pmdata, :pmdescricao, :pmobs)';
      Salvar.ParamByName('pmcodpaciente').AsInteger := FCodPac;
    end
  Else
    if (Estado = 1) Then
      begin
        Salvar.SQL.Text :=
          'UPDATE plano SET plano_data = :pmdata, ' +
          'plano_desc = :pmdescricao, plano_obs = :pmobs ' +
          'WHERE plano_cod = :pmcodplano';
        Salvar.ParamByName('pmcodplano').AsInteger := FCodigo;
      end
  Else
    begin
      Salvar.SQL.Text :=
        'DELETE FROM plano WHERE plano_cod = :pmcodplano';
      Salvar.ParamByName('pmcodplano').AsInteger := FCodigo;
    end;
  if (Estado < 2) Then
    begin
      Salvar.ParamByName('pmdescricao').AsString := FDescricao;
      Salvar.ParamByName('pmdata').AsDateTime := FData;
      Salvar.ParamByName('pmobs').AsString := FObservacao;
    end;
end;

```

```
end;

try
  Salvar.ExecSQL;
  if (Estado = 0) Then
    begin
      Salvar.Close;
      Salvar.SQL.Text :=
        'SELECT MAX(plano_cod) as qtde FROM plano';
      Salvar.ExecSQL;
      Salvar.Open;
      FCodigo := Salvar.FieldByName('qtde').AsInteger;
    end;
  if Estado = 1 Then
    MessageDlg('Plano de Tratamento salvo com sucesso',
               mtInformation, [mbOk], 0)
  else if Estado = 2 Then
    MessageDlg('Plano de Tratamento excluído com sucesso',
               mtInformation, [mbOk], 0);
  Except
    MessageDlg('Não foi possível salvar! Verifique a conexão ' +
               'com o Banco de Dados', mtError, [mbOk], 0);
  End;

Finally
  FreeAndNil(Salvar);
end;

end;

destructor TPlano.Destroy;
begin
  inherited;
end;

end.
```

27 UNIT UCOLPLANO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uColPlano;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, uPlanoTratamento;  
  
type  
  TColPlano = class(TComponent)  
  private  
    Ocolecao : TQuery;  
  
  public  
  
    CCodPlano: Integer;  
    CDataPlano: String;  
    CDescPlano: String;  
    CCodPaciente: Integer;  
  
    constructor Create(Dono: TComponent);  
    destructor Destroy; override;  
  
    procedure Seleciona;  
    procedure Primeiro;  
    procedure Anterior;  
    procedure Proximo;  
    procedure Ultimo;  
    procedure Coloca;  
    function Comeco: Boolean;  
    function Final: Boolean;  
  
  end;  
  
implementation  
  
{ TColPlano }  
  
constructor TColPlano.Create(Dono: TComponent);  
begin  
  inherited Create(Dono);  
  Ocolecao := TQuery.Create(Self);  
  Ocolecao.DatabaseName := 'PGODONTO';  
end;  
  
destructor TColPlano.Destroy;  
begin  
  inherited;  
  Ocolecao.Free;  
  
end;  
  
procedure TColPlano.Coloca;  
begin
```

```

CCodPlano := OColecao.FieldByName('plano_cod').AsInteger;
CDataPlano:= OColecao.FieldByName('plano_data').AsString;
CDescPlano:= OColecao.FieldByName('plano_desc').AsString;

end;

procedure TColPlano.Primeiro;
begin
  OColecao.First;
end;

procedure TColPlano.Seleciona;
begin

  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT plano_cod, plano_data, plano_desc FROM plano ' +
    'WHERE pac_cod = :pmcod ORDER BY plano_data DESC';
  OColecao.ParamByName('pmcod').AsInteger := CCodPaciente;
  OColecao.ExecSQL;
  OColecao.Open;

end;

procedure TColPlano.Ultimo;
begin
  OColecao.Last;
end;

procedure TColPlano.Anterior;
begin
  OColecao.Prior;
end;

procedure TColPlano.Proximo;
begin
  OColecao.Next;
end;

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

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

end.

```

28 UNIT UODONTO

```

////////// unit uOdonto;
//
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uOdonto;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, uColOperacao, Dialogs;

type
  TOdontograma = class(TComponent)

private
  //Fields
  FCodigo: Integer;
  FCodPlano: Integer;
  FDescricao: String;
  FData: TDate;
  FObservacao: String;

public
  Operacoes: TColOperacao;

  //Property's
  property PCodigo: integer read FCodigo write FCodigo;
  property PCodPlano: integer read FCodPlano write FCodPlano;
  property PDescricao: String read FDescricao write FDescricao;
  property PData: TDate read FData write FData;
  property PObservacao: String read FObservacao write FObservacao;

  //Operações
  constructor Create(Dono: TComponent);
  destructor Destroy; override;
  procedure Carregar(Codigo: Integer);
  procedure Salvar(Estado: Integer);

end;

implementation

{ TOdontograma }

procedure TOdontograma.Carregar(Codigo: Integer);
var
  Ler: TQuery;
begin
  try
    Ler := TQuery.Create(Self);
    Ler.DatabaseName := 'PGODONTO';
    Ler.SQL.Text :=
      'SELECT odonto_data, odonto_desc, odonto_obs FROM odontograma ' +
      'WHERE odonto_cod = :pmcod';
    Ler.ParamByName('pmcod').AsInteger := Codigo;
    Ler.ExecSQL;
  except
  end;
end;

```

```

Ler.Open;

FData := Ler.FieldByName('odonto_data').AsDateTime;
FDescricao := Ler.FieldByName('odonto_desc').AsString;
FObservacao := Ler.FieldByName('odonto_obs').AsString;

Ler.Close;

Finally
  FreeAndNil(Ler);
end;

end;

constructor TOdontograma.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Operacoes := TColOperacao.Create(Self);
end;

destructor TOdontograma.Destroy;
begin
  inherited;
end;

procedure TOdontograma.Salvar(Estado: Integer);
var
  Salvar: TQuery;
begin
  //Estado = 0 => Inserir
  //Estado = 1 => Atualizar
  //Estado = 2 => Excluir

  try
    Salvar := TQuery.Create(Application);
    Salvar.DatabaseName := 'PGODONTO';

    if (Estado = 0) Then
      begin
        Salvar.Close;
        Salvar.SQL.Text :=
          'INSERT INTO odontograma (plano_cod, odonto_data, odonto_desc, ' +
          'odonto_obs) VALUES (:pmcodplano, :pmdata, :pmdesc, :pmobs)';
        Salvar.ParamByName('pmcodplano').AsInteger := FCodPlano;
      end
    Else
      if (Estado = 1) Then
        begin
          Salvar.SQL.Text :=
            'UPDATE odontograma SET odonto_data = :pmdata, ' +
            'odonto_desc = :pmdesc, odonto_obs = :pmobs WHERE ' +
            'odonto_cod = :pmcod';
          Salvar.ParamByName('pmcod').AsInteger := FCodigo;
        end
      Else
        begin
          Salvar.SQL.Text :=
            'DELETE FROM odontograma WHERE odonto_cod = :pmcod';
          Salvar.ParamByName('pmcod').AsInteger := FCodigo;
        end;
    end;
    if (Estado < 2) Then
      begin
        Salvar.ParamByName('pmdata').AsDateTime := FData;
        Salvar.ParamByName('pmdesc').AsString := FDescricao;
        Salvar.ParamByName('pmobs').AsString := FObservacao;
      end;
  end;
end;

```

```
try
  Salvar.ExecSQL;
  if Estado = 0 Then
    begin
      Salvar.Close;
      Salvar.SQL.Text :=
        'SELECT MAX(odonto_cod) as qtde FROM odontograma';
      Salvar.ExecSQL;
      Salvar.Open;
      FCodigo := Salvar.FieldByName('qtde').AsInteger;
    end;
  if Estado = 1 Then
    MessageDlg('Odontograma salvo com sucesso', mtInformation, [mbOk], 0)
  else if Estado = 2 Then
    MessageDlg('Odontograma excluído com sucesso', mtInformation, [mbOk], 0);
  Except
    MessageDlg('Não foi possível salvar! Verifique a conexão ' +
      'com o Banco de Dados', mtError, [mbOk], 0);
  End;

  Finally
    FreeAndNil(Salvar);
  end;
end;
end.
```

29 UNIT UCOLODONTO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uColodonto;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs, uOdonto;  
  
type  
  TColodonto = class(TComponent)  
  private  
    Ocolecao : TQuery;  
  
  public  
  
    CCodOdonto: Integer;  
    CDataOdonto: String;  
    CDescOdonto: String;  
    CCodPlano: Integer;  
  
    constructor Create(Dono: TComponent);  
    destructor Destroy; override;  
  
    procedure Seleciona;  
    procedure Primeiro;  
    procedure Anterior;  
    procedure Proximo;  
    procedure Ultimo;  
    procedure Coloca;  
    function Comeco: Boolean;  
    function Final: Boolean;  
  end;  
  
implementation  
  
{ TColodonto }  
  
constructor TColodonto.Create(Dono: TComponent);  
begin  
  inherited Create(Dono);  
  Ocolecao := TQuery.Create(Self);  
  Ocolecao.DatabaseName := 'PGODONTO';  
  
end;  
  
destructor TColodonto.Destroy;  
begin  
  inherited;  
  Ocolecao.Free;  
end;  
  
procedure TColodonto.Coloca;  
begin  
  CCodOdonto := OColecao.FieldByName('odonto_cod').AsInteger;  
end;
```

```

CDataOdonto:= OColecao.FieldByName('odonto_data').AsString;
CDescOdonto:= OColecao.FieldByName('odonto_desc').AsString;

end;

procedure TColOdonto.Primeiro;
begin
  OColecao.First;
end;

procedure TColOdonto.Seleciona;
begin
  OColecao.Close;
  OColecao.SQL.Text := 
    'SELECT odonto_cod, odonto_data, odonto_desc FROM odontograma ' +
    'WHERE plano_cod = :pmcod ORDER BY odonto_data DESC';
  OColecao.ParamByName('pmcod').AsInteger := CCodPlano;
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColOdonto.Ultimo;
begin
  OColecao.Last;
end;

procedure TColOdonto.Anterior;
begin
  OColecao.Prior;
end;

procedure TColOdonto.Proximo;
begin
  OColecao.Next;
end;

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

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

end.

```

30 UNIT UOPERACAO

```

////////// /////////////////////////////////////////////////////
// // PROJETO CLÍNICA ODONTOLÓGICA // //
// Tecnologia em Informática // //
// Projeto de Conclusão de Curso // //
// 3º ano - 2004 // //
// // Desenvolvido por: // //
// - José Altair Ribeiro dos Santos // //
// - Rafael Winter // //
// ///////////////////////////////////////////////////
unit uOperacao;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms;

type
  TOperacao = class(TComponent)

private
  //Fields
  FCodigo: Integer;
  FCodOdonto: Integer;
  FOperacao: Integer;
  FDenteNome: String;
  FDescricao: String;
  FPosX: Integer;
  FPosY: Integer;
  FCor: Integer;

public
  Situacao: Integer;

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

//Property's
property PCodigo: Integer read FCodigo write FCodigo;
property PCodOdonto: Integer read FCodOdonto write FCodOdonto;
property POperacao: Integer read FOperacao write FOperacao;
property PDenteNome: String read FDenteNome write FDenteNome;
property PPosX: Integer read FPosX write FPosX;
property PPosY: Integer read FPosY write FPosY;
property PDescricao: String read FDescricao write FDescricao;
property PCor: Integer read FCor write FCor;

//Operações
procedure Carregar(Codigo: Integer);
procedure Salvar(Estado: Integer);

function LerPosX(Valor: String): Integer;
function LerPosY(Valor: String): Integer;
function LerDescricao(Valor: String): String;

end;

implementation

{ TOperacao }

procedure TOperacao.Carregar(Codigo: Integer);
var
  Ler: TQuery;

```

```

    Descr: String;
begin
try
  Ler := TQuery.Create(Self);
  Ler.DatabaseName := 'PGODONTO';
  Ler.SQL.Text := 'SELECT dente_operacao, dente_desc, dente_cor FROM dente WHERE ' +
  'dente_cod = :pmcod';
  Ler.ParamByName('pmcod').AsInteger := Codigo;
  FOperacao := Ler.FieldByName('dente_operacao').AsInteger;
  FCor := Ler.FieldByName('dente_cor').AsInteger;
  Descr := Ler.FieldByName('dente_desc').AsString;
  FPosX := LerPosX(Descr);
  FPosY := LerPosY(Descr);
  FDescricao := LerDescricao(Descr);
Finally
  FreeAndNil(Ler);
End;

end;

constructor TOperacao.Create(DoNo: TComponent);
begin
  inherited Create(DoNo);
end;

destructor TOperacao.Destroy;
begin
  inherited;
end;

function TOperacao.LerDescricao(Valor: String): String;
var
  a: Integer;
begin
  if (Valor[1] <> '[') Then
    Result := Valor
  else
    begin
      a := 1;
      while valor[a] <> ']' do
        Inc(a);
      Inc(a);
      Result := COPY(Valor, a, Length(Valor) -1);
    end;
end;

function TOperacao.LerPosX(Valor: String): Integer;
var
  a: integer;
  x: String;
begin
  if Valor[1] <> '[' Then
    Result := -1
  Else
    begin
      a := 2;
      x := '';
      while Valor[a] <> ',' do
        begin
          x := x + Valor[a];
          Inc(a);
        end;
      Result := StrToInt(x);
    end;
end;

function TOperacao.LerPosY(Valor: String): Integer;
var
  a: integer;
  y: String;

```

```

begin
  if Valor[1] <> '[' Then
    Result := -1
  Else
    begin
      a := 2;
      y := '';
      while Valor[a] <> ',' do
        Inc(a);
      Inc(a);
      While Valor[a] <> ']' do
        begin
          y := y + Valor[a];
          Inc(a);
        end;
      Result := StrToInt(y);
    end;
end;

procedure TOperacao.Salvar(Estado: Integer);
var
  Salvar: TQuery;
  NomeDente: array[1..3] of char;
begin
  //Estado = 0 => Inserir
  //Estado = 2 => Excluir
  try
    Salvar := TQuery.Create(Application);
    Salvar.DatabaseName := 'PGODONTO';
    if (Estado = 0) Then
      begin
        Salvar.SQL.Text :=
          'INSERT INTO dente (odonto_cod, dente_nome, dente_operacao, ' +
          'dente_desc, dente_cor) VALUES (:pmcododonto, :pmnomedente, ' +
          ':pmoperacao, :pmdescricao, :pmcor)';
        Salvar.ParamByName('pmcododonto').AsInteger := FCodOdonto;
        Salvar.ParamByName('pmoperacao').AsInteger := FOperacao;
        Salvar.ParamByName('pmnomedente').AsString := FDenteNome;
        if FOperacao = 1 Then
          begin
            Salvar.ParamByName('pmdescricao').AsString := '[' +
              IntToStr(FPosX) + ',' + IntToStr(FPosY) + ']' + FDescricao;
          end
        else
          Salvar.ParamByName('pmdescricao').AsString := FDescricao;
        Salvar.ParamByName('pmcor').AsInteger := FCor;
      end
    Else
      if (Estado = 2) Then
        begin
          Salvar.SQL.Text :=
            'DELETE FROM dente WHERE dente_cod = :pmcod';
          Salvar.ParamByName('pmcod').AsInteger := FCodigo;
        end;
    Salvar.ExecSQL;

    if (FOperacao = 3) and (FDenteNome[1] <> 'c') and
      (FDenteNome[1] <> 'd') Then
      begin
        if (Estado = 0) Then
          begin
            Salvar.SQL.Text :=
              'INSERT INTO dente (odonto_cod, dente_nome, dente_operacao, ' +
              'dente_desc, dente_cor) VALUES (:pmcododonto, :pmnomedente, ' +
              '3, :pmdescricao, :pmcor)';
            Salvar.ParamByName('pmdescricao').AsString := FDescricao;
            Salvar.ParamByName('pmcor').AsInteger := FCor;
          end
        else
          if (Estado = 2) Then

```

```
begin
  Salvar.SQL.Text :=
    'DELETE FROM dente WHERE odonto_cod = :pmcododonto and ' +
    'dente_nome = :pmnomedente and dente_operacao = 3';
  end;
  Salvar.ParamByName('pmcododonto').AsInteger := FCodOdonto;

  if FDenteNome[1] = 'a' Then NomeDente[1] := 'b';
  if FDenteNome[1] = 'b' Then NomeDente[1] := 'a';
  if FDenteNome[1] = 'e' Then NomeDente[1] := 'f';
  if FDenteNome[1] = 'f' Then NomeDente[1] := 'e';

  NomeDente[2] := FDenteNome[2];
  NomeDente[3] := FDenteNome[3];
  Salvar.ParamByName('pmnomedente').AsString :=
    NomeDente[1] + NomeDente[2] + NomeDente[3];
  Salvar.ExecSQL;
end;
Finally
  FreeAndNil(Salvar);
end;
end;
end.
```

31 UNIT UCOLOPERACAO

```

////////// /////////////////////////////////////////////////
// ///////////////////////////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA ///////////////////////////////////////////////////
// Tecnologia em Informática ///////////////////////////////////////////////////
// Projeto de Conclusão de Curso ///////////////////////////////////////////////////
// 3º ano - 2004 ///////////////////////////////////////////////////
// ///////////////////////////////////////////////////
// Desenvolvido por: ///////////////////////////////////////////////////
// - José Altair Ribeiro dos Santos ///////////////////////////////////////////////////
// - Rafael Winter ///////////////////////////////////////////////////
// ///////////////////////////////////////////////////
unit uColOperacao;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs, uOperacao;

type
  TColOperacao = class(TComponent)
  private
    Ocolecao : TQuery;

  public
    CCodigo: Integer;
    CCodOdonto: Integer;
    COperacao: Integer;
    CNomeDente: String;
    CDesc: String;
    CPosX: Integer;
    CPosY: Integer;
    CCor: Integer;

    operacao: TOperacao;

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

    procedure Seleciona;
    procedure SelecionaDente;
    procedure Primeiro;
    procedure Anterior;
    procedure Proximo;
    procedure Ultimo;
    procedure Coloca;
    function Comeco: Boolean;
    function Final: Boolean;

  end;

implementation

{ TColOperacao }

constructor TColOperacao.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Operacao := TOperacao.Create(Self);
  Ocolecao := TQuery.Create(Self);
  Ocolecao.DatabaseName := 'PGODONTO';
end;

destructor TColOperacao.Destroy;
begin
  inherited;

```

```

Ocolecao.Free;

end;

procedure TColOperacao.Colo;
begin
  CCodigo := Ocolecao.FieldByName('dente_cod').AsInteger;
  COperacao := Ocolecao.FieldByName('dente_operacao').AsInteger;
  CNomeDente := Ocolecao.FieldByName('dente_nome').AsString;
  CDesc := Operacao.LerDescricao(Ocolecao.FieldByName('dente_desc').AsString);
  CPosX := Operacao.LerPosX(Ocolecao.FieldByName('dente_desc').AsString);
  CPosY := Operacao.LerPosY(Ocolecao.FieldByName('dente_desc').AsString);
  CCor := Ocolecao.FieldByName('dente_cor').AsInteger;
end;

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

procedure TColOperacao.Seleciona;
begin
  Ocolecao.Close;
  Ocolecao.SQL.Text :=
    'SELECT dente_cod, dente_operacao, dente_nome, dente_desc, dente_cor FROM dente ' +
    'WHERE odonto_cod = :pmcod';
  Ocolecao.ParamByName('pmcod').AsInteger := CCodOdonto;
  Ocolecao.ExecSQL;
  Ocolecao.Open;
end;

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

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

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

function TColOperacao.Comeco: Boolean;
begin
  Result := Ocolecao.Bof;
end;

function TColOperacao.Final: Boolean;
begin
  Result := Ocolecao.Eof;
end;

procedure TColOperacao.SelecionaDente;
begin
  Ocolecao.Close;
  Ocolecao.SQL.Text :=
    'SELECT dente_cod, dente_nome, dente_operacao, dente_desc, dente_cor FROM dente ' +
    'WHERE dente_nome = :pmnome and odonto_cod = :pmcod';
  Ocolecao.ParamByName('pmnome').AsString := CNomeDente;
  Ocolecao.ParamByName('pmcod').AsInteger := CCodOdonto;
  Ocolecao.ExecSQL;
  Ocolecao.Open;
end;

end.

```

32 UNIT URADIO

```

///////////////////////////////
// PROJETO CLÍNICA ODONTOLÓGICA
// Tecnologia em Informática
// Projeto de Conclusão de Curso
// 3º ano - 2004
//
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
//
unit uRadio;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, Jpeg, Dialogs;

type
  TRadio = class(TComponent)

private
  //Fields
  FCodigo: Integer;
  FCodPlano: Integer;
  FDescricao: String;
  FData: TDate;
  FObservacao: String;

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

  //Property's
  property PCodigo: integer read FCodigo write FCodigo;
  property PCodPlano: integer read FCodPlano write FCodPlano;
  property PDescricao: String read FDescricao write FDescricao;
  property PData: TDate read FData write FData;
  property PObservacao: String read FObservacao write FObservacao;

  //Operações
  procedure Carregar(Codigo: Integer);
  procedure Salvar(Estado: Integer);

end;

implementation

{ TRadio }

procedure TRadio.Carregar(Codigo: Integer);
var
  Busca: TQuery;
begin
  try
    Busca := TQuery.Create(Application);
    Busca.DatabaseName := 'PGODONTO';
    Busca.SQL.Text :=
      'SELECT radio_data, radio_desc, radio_obs, lo_export(radio_imagem, ' +
      ':pmimagem) FROM radiografia WHERE radio_cod = :pmcod';
    Busca.ParamByName('pmimagem').AsString := 'c:/temp.jpg';
    Busca.ParamByName('pmcod').AsInteger := Codigo;
    Busca.ExecSQL;
  except
  end;
end;

```

```

Busca.Open;

FCodigo := Codigo;
FData := Busca.FieldByName('radio_data').AsDateTime;
FDescricao := Busca.FieldByName('radio_desc').AsString;
FObservacao := Busca.FieldByName('radio_obs').AsString;
Busca.Close;
Finally
  FreeAndNil(Busca);
end;
end;

constructor TRadio.Create(Dono: TComponent);
begin
  inherited Create(Dono);
end;

destructor TRadio.Destroy;
begin
  inherited;
end;

procedure TRadio.Salvar(Estado: Integer);
var
  Salvar: TQuery;
begin
//Estado = 0 => Inserir
//Estado = 1 => Atualizar
//Estado = 2 => Excluir
try
  Salvar := TQuery.Create(Application);
  Salvar.DatabaseName := 'PGODONTO';

  if (Estado = 0) Then
    begin
      Salvar.Close;
      Salvar.SQL.Text :=
        'INSERT INTO radiografia (plano_cod, radio_imagem, radio_desc, ' +
        'radio_data, radio_obs) VALUES (:pmcodplano, ' +
        ':lo_import(:pmimagem), :pmdesc, :pmdata, :pmobs)';
      Salvar.ParamByName('pmimagem').AsString := 'c:/temp.jpg';
      Salvar.ParamByName('pmcodplano').AsInteger := FCodPlano;
    end
  Else
    if (Estado = 1) Then
      begin
        Salvar.SQL.Text :=
          'UPDATE radiografia SET radio_data = :pmdata, radio_desc = :pmdesc, ' +
          'radio_obs = :pmobs WHERE radio_cod = :pmcodradio';
        Salvar.ParamByName('pmcodradio').AsInteger := FCodigo;
      end
  Else
    if (Estado = 2) Then
      begin
        Salvar.SQL.Text :=
          'DELETE FROM radiografia WHERE radio_cod = :pmcodradio';
        Salvar.ParamByName('pmcodradio').AsInteger := FCodigo;
      end;
  if (Estado < 2) Then
    begin
      Salvar.ParamByName('pmdesc').AsString := FDescricao;
      Salvar.ParamByName('pmdata').AsDateTime := FData;
      Salvar.ParamByName('pmobs').AsString := FObservacao;
    end;
try
  Salvar.ExecSQL;
  if Estado = 0 Then
begin

```

```
    Salvar.Close;
    Salvar.SQL.Text := 
      'SELECT MAX(radio_cod) as qtde FROM radiografia';
    Salvar.ExecSQL;
    Salvar.Open;
    FCodigo := Salvar.FieldByName('qtde').AsInteger;
  end;

  if Estado = 1 Then
    MessageDlg('Radiografia salva com sucesso', mtInformation, [mbOk], 0)
  else if Estado = 2 Then
    MessageDlg('Radiografia excluída com sucesso', mtInformation, [mbOk], 0);
  Except
    MessageDlg('Não foi possível salvar! Verifique a conexão ' +
      'com o Banco de Dados', mtError, [mbOk], 0);
  End;

  Finally
    FreeAndNil(Salvar);
  end;
end;
end.
```

33 UNIT UCOLRADIO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uColRadio;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs, uRadio;  
  
type  
  TColRadio = class(TComponent)  
  private  
    OColecao : TQuery;  
  
  public  
  
    CCodRadio: Integer;  
    CDataRadio: TDate;  
    CDescRadio: String;  
    CCodPlano: Integer;  
  
    constructor Create(Dono: TComponent);  
    destructor Destroy; override;  
    procedure Seleciona;  
    procedure Primeiro;  
    procedure Anterior;  
    procedure Proximo;  
    procedure Ultimo;  
    procedure Coloca;  
    function Comeco: Boolean;  
    function Final: Boolean;  
  
  end;  
  
implementation  
  
{ TColRadio }  
  
constructor TColRadio.Create(Dono: TComponent);  
begin  
  inherited Create(Dono);  
  OColecao := TQuery.Create(nil);  
  OColecao.DatabaseName := 'PGODONTO';  
  
end;  
  
destructor TColRadio.Destroy;  
begin  
  inherited;  
  OColecao.Free;  
  
end;  
  
procedure TColRadio.Coloca;  
begin
```

```

CCodRadio := OColecao.FieldByName('radio_cod').AsInteger;
CDataRadio:= OColecao.FieldByName('radio_data').AsDateTime;
CDescRadio:= OColecao.FieldByName('radio_desc').AsString;

end;

procedure TColRadio.Primeiro;
begin
  OColecao.First;
end;

procedure TColRadio.Seleciona;
begin

OColecao.Close;
OColecao.SQL.Text :=
  'SELECT radio_cod, radio_data, radio_desc FROM radiografia ' +
  'WHERE plano_cod = :pmcod ORDER BY radio_data';
OColecao.ParamByName('pmcod').AsInteger := CCodPlano;
OColecao.ExecSQL;
OColecao.Open;

end;

procedure TColRadio.Ultimo;
begin
  OColecao.Last;
end;

procedure TColRadio.Anterior;
begin
  OColecao.Prior;
end;

procedure TColRadio.Proximo;
begin
  OColecao.Next;
end;

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

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

end.

```

34 UNIT UCOMPROMISMO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uCompromisso;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs;  
  
type  
  TCompromisso = class(TComponent)  
  
private  
  //Fields  
  FCodigo: Integer;  
  FCodPaciente: Integer;  
  FData: TDate;  
  FHora: TDateTime;  
  FDuracao: TDateTime;  
  FHoraAtend: TDateTime;  
  FCliente: String;  
  FFone1: String;  
  FFone2: String;  
  FDescricao: String;  
  FConfirma: Boolean;  
  FObservacao: String;  
  
public  
  
  //Property's  
  property PCodigo: Integer read FCodigo write FCodigo;  
  property PCodPaciente: Integer read FCodPaciente write FCodPaciente;  
  property PData: TDate read FData write FData;  
  property PHora: TDateTime read FHora write FHora;  
  property PDuracao: TDateTime read FDuracao write FDuracao;  
  property PHoraAtend: TDateTime read FHoraAtend write FHoraAtend;  
  property PCliente: String read FCliente write FCliente;  
  property PFone1: String read FFone1 write FFone1;  
  property PFone2: String read FFone2 write FFone2;  
  property PDescricao: String read FDescricao write FDescricao;  
  property PConfirma: Boolean read FConfirma write FConfirma;  
  property PObservacao: String read FObservacao write FObservacao;  
  
  //Operações  
  constructor Create(Dono: TComponent);  
  destructor Destroy; override;  
  procedure Carregar(Codigo: Integer);  
  procedure Salvar(Estado: Integer);  
  
  function SelecionaMaximo(Data: TDate; Hora: TTime): TTime;  
  function SelecionaMinimo(Data: TDate; Hora: TTime): TTime;  
  
end;  
  
implementation
```

```

{ TCompromisso }

procedure TCompromisso.Carregar(Codigo: Integer);
var
  Busca: TQuery;
begin
try
  Busca := TQuery.Create(Application);
  Busca.DatabaseName := 'PGODONTO';
  Busca.SQL.Text :=
    'SELECT agenda_cliente, agenda_fone1, agenda_fone2, agenda_data, ' +
    'agenda_hora, agenda_duracao, agenda_atend, ' +
    'agenda_codpac, agenda_desc, agenda_confirm, agenda_obs FROM agenda ' +
    'WHERE agenda_cod = :pmcod';

  Busca.ParamByName('pmcod').AsInteger := Codigo;

  Busca.ExecSQL;
  Busca.Open;

  FCodigo := Codigo;
  FCliente := Busca.FieldByName('agenda_cliente').AsString;
  FFone1 := Busca.FieldByName('agenda_fone1').AsString;
  FFone2 := Busca.FieldByName('agenda_fone2').AsString;
  FData := Busca.FieldByName('agenda_data').AsDateTime;
  FHora := Busca.FieldByName('agenda_hora').AsDateTime;
  FDuracao := Busca.FieldByName('agenda_duracao').AsDateTime;
  FHoraAtend := Busca.FieldByName('agenda_atend').AsDateTime;
  FCodPaciente := Busca.FieldByName('agenda_codpac').AsInteger;
  FDescricao := Busca.FieldByName('agenda_desc').AsString;
  FObservacao := Busca.FieldByName('agenda_obs').AsString;

  if Busca.FieldByName('agenda_confirm').AsString = '1' then
    FConfirm := True
  else
    FConfirm := False;

  Busca.Close;

  Finally
    FreeAndNil(Busca);
  end;
end;

constructor TCompromisso.Create(Dono: TComponent);
begin
  inherited Create(Dono);
end;

destructor TCompromisso.Destroy;
begin
  inherited;
end;

procedure TCompromisso.Salvar(Estado: Integer);
var
  Salva: TQuery;
begin
//Estado = 0 => Inserir
//Estado = 1 => Atualizar
//Estado = 2 => Excluir
try
  Salva := TQuery.Create(Application);
  Salva.DatabaseName := 'PGODONTO';

  if (Estado = 0) Then
  begin
    Salva.SQL.Text :=
      'INSERT INTO agenda (agenda_data, agenda_hora, agenda_duracao, ' +

```

```

'agenda_atend, agenda_codpac, agenda_cliente, agenda_fone1, ' +
'agenda_fone2, agenda_desc, agenda_confirm, agenda_obs) ' +
'VALUES (:pmdata, :pmhora, :pmduracao, :pmatend, :pmcodpac, :pmcliente, ' +
':pmfone1, :pmfone2, :pmdesc,:pmconfirm, :pmobs)';
end
Else
if (Estado = 1) Then
begin
  Salva.SQL.Text :=
  'UPDATE agenda SET agenda_data = :pmdata, agenda_hora = :pmhora, ' +
  'agenda_duracao = :pmduracao, agenda_atend = :pmatend, agenda_codpac = ' +
  ':pmcodpac, agenda_cliente = :pmcliente, agenda_fone1 = :pmfone1, ' +
  'agenda_fone2 = :pmfone2, agenda_desc = :pmdesc, agenda_confirm = :pmconfirm, ' +
  'agenda_obs = :pmobs WHERE agenda_cod = :pmcod';
  Salva.ParamByName('pmcod').AsInteger := FCodigo;
end
Else
begin
  Salva.SQL.Text :=
  'DELETE FROM agenda WHERE agenda_cod = :pmcod';
  Salva.ParamByName('pmcod').AsInteger := FCodigo;
end;

if (Estado < 2) Then
begin
  Salva.ParamByName('pmdata').AsDateTime := FData;
  Salva.ParamByName('pmhora').AsDateTime := FHora;
  Salva.ParamByName('pmduracao').AsDateTime := FDuracao;
  Salva.ParamByName('pmatend').AsDateTime := FHoraAtend;
  Salva.ParamByName('pmcodpac').AsInteger := FCodPaciente;
  Salva.ParamByName('pmcliente').AsString := FCliente;
  Salva.ParamByName('pmfone1').AsString := FFone1;
  Salva.ParamByName('pmfone2').AsString := FFone2;
  Salva.ParamByName('pmdesc').AsString := FDescricao;
  Salva.ParamByName('pmobs').AsString := FObservacao;
  Salva.ParamByName('pmconfirm').AsBoolean := FConfirm;
end;

Salva.ExecSQL;
Finally
  FreeAndNil(Salva);
end;
end;

```

```

function TCompromisso.SelecionaMaximo(Data: TDate; Hora: TTime): TTime;
var
  Seleciona: TQuery;
  Resultado: TTime;
  StrData, StrHora: String;
begin
  try
    StrData := copy(DATEToStr(Data), 7, 4) +
               copy(DATEToStr(Data), 4, 2) +
               copy(DATEToStr(Data), 1, 2);
    StrHora := TIMEToStr(Hora);

    Seleciona := TQuery.Create(Application);
    Seleciona.DatabaseName := 'PGODONTO';
    Seleciona.SQL.Text :=
    'SELECT MAX(agenda_hora) as valor FROM agenda WHERE agenda_data = ' +
    '+' + QuotedStr(StrData) + ' and agenda_hora < ' + QuotedStr(StrHora);
    Seleciona.ExecSQL;
    Seleciona.Open;

    Resultado := Seleciona.FieldByName('valor').AsDateTime;

    Seleciona.Close;
  Except
    FreeAndNil(Seleciona);
  End;

```

```
    Result := Resultado;
end;

function TCompromisso.SelecionaMinimo(Data: TDate; Hora: TTime): TTime;
var
  Seleciona: TQuery;
  Resultado: TTime;
  StrData, StrHora: String;
begin
  try
    StrData := copy(DateToStr(Data), 7, 4) +
               copy(DateToStr(Data), 4, 2) +
               copy(DateToStr(Data), 1, 2);
    StrHora := TimeToStr(Hora);
    Seleciona := TQuery.Create(Application);
    Seleciona.DatabaseName := 'PGODONTO';
    Seleciona.SQL.Text :=
      'SELECT MIN(agenda_hora) as valor FROM agenda WHERE agenda_data = ' +
      QuotedStr(StrData) + ' and agenda_hora > ' + QuotedStr(StrHora);
    Seleciona.ExecSQL;
    Seleciona.Open;

    Resultado := Seleciona.FieldByName('valor').AsDateTime;

    Seleciona.Close;
  Except
    FreeAndNil(Seleciona);
  End;
  Result := Resultado;
end;
end.
```

35 UNIT UCOLCOMPROMISSO

```

////////// PROJETO CLÍNICA ODONTOLÓGICA //////////
// Projeto de Conclusão de Curso
// 3º ano - 2004
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
unit uColCompromisso;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs;

type
  TColCompromisso = class(TComponent)
  private
    Ocolecao : TQuery;
  public
    CCodCompromisso: Integer;
    CDataCompromisso: TDateTime;
    CHoraCompromisso: TDateTime;
    CDuracaoCompromisso: TDateTime;
    CAgendCompromisso: TDateTime;
    CDescCompromisso: String;
    CNomeCompromisso: String;
    CFone1Compromisso: String;
    CFone2Compromisso: String;
    CStatus: String;

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

    procedure Seleciona;
    procedure Primeiro;
    procedure Anterior;
    procedure Proximo;
    procedure Ultimo;
    procedure Coloca;

    procedure Pesquisa(Nome, Assunto: String; Data1, Data2: TDate; Horal, Hora2:TTime; INo,
IAS, IDa, IHo: Boolean);
    procedure PesquisaPaciente(CodPaciente: Integer; Data: TDate);
    procedure ColocaPesquisa;

    function Comeco: Boolean;
    function Final: Boolean;

  end;

implementation

{ TColCompromisso }

constructor TColCompromisso.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  Ocolecao := TQuery.Create(nil);
  Ocolecao.DatabaseName := 'PGODONTO';

```

```

end;

destructor TColCompromisso.Destroy;
begin
  inherited;
  OColecao.Free;
end;

procedure TColCompromisso.Coloca;
begin
  CCodCompromisso := OColecao.FieldByName('agenda_cod').AsInteger;
  CHoraCompromisso := OColecao.FieldByName('agenda_hora').AsDateTime;
  CDuracaoCompromisso := OColecao.FieldByName('agenda_duracao').AsDateTime;
  CAgendatCompromisso := OColecao.FieldByName('agenda_atend').AsDateTime;
  CDescCompromisso := OColecao.FieldByName('agenda_desc').AsString;
  CNomeCompromisso := OColecao.FieldByName('agenda_cliente').AsString;
  CFone1Compromisso := OColecao.FieldByName('agenda_fone1').AsString;
  CFone2Compromisso := OColecao.FieldByName('agenda_fone2').AsString;
  CStatus := OColecao.FieldByName('agenda_confirm').AsString;
end;

procedure TColCompromisso.Primeiro;
begin
  OColecao.First;
end;

procedure TColCompromisso.Seleciona;
var
  Valor, Data: String;
begin
  Valor := DateToStr(CDataCompromisso);
  Data := copy(Valor, 7, 4) + copy(Valor, 4, 2) + copy(Valor, 1, 2);
  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT agenda_cod, agenda_hora, agenda_duracao, agenda_atend, agenda_desc, ' +
    'agenda_cliente, agenda_fone1, agenda_fone2, agenda_confirm FROM agenda ' +
    'WHERE agenda_data = :pmdata ORDER BY agenda_hora';
  OColecao.ParamByName('pmdata').AsString := Data;
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColCompromisso.Ultimo;
begin
  OColecao.Last;
end;

procedure TColCompromisso.Anterior;
begin
  OColecao.Prior;
end;

procedure TColCompromisso.Proximo;
begin
  OColecao.Next;
end;

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

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

procedure TColCompromisso.Pesquisa(Nome, Assunto: String; Data1,
  Data2: TDate; Horal, Hora2: TTime; INo, IAs, IDa, IHo: Boolean);

```

```

var
  DataIni, DataFin, HoraIni, HoraFin,
  INome, IData, IHora, IAssunto: String;
begin

  DataIni := copy(DateToStr(Data1), 7, 4) +
    copy(DateToStr(Data1), 4, 2) +
    copy(DateToStr(Data1), 1, 2);
  DataFin := copy(DateToStr(Data2), 7, 4) +
    copy(DateToStr(Data2), 4, 2) +
    copy(DateToStr(Data2), 1, 2);
  HoraIni := TimeToStr(Hora1);
  HoraFin := TimeToStr(Hora2);
  INome := 'agenda_cliente like ' + QuotedStr('%' + Nome + '%');
  IData := 'agenda_data BETWEEN ' + QuotedStr(DataIni) + ' AND ' + QuotedStr(DataFin);
  IHora := 'agenda_hora BETWEEN ' + QuotedStr(HoraIni) + ' AND ' + QuotedStr(HoraFin);
  IAssunto := 'agenda_desc like ' + QuotedStr('%' + Assunto + '%');

  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT agenda_cod, agenda_cliente, agenda_desc, agenda_data, agenda_hora, '+
    'agenda_duracao, agenda_confirmada FROM agenda';

  if INo or IDa or IHo or IAs Then
    OColecao.SQL.Text := OColecao.SQL.Text + 'WHERE ';
  if INo Then
    OColecao.SQL.Text := OColecao.SQL.Text + INome;
  if IDa Then
    begin
      if INo Then
        OColecao.SQL.Text := OColecao.SQL.Text + ' AND ';
      OColecao.SQL.Text := OColecao.SQL.Text + IData;
    end;
  if IHo Then
    begin
      if INo or IDa Then
        OColecao.SQL.Text := OColecao.SQL.Text + ' AND ';
      OColecao.SQL.Text := OColecao.SQL.Text + IHora;
    end;
  if IAs Then
    begin
      if INo or IDa or IHo Then
        OColecao.SQL.Text := OColecao.SQL.Text + ' AND ';
      OColecao.SQL.Text := OColecao.SQL.Text + IAssunto;
    end;

  OColecao.SQL.Text := OColecao.SQL.Text + ' ORDER BY agenda_data';
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColCompromisso.ColocaPesquisa;
begin
  CCodCompromisso := OColecao.FieldByName('agenda_cod').AsInteger;
  CDataCompromisso := OColecao.FieldByName('agenda_data').AsDateTime;
  CHoraCompromisso := OColecao.FieldByName('agenda_hora').AsDateTime;
  CDuracaoCompromisso := OColecao.FieldByName('agenda_duracao').AsDateTime;
  CDescCompromisso := OColecao.FieldByName('agenda_desc').AsString;
  CNomeCompromisso := OColecao.FieldByName('agenda_cliente').AsString;
  CStatus := OColecao.FieldByName('agenda_confirmada').AsString;
end;

procedure TColCompromisso.PesquisaPaciente(CodPaciente: Integer;
  Data: TDate);
var
  DataHoje: String;
begin
  DataHoje := copy(DateToStr(Data), 7, 4) +
    copy(DateToStr(Data), 4, 2) +
    copy(DateToStr(Data), 1, 2);
  OColecao.Close;

```

```
Ocolecao.SQL.Text :=  
'SELECT agenda_cod, agenda_cliente, agenda_desc, agenda_data, agenda_hora, '+  
'agenda_duracao, agenda_confirmada FROM agenda WHERE agenda_codpac = ' +  
' :pmcod AND agenda_data >= :pmdata ORDER BY agenda_data, agenda_hora';  
OColecao.ParamByName('pmcod').AsInteger := CodPaciente;  
OColecao.ParamByName('pmdata').AsString := DataHoje;  
OColecao.ExecSQL;  
OColecao.Open;  
end;  
end.
```

36 UNIT UORCAMENTO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uOrcamento;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs;  
  
type  
  TOrcServ = class(TObject)  
  private  
  public  
    PCodigo: Integer;  
    PCodOrc: Integer;  
    PCodSer: Integer;  
    PServNome: String;  
    PValSer: Real;  
    PQtde: Integer;  
    Total: Real;  
    Situacao: Integer;  
    procedure Salvar;  
  end;  
  
TOrcamento = class(TComponent)  
  private  
    //Fields  
    FCodigo: Integer;  
    FCodPac: Integer;  
    FNomePaciente: String;  
    FFone1: String;  
    FFone2: String;  
    FDataOrcamento: TDate;  
    FDesconto: Real;  
    FFormaPagamento: Boolean;  
    FValorEntrada: Real;  
    FDataPriParc: TDate;  
    FNumParc: Integer;  
    FAprovado: Boolean;  
  
  public  
    Servico: array of TOrcServ;  
  
  constructor Create(Dono: TComponent);  
  destructor Destroy; override;  
  
  function LerTotal: Real;  
  function LerDesc: Real;  
  
  //Property's  
  property PCodigo: Integer read FCodigo write FCodigo;  
  property PCodPac: Integer read FCodPac write FCodPac;  
  property PNomePaciente: String read FNomePaciente write FNomePaciente;  
  property PFone1: String read FFone1 write FFone1;  
  property PFone2: String read FFone2 write FFone2;  
  property PDataOrcamento: TDate read FDataOrcamento write FDataOrcamento;
```

```

property PDesconto: Real read FDesconto write FDesconto;
property PFormaPg: Boolean read FFormaPagamento write FFormaPagamento;
property PValorEntrada: Real read FValorEntrada write FValorEntrada;
property PDataPriParc: TDate read FDataPriParc write FDataPriParc;
property PNumParc: Integer read FNumParc write FNumParc;
property PAprovado: Boolean read FAprovado write FAprovado;

//Operações
procedure Carregar;
procedure Salvar(Estado: Integer);
procedure LerServicos;

end;

implementation

{ TOrcamento }

constructor TOrcamento.Create(Dono: TComponent);
begin
  inherited Create(Dono);

end;

procedure TOrcamento.Carregar;
var
  Ler: TQuery;
begin
  try
    Ler := TQuery.Create(Self);
    Ler.DatabaseName := 'PGODONTO';
    Ler.SQL.Text :=
      'SELECT a.orc_cod AS codigo, a.pac_cod AS codpac, b.pac_nome AS nome, ' +
      'b.pac_fone1 AS f1, b.pac_fone2 AS f2, a.orc_data AS data, ' +
      'a.orc_desc AS desconto, a.orc_formpag AS forma, a.orc_numpar AS npar, ' +
      'a.orc_aprov AS aprovado, a.orc_valorent AS entrada, ' +
      'a.orc_pripar AS pripar FROM orcamento a, paciente b ' +
      'WHERE orc_cod = :pmcod AND a.pac_cod = b.pac_cod';
    Ler.ParamByName('pmcod').AsInteger := FCodigo;
    Ler.ExecSQL;
    Ler.Open;
    FCodigo := Ler.FieldByName('codigo').AsInteger;
    FCodPac := Ler.FieldByName('codpac').AsInteger;
    FNomePaciente := Ler.FieldByName('nome').AsString;
    FFone1 := Ler.FieldByName('f1').AsString;
    FFone2 := Ler.FieldByName('f2').AsString;
    FDataOrcamento := Ler.FieldByName('data').AsDateTime;
    FDesconto := Ler.FieldByName('desconto').AsFloat;
    if Ler.FieldByName('forma').AsString = '1' Then
      FFormaPagamento := True
    else
      FFormaPagamento := False;
    FValorEntrada := Ler.FieldByName('entrada').AsFloat;
    FDataPriParc := Ler.FieldByName('pripar').AsDateTime;
    FNumParc := Ler.FieldByName('npar').AsInteger;
    if Ler.FieldByName('aprovado').AsString = '1' Then
      FAprovado := True
    else
      FAprovado := False;
    Ler.Close;
    LerServicos;
  Finally
    FreeAndNil(Ler);
  end;
end;

procedure TOrcamento.Salvar(Estado: Integer);

```

```

var
  Salvar: TQuery;
  a: Integer;
begin
//Estado = 0 => Inserir
//Estado = 1 => Atualizar
//Estado = 2 => Excluir

try
  Salvar := TQuery.Create(Application);
  Salvar.DatabaseName := 'PGODONTO';

  if (Estado = 0) Then
    begin
      Salvar.SQL.Text :=
        'INSERT INTO orcamento (pac_cod, orc_data, orc_desc, orc_formpag, ' +
        'orc_numpar, orc_aprov, orc_valorent, orc_pripar) VALUES ' +
        '(:pmcodpac, :pmdata, :pmdesc, :pmformpag, :pmnumpar, :pmaprov, ' +
        ':pmvalorent, :ppripar)';
    end
  Else
    if (Estado = 1) Then
      begin
        Salvar.SQL.Text :=
          'UPDATE orcamento SET pac_cod = :pmcodpac, orc_data = :pmdata, ' +
          'orc_desc = :pmdesc, orc_formpag = :pmformpag, ' +
          'orc_numpar = :pmnumpar, orc_aprov = :pmaprov, ' +
          'orc_valorent = :pmvalorent, orc_pripar = :ppripar WHERE ' +
          'orc_cod = :pmcod';
        Salvar.ParamByName('pmcod').AsInteger := FCodigo;
      end
    Else
      begin
        Salvar.SQL.Text :=
          'DELETE FROM orcamento WHERE orc_cod = :pmcod';
        Salvar.ParamByName('pmcod').AsInteger := FCodigo;
      end;
  if (Estado < 2) Then
    begin
      Salvar.ParamByName('pmcodpac').AsInteger := FCodPac;
      Salvar.ParamByName('pmdata').AsDateTime := FDataOrcamento;
      Salvar.ParamByName('pmdesc').AsFloat := FDesconto;
      Salvar.ParamByName('pmformpag').AsBoolean := FFormaPagamento;
      Salvar.ParamByName('pmnumpar').AsInteger := FNumParc;
      Salvar.ParamByName('pmaprov').AsBoolean := FAprovado;
      Salvar.ParamByName('pmvalorent').AsFloat := FValorEntrada;
      Salvar.ParamByName('ppripar').AsDateTime := FDataPriParc;

    end;
  try
    Salvar.ExecSQL;
    if (Estado = 0) Then
      begin
        Salvar.Close;
        Salvar.SQL.Text :=
          'SELECT MAX(orc_cod) as qtde FROM orcamento';
        Salvar.ExecSQL;
        Salvar.Open;
        FCodigo := Salvar.FieldByName('qtde').AsInteger;
      end;
    if (Estado < 2) then
      for a:= 0 to High(Servico) do
        begin
          Servico[a].PCodOrc := FCodigo;
          Servico[a].Salvar;
        end;
    if (Estado < 2) Then
      MessageDlg('Dados do orçamento salvos com sucesso', mtInformation,
                 [mbOk], 0)
  else

```

```

        MessageDlg('Orçamento excluído com sucesso', mtInformation,
                   [mbOk], 0);
    Except
        MessageDlg('Não foi possível salvar! Verifique a conexão ' +
                   'com o Banco de Dados', mtError, [mbOk], 0);
    End;

    Finally
        FreeAndNil(Salvar);
    end;
end;

destructor TOrcamento.Destroy;
begin
    inherited;
end;

procedure TOrcamento.LerServicos;
var
    LerServicos: TQuery;
    a: Integer;
begin
    try
        LerServicos := TQuery.Create(Self);
        LerServicos.DatabaseName := 'PGODONTO';
        LerServicos.SQL.Text :=
            'SELECT a.orcserv_cod AS f1, a.orc_cod AS f2, a.serv_cod AS f3, ' +
            'a.orcserv_valor AS f4, b.serv_nome AS f5, a.orcserv_qtde AS f6 FROM ' +
            'orcserv a, servicos b WHERE a.orc_cod = :pmorccod ' +
            'AND a.serv_cod = b.serv_cod';
        LerServicos.ParamByName('pmorccod').AsInteger := FCodigo;
        LerServicos.ExecSQL;
        LerServicos.Open;
        LerServicos.First;
        a:= 0;
        while not LerServicos.Eof do
        begin
            begin
                SetLength(Servico, a+1);
                if not ASSIGNED(Servico[a]) Then
                    Servico[a] := TOrcServ.Create;
                Servico[a].PCodigo :=
                    LerServicos.FieldByName('f1').AsInteger;
                Servico[a].PCodOrc :=
                    LerServicos.FieldByName('f2').AsInteger;
                Servico[a].PCodSer :=
                    LerServicos.FieldByName('f3').AsInteger;
                Servico[a].PValSer :=
                    LerServicos.FieldByName('f4').AsFloat;
                Servico[a].PServNome :=
                    LerServicos.FieldByName('f5').AsString;
                Servico[a].PQtde :=
                    LerServicos.FieldByName('f6').AsInteger;
                Servico[a].Total := Servico[a].PQtde * Servico[a].PValSer;
                Servico[a].Situacao := 0;
                LerServicos.Next;
                Inc(a);
            end;
        end;
    Finally
        FreeAndNil(LerServicos);
    end;
end;

function TOrcamento.LerTotal: Real;
var
    a: Integer;
    valor: Real;
begin
    valor := 0;

```

```

for a:= 0 to High(Servico) do
  if (Servico[a].Situacao < 3) Then
    valor := valor + Servico[a].Total;
  Result := Valor;
end;

function TOrcamento.LerDesc: Real;
begin
  Result := LerTotal - PDesconto;
end;

{ TOrcServ }

procedure TOrcServ.Salvar;
var
  Salvar: TQuery;
begin
  //Situacao = 0 >> LER
  //Situacao = 1 >> SALVAR
  //Situacao = 2 >> ATUALIZAR
  //Situacao = 3 >> DELETAR
  //Situacao = 4 >> DESCONSIDERAR
  if (Situacao > 0) and (Situacao < 4) Then
    begin
      try
        Salvar := TQuery.Create(nil);
        Salvar.DatabaseName := 'PGODONTO';
        case Situacao of
          1: begin
            Salvar.SQL.Text :=
              'INSERT INTO orcserv (orc_cod, serv_cod, orcserv_qtde, ' +
              'orcserv_valor) VALUES (:pmorccod, :pmservcod, :pmqtde, ' +
              ':pmvalor)';
            Salvar.ParamByName('pmorccod').AsInteger := PCodOrc;
            Salvar.ParamByName('pmservcod').AsInteger := PCodSer;
            Salvar.ParamByName('pmqtde').AsInteger := PQtde;
            Salvar.ParamByName('pmvalor').AsFloat := PValSer;
          end;
          2: begin
            Salvar.SQL.Text :=
              'UPDATE orcserv SET orcserv_qtde = :pmqtde, orcserv_valor = ' +
              ':pmvalor WHERE orcserv_cod = :pmcod';
            Salvar.ParamByName('pmcod').AsInteger := PCodigo;
            Salvar.ParamByName('pmqtde').AsInteger := PQtde;
            Salvar.ParamByName('pmvalor').AsFloat := PValSer;
          end;
          3: begin
            Salvar.SQL.Text :=
              'DELETE FROM orcserv WHERE orcserv_cod = :pmcod';
            Salvar.ParamByName('pmcod').AsInteger := PCodigo;
          end;
        end;
        Salvar.ExecSQL;
        if (Situacao = 1) or (Situacao = 2) Then
          Situacao := 0;
      Finally
        FreeAndNil(Salvar);
      End;
    end;
  end;
end.

```

37 UNIT UCOLORCAMENTO

```

////////// PROJETO CLÍNICA ODONTOLÓGICA //////////
// Projeto de Conclusão de Curso
// 3º ano - 2004
// Desenvolvido por:
// - José Altair Ribeiro dos Santos
// - Rafael Winter
unit uColOrcamento;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, Dialogs, uOrcamento;

type
  TColOrcamento = class(TComponent)
  private
    OColecao : TQuery;
  public
    CCodOrcamento: Integer;
    CNomePaciente: String;
    CValor: Real;
    OrcamentoAtual: TOrcamento;
    constructor Create(Dono: TComponent);
    destructor Destroy; override;
    procedure Seleciona(Campo, Valor: String);
    procedure Primeiro;
    procedure Anterior;
    procedure Proximo;
    procedure Ultimo;
    procedure Coloca;
    function Comeco: Boolean;
    function Final: Boolean;
  end;

implementation

{ TColOrcamento }

constructor TColOrcamento.Create(Dono: TComponent);
begin
  inherited Create(Dono);
  OrcamentoAtual := TOrcamento.Create(Self);
  OColecao := TQuery.Create(nil);
  OColecao.DatabaseName := 'PGODONTO';
end;

destructor TColOrcamento.Destroy;
begin
  inherited;
  OColecao.Free;
end;

procedure TColOrcamento.Coloca;
begin

```

```

CCodOrcamento := OColecao.FieldByName('f1').AsInteger;
CNomePaciente := OColecao.FieldByName('f2').AsString;
CValor := OColecao.FieldByName('valor').AsFloat;
end;

procedure TColOrcamento.Primeiro;
begin
  OColecao.First;
end;

procedure TColOrcamento.Seleciona(Campo, Valor: String);
begin
  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT a.orc_cod AS f1, c.pac_nome AS f2, ' +
    '(sum(b.orcserv_valor * b.orcserv_qtde)) - a.orc_desc AS valor ' +
    'FROM orcamento a, orcserv b, paciente c WHERE b.orc_cod = a.orc_cod AND c.pac_cod =
a.pac_cod ';
  if Campo = 'a.orc_cod' then
    begin
      if Valor = '' Then
        Valor := '0';
      OColecao.SQL.Text := OColecao.SQL.Text + ' AND ' + Campo + ' = ' + Valor;
    end
  else
    if Campo = 'c.pac_nome' then
      OColecao.SQL.Text := OColecao.SQL.Text + ' AND ' + Campo + ' like ' +
      QuotedStr('%'+ Valor + '%');

  OColecao.SQL.Text := OColecao.SQL.Text + ' GROUP BY a.orc_cod, ' +
  'a.orc_desc, c.pac_nome ORDER BY c.pac_nome';
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColOrcamento.Ultimo;
begin
  OColecao.Last;
end;

procedure TColOrcamento.Anterior;
begin
  OColecao.Prior;
end;

procedure TColOrcamento.Proximo;
begin
  OColecao.Next;
end;

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

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

end.

```

38 UNIT USERVICOS

```

////////// 
// 
// PROJETO CLÍNICA ODONTOLÓGICA 
// Tecnologia em Informática 
// Projeto de Conclusão de Curso 
// 3º ano - 2004 
// 
// Desenvolvido por: 
// - José Altair Ribeiro dos Santos 
// - Rafael Winter 
// 
////////// 
unit uServicos;

interface

uses
  Controls, Classes, DBTables, SysUtils, Forms, dialogs;

type
  TServicos = class(TComponent)

private
  //Fields
  FCodigo: integer;
  FNome: string;
  FValor: real;

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

  //Property's
  property PCodigo: Integer read FCodigo write FCodigo;
  property PNome: String read FNome write FNome;
  property PValor: Real read FValor write FValor;

  //Operações
  procedure Salvar(Estado: integer);
  procedure Carregar(campo, valor: string);

end;

implementation

{ TServicos }

constructor TServicos.Create(Dono: TComponent);
begin
  inherited Create(Dono);
end;

procedure TServicos.Salvar(estado: integer);
var
  Salvar: TQuery;
begin
  //Estado = 0 >> Inserir
  //Estado = 1 >> Atualizar
  //Estado = 2 >> Excluir
  try
    Salvar := TQuery.Create(Self);
    Salvar.DatabaseName := 'PGODONTO';

    if (Estado = 0) then

```

```

begin
  Salvar.Close;
  //INSERINDO DADOS NA TABELA
  Salvar.SQL.Text :=
    'INSERT INTO servicios (serv_nome, serv_valor) ' +
    'VALUES (:pmnome, :pmvalor)';
  end
else
if (Estado = 1) then
begin
  //ATUALIZANDO DADOS NA TABELA
  Salvar.SQL.Text :=
    'UPDATE servicios SET serv_nome = :pmnome, serv_valor = :pmvalor ' +
    'WHERE serv_cod = :pmcod';
  Salvar.ParamByName('pmcod').AsInteger := FCodigo;
end;

//ATRIBUIR PARÂMETROS
Salvar.ParamByName('pmnome').AsString := FNome;
Salvar.ParamByName('pmvalor').AsFloat := FValor;

Try
  Salvar.ExecSQL;
  MessageDlg('Dados salvos com sucesso', mtInformation, [mbOk], 0)
Except
  MessageDlg('Não foi possível salvar! Verifique a conexão ' +
    'com o Banco de Dados', mtError, [mbOk], 0);
End;
Finally
  FreeAndNil(Salvar);
End;

end;

procedure TServicos.Carregar(campo, valor: string);
var
  Ler: TQuery;
begin
  Ler := TQuery.Create(Self);

  try
    Ler.DatabaseName := 'PGODONTO';
    Ler.SQL.Text := 'SELECT serv_cod, serv_nome, serv_valor FROM servicios ' +
      'WHERE ' + campo + ' = :pmvalor';

    if campo = 'serv_cod' then
      Ler.ParamByName('pmvalor').AsInteger := StrToInt(Valor)
    else
      if campo = 'serv_nome' then
        Ler.ParamByName('pmvalor').AsString := Valor;

    Ler.Open;
    FCodigo := Ler.fieldbyname('serv_cod').AsInteger;
    FNome := Ler.fieldbyname('serv_nome').AsString;
    FValor := Ler.fieldbyname('serv_valor').AsFloat;
  Finally
    FreeAndNil(Ler);
  End;

end;

destructor TServicos.Destroy;
begin
  inherited;
end;

end.

```

39 UNIT UCOLSERVICOS

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uColServicos;  
  
interface  
  
uses  
  Controls, Classes, DBTables, SysUtils, Forms, uServicos;  
  
type  
  TColServicos = class(TComponent)  
  private  
    OColecao : TQuery;  
  
  public  
  
    CCodigo: Integer;  
    CNome: String;  
    CValor: Real;  
  
    //ServiçoAtual: TServicos;  
  
    constructor Create(Dono: TComponent);  
    destructor Destroy; override;  
  
    procedure Seleciona(Nome: String);  
    procedure Primeiro;  
    procedure Anterior;  
    procedure Proximo;  
    procedure Ultimo;  
    procedure Coloca;  
    function Comeco: Boolean;  
    function Final: Boolean;  
  
  end;  
  
implementation  
  
{ TColPlano }  
  
constructor TColServicos.Create(Dono: TComponent);  
begin  
  inherited Create(Dono);  
  // ServicoAtual := TServicos.Create(Self);  
  
  OColecao := TQuery.Create(Self);  
  OColecao.DatabaseName := 'PGODONTO';  
  
end;  
  
destructor TColServicos.Destroy;  
begin  
  inherited;  
  OColecao.Free;  
  
end;
```

```

procedure TColServicos.Coloa;
begin
  CCodigo := OColecao.FieldByName('serv_cod').AsInteger;
  CNome := OColecao.FieldByName('serv_nome').AsString;
  CValor := OColecao.FieldByName('serv_valor').AsFloat;
end;

procedure TColServicos.Primeiro;
begin
  OColecao.First;
end;

procedure TColServicos.Seleciona(Nome: String);
begin
  OColecao.Close;
  OColecao.SQL.Text :=
    'SELECT serv_cod, serv_nome, serv_valor FROM servicios ' +
    'WHERE serv_nome like :pmnome ORDER BY serv_nome';
  OColecao.ParamByName('pmnome').AsString := '%' + Nome + '%';
  OColecao.ExecSQL;
  OColecao.Open;
end;

procedure TColServicos.Ultimo;
begin
  OColecao.Last;
end;

procedure TColServicos.Anterior;
begin
  OColecao.Prior;
end;

procedure TColServicos.Proximo;
begin
  OColecao.Next;
end;

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

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

end.

```

40 UNIT UUSUARIO

```
//////////  
//  
// PROJETO CLÍNICA ODONTOLÓGICA  
// Tecnologia em Informática  
// Projeto de Conclusão de Curso  
// 3º ano - 2004  
//  
// Desenvolvido por:  
// - José Altair Ribeiro dos Santos  
// - Rafael Winter  
//  
/////////  
unit uUsuario;  
  
interface  
  
uses  
  Controls, DBTables, SysUtils, Forms, Dialogs;  
  
type  
  TUsuario = class(TObject)  
  
private  
  //Fields  
  FCodigo: Integer;  
  FNome: String;  
  FLogin: String;  
  FSenha: String;  
  FRoot: Boolean;  
  FDadosPaciente: Boolean;  
  FAnamnese: Boolean;  
  FExame: Boolean;  
  FPlano: Boolean;  
  FAgenda: Boolean;  
  FOrcamento: Boolean;  
  FAtestado: Boolean;  
  FFelicitacao: Boolean;  
  
public  
  
  constructor Create;  
  destructor Destroy; override;  
  
  //Property's  
  property PCodigo: Integer read FCodigo write FCodigo;  
  property PNome: String read FNome write FNome;  
  property PLogin: String read FLogin write FLogin;  
  property PSenha: String read FSenha write FSenha;  
  property PRoot: Boolean read FRoot write FRoot;  
  property PDadosPaciente: Boolean read FDadosPaciente write FDadosPaciente;  
  property PAnamnese: Boolean read FAnamnese write FAnamnese;  
  property PExame: Boolean read FExame write FExame;  
  property PPlano: Boolean read FPlano write FPlano;  
  property PAgenda: Boolean read FAgenda write FAgenda;  
  property POrcamento: Boolean read FOrcamento write FOrcamento;  
  property PAtestado: Boolean read FAtestado write FAtestado;  
  property PFelicitacao: Boolean read FFelicitacao write FFelicitacao;  
  
  //Operações  
  procedure Carregar(Login: String);  
  procedure Salvar(Estado: Integer);  
  
end;  
  
implementation
```

```

{ TUusuario }

constructor TUusuario.Create;
begin
  inherited Create;
end;

procedure TUusuario.Carregar(Login: String);
var
  Busca: TQuery;
begin
  try
    Busca := TQuery.Create(Application);
    Busca.DatabaseName := 'PGODONTO';
    Busca.SQL.Text :=
      'SELECT * FROM usuarios WHERE usu_login = :pmlogin';
    Busca.ParamByName('pmlogin').AsString := Login;
    Busca.ExecSQL;

    Busca.Open;

    FCodigo := Busca.FieldByName('usu_cod').AsInteger;
    FName := Busca.FieldByName('usu_nome').AsString;
    FLogin := Busca.FieldByName('usu_login').AsString;
    FSenha := Busca.FieldByName('usu_senha').AsString;
    if Busca.FieldByName('usu_root').AsString = '1' Then
      FRoot := True
    else
      FRoot := False;
    if Busca.FieldByName('usu_paciente').AsString = '1' Then
      FDadosPaciente := True
    else
      FDadosPaciente := False;
    if Busca.FieldByName('usu_anamnese').AsString = '1' Then
      FAnamnese := True
    else
      FAnamnese := False;
    if Busca.FieldByName('usu_exame').AsString = '1' Then
      FExame := True
    else
      FExame := False;
    if Busca.FieldByName('usu_plano').AsString = '1' Then
      FPlano := True
    else
      FPlano := False;
    if Busca.FieldByName('usu_agenda').AsString = '1' Then
      FAgenda := True
    else
      FAgenda := False;
    if Busca.FieldByName('usu_orcamento').AsString = '1' Then
      FOrcamento := True
    else
      FOrcamento := False;
    if Busca.FieldByName('usu_atestado').AsString = '1' Then
      FAtestado := True
    else
      FAtestado := False;
    if Busca.FieldByName('usu_felicitacao').AsString = '1' Then
      FFelicitacao := True
    else
      FFelicitacao := False;
    Busca.Close;
  Finally
    FreeAndNil(Busca);
  end;

  end;

procedure TUusuario.Salvar(Estado: Integer);
var

```

```

    Salva: TQuery;
begin
//Estado = 0 => Inserir
//Estado = 1 => Atualizar
//Estado = 2 => Excluir

try

    Salva := TQuery.Create(Application);
    Salva.DatabaseName := 'PGODONTO';

    if (Estado = 0) Then
        begin
            Salva.SQL.Text :=
                'INSERT INTO usuarios (usu_login, usu_nome, usu_senha, usu_root, ' +
                'usu_paciente, usu_anamnese, usu_exame, usu_plano, usu_agenda, ' +
                'usu_orcamento, usu_atestado, usu_felicitacao) VALUES (:pmlogin, ' +
                ':pmsenha, :pmroot, :mpaciente, :pmanamnese, :pmexame, ' +
                ':pmlongo, :magenda, :morcamento, :patestado, :mfelicitacao)';
            Salva.ParamByName('pmnome').AsString := FNome;
            Salva.ParamByName('pmsenha').AsString := FSenha;
            Salva.ParamByName('pmroot').AsBoolean := FRoot;
            Salva.ParamByName('mpaciente').AsBoolean := FDadosPaciente;
            Salva.ParamByName('pmanamnese').AsBoolean := FAnamnese;
            Salva.ParamByName('pmexame').AsBoolean := FExame;
            Salva.ParamByName('pmlongo').AsBoolean := FPlano;
            Salva.ParamByName('magenda').AsBoolean := FAgenda;
            Salva.ParamByName('morcamento').AsBoolean := FOrcamento;
            Salva.ParamByName('patestado').AsBoolean := FAtestado;
            Salva.ParamByName('mfelicitacao').AsBoolean := FFelicitacao;
        end
    Else
        if (Estado = 2) Then
            begin
                Salva.SQL.Text :=
                    'DELETE FROM usuarios WHERE usu_login = :pmlogin';
            end;

    Salva.ParamByName('pmlogin').AsString := FLogin;
    try
        Salva.ExecSQL;
        if Estado = 0 Then
            MessageDlg('Usuário cadastrado com sucesso!', mtInformation, [mbOk], 0)
        else
            if Estado = 2 Then
                MessageDlg('Usuário excluído com sucesso!', mtInformation, [mbOk], 0)
    Except
        MessageDlg('Não foi possível cadastrar/excluir usuário! Verifique a conexão ' +
                   'com o Banco de Dados', mtError, [mbOk], 0);
    End;
    Finally
        FreeAndNil(Salva);
    end;
end;

destructor TUsuario.Destroy;
begin
    inherited;
end;

end.

```