UNIVERSIDADE FEDERAL DO PARANÁ
PÓS-GRADUAÇÃO EM INFORMÁTICA, SETOR DE CIÊNCIAS EXATAS

RODRIGO DE AZEVEDO CARVALHO

ANÁLISE TECNOLOGICA DE DESENVOLVIMENTO DE SOFTWARE PARA GERENCIAMENTO DE REPRESENTAÇÃO COMERCIAL

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2012

RODRIGO DE AZEVEDO CARVALHO

ANÁLISE TECNOLÓGICA DE DESENVOLVIMENTO DE SOFTWARE PARA GERENCIAMENTO DE REPRESENTAÇÃO COMERCIAL

Monografia apresentada à disciplina Desenvolvimento de Sistemas Web como requisito parcial à conclusão do Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Dr. Andrey Ricardo Pimentel

CURITIBA

2012

Parecer de Aprovação
Monografia de Especialização em Informática
Ênfase em Desenvolvimento de Sistemas para a Web
Programa de Pós-Graduação em Informática/UFPR

Declaramos que o aluno **RODRIGO DE AZEVEDO CARVALHO** entregou a versão final da sua Monografia de Especialização em Informática da Universidade Federal do Paraná, com Ênfase em Desenvolvimento de Sistemas para a Web, intitulada ***Análise tecnologica de desenvolvimento de software para gerenciamento de representação comercial.***

Curitiba, 28 de Novembro 2012

ANDREY RICARDO PIMENTEL
Professor Adjunto
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

NELSON SUGA
Professor Adjunto
Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Informática
Caixa Postal 19081
CEP 81531-990 - Curitiba-PR

RESUMO

As empresas de representação comercial tem se apresentado muito conservadoras no que se trata a tecnologia e comunicação. Com a evolução tecnológica hoje é necessário acesso rápido, inteligente e organizado as informações, por este motivo, assim como em outros setores, as representações comerciais necessitam aprimorar suas formas de comunicação, de modo melhorar a agilidade do atendimento possibilitando um melhor atendimento e organização para com seus clientes e fornecedores, assim como seus colaboradores.

Este trabalho tem como objetivo abordar as carências de um setor que se encontra em expansão e necessita de uma maior atenção na organização de seus métodos de gerenciamento financeiro, de estoque e em suas comunicações. Serão vistos aspectos de maior deficiência para tornar o objetivo direto do desenvolvimento desta ferramenta.

Palavras-chave: Representação. Comercial. Gerenciamento Comercial. Administração de representações comerciais.

ABSTRACT

Representation companies has been very conservative when it comes to technology and communication. With the current technological trends, you need quick access, intelligent and organized at information, for this reason, as well as in other sectors, commercial representation need to improve their ways of communication, so improving the agility of care by providing better service and organization to with its customers and suppliers, as well as its employees.

This work aims to address the needs of an industry that is growing and needs more attention in your organization and methods of financial management, inventory and communications. Will be seen most aspects of deficiences to make the direct purpose of developing this tool.
Keywords: Representation. Commercial. Commercial Management. Administration commercial representations.

LISTA DE ABREVIATURAS E LISTA DE SIGLAS

AJAX - *Asynchronous JavaScript and XML*
ANSI - *American National Standards Institute*
API - *Application Programming Interface*
AWT - *Abstract Window Toolkit*
BD - Banco de dados
BPEL - *Business Process Execution Language*
CPU - *Central Processing Unit*
CSS - *Cascading Style Sheets*
CVS - *Concurrent Versioning System*
EIS - *Enterprise Information System*
EJB - *Enterprise JavaBeans*.
FAQ - *Frequently Asked Questions*
GIS - *Geographic information system*
GPL - *General Public License*
GUI - *Grafic User Interface*
GWT - *Google Web Tool Kit*
HQL - *Hibernate Query Language*
HTML - *HyperText Markup Language*
HTTP - *Hyper Text Transfer Protocol*
IDE - *Integrated Development Environment*
IDL - *Interface Definition Language*
IO - IN/OUT
J2EE - *Java 2 Enterprise Edition*
J2ME - *Java 2 Micro Edition*.
J2SE - *Java 2 Standard Editon*
JAR - *Java ARchive*
JCA - *Java Connector Architecture*
JDBC - *Java Database Connectivity*
JDK - *Java Developer Kit*
JDO - *Java Data Objects*
JMS - *Java Messaging System*

JRE - *Java Runtime Enviroment*
JSF - *Java Server Faces*
JSP - *Java Server Pages*
JTA - *Java Transaction API*
JVM - *Java Virtual Machine*
LGPL - *Lesser General Public License*
LINK - *Language Integrated Query*
MOM - *Message Oriented Middleware*
OO - Orientação a Objetos
ODBC - *Open Database Connectivity*
OLAP - O*nline Analytical Processing*
OLTP - *Online Transaction Processing*
PSQL - *Stored Procedures e Triggers*
RAC - *Oracle Real Application Clusters*
REST - *Representational State Transfer*
RH - Recursos Humanos
RPC - *Remote Procedure Calls*
RMI - *Remote Method Invocation*
SAP - *Systems, Applications, and Products*
SGBD - Sistema de Gerenciamento de Banco de Dados
SOAP - *Simple Object Access Protocol*
SQL - *Structured Query Language*
TCP/IP - Transmissão *Control Protocol/Internet Protocol*
T-SQL - Transact - *Structured Query Language*
UI - *User Interface*
UML - *Unified Modeling Language*
WML - *Wireless Markup Language*
WSDL - *Web Service Description Language*
XML - *Extensible Markup Language*

SUMÁRIO

1. **INTRODUÇÃO**

Atualmente toda a empresa que pretende expandir necessita informatizar os seus procedimentos. Isso ocorre por conta do constante aumento da necessidade de comunicação e armazenagem de dados. As empresas que possuem um *software* adequado, e fazem o correto uso das informações obtidas por ele, tem múltiplos benefícios e grandes chances de expansão.

A rápida evolução de dispositivos como os computadores, celulares, tablets, televisores e outros aparelhos eletroeletrônicos possibilitam a facilidade de comunicação entre as empresas e seus clientes. Hoje, os dispositivos móveis tem se popularizado rapidamente e isso faz com que as empresas que os produzem invistam cada vez mais em novos recursos e capacidades. Estes aparelhos já possuem processadores de última geração com múltiplos núcleos, podendo atingir velocidades muitas vezes superior a de computadores pessoais (desktops) e notebooks. Além disso, já possuem uma ampla memória e uma capacidade de armazenamento de certa forma ilimitada, bastando apenas fazer a troca de pequenos cartões de memória ou dispositivos de armazenamento USB que podem ter até 64GB.

As empresas da área de representações tendem a usufruir dessa tecnologia com uma maior abrangência, ao manter contato com clientes de diversas cidades e em várias regiões. A utilização de tecnologias móveis vem a ajudar no trabalho de comunicação entre os representantes (vendedores) e a central da empresa, assim como, na comunicação.

Este trabalho foi motivado pelo interesse em auxiliar empresas do ramo de representações comerciais. Este empreendimento está em expansão e necessita de um rápido aprimoramento de seus métodos de gerenciamento financeiro, de organização de estoque e em suas comunicações.

O propósito deste projeto é analisar a tecnologia que envolve o desenvolvimento de um *software* capaz de prover a armazenagem e organização dos dados de uma empresa desta área, através de métodos para o desenvolvimento de *software*s específicos.

2. **REFERENCIAL TEÓRICO**

Neste capítulo, será tratado o que são as representações comerciais e quais as formas de abordagem para este setor. Como devemos enxergar as diversas atividades que são de responsabilidade de seus funcionários e como podemos deixar este processo mais enxuto.

2.1. REPRESENTAÇÃO COMERCIAL

A representação comercial é uma importante atividade de apoio às vendas das indústrias e do comércio atacadista. Como atua no processo de distribuição de produtos de diversos segmentos, a representação comercial é uma das formas mais usuais de expandir os negócios. O representante comercial intermedia e facilita o relacionamento entre o produtor ou fornecedor do produto e o cliente.

É importante definir o que é a representação comercial, algumas de suas obrigações e quais as atividades que são executadas pelos representantes comerciais e pelas empresas que utilizam os seus serviços.

> Definição de representação comercial, de acordo com a legislação, “Exerce a representação comercial autônoma a pessoa jurídica ou a pessoa física, sem relação de emprego, que desempenha, em caráter não eventual por conta de uma ou mais pessoas, a mediação para a realização de negócios mercantis, agenciando propostas ou pedidos, para, transmiti los aos representados, praticando ou não atos relacionados com a execução dos negócios. (Art . 1º, LEI Nº 4.886, de 9 de Dezembro de 1965.)

Portanto, a representação é uma modalidade de intermediação de negócios mercantis. Os representantes comerciais têm a função de facilitar os negócios envolvendo a venda de produtos ou mercadorias de seus clientes, chamados de empresas representadas. Esta mediação envolve de um lado as empresas representadas, indústrias e/ou empresas dedicadas ao comércio atacadistas, e de outro lado seus clientes, outras empresas atacadistas ou varejistas. Dessa forma

cabe ao representante comercial fazer esta ligação, de modo a aumentar o número de negócios entre elas.

A legislação que regulamenta a atividade dos representantes comerciais estabelece uma série de obrigações, tanto para o representante como para as empresas representadas, dentre elas citamos:

a) não deve haver subordinação entre o representante comercial e a empresa representada, devendo o representante comercial possuir autonomia para o exercício de suas atividades. Ou seja, o representante é uma empresa independente.
b) as atividades de representação comercial podem ser prestadas tanto por pessoas físicas (autônomos) como por pessoas jurídicas (empresa).

Existe um fator de grande importância que envolve o representante comercial, e não permite que este atue meramente como um vendedor, mas como o responsável pelo produto que representa e comercializa. Sendo assim, uma empresa de representações deve ser ágil e objetiva, com os dados das representadas e de seus produtos sempre atualizados e organizados.

Tais informações são manipuladas de diversas formas, por diferentes setores e profissionais, muitas vezes simultaneamente. Portanto, além do armazenamento dos dados é fundamental manter a consistência e a segurança destes.

2.1.1. Setores e Objetivos

Uma empresa de representação comercial de pequeno porte está organizada e dividida em diversos campos de atuação, que correspondem aos seguintes:

a) setor de vendas – composto por profissionais da área de vendas. Estes precisam ter carisma e boa apresentação, pois são responsáveis pelo contato direto com o cliente final. Os vendedores devem manter constantemente uma comunicação com o setor de “gestão de vendas” para que a troca de informações e auxílio técnico e logístico seja feito de forma ágil e eficiente. Cada vendedor possui uma zona de atuação onde será responsável pelo atendimento de todos os clientes e a captação de novas oportunidades de negócio.

b) setor de gestão de vendas – é formado por diversos profissionais que ocupam uma estrutura em um local fixo. Possui a responsabilidade de acompanhar o processo, do pedido do cliente à chegada da ordem de pedido na fábrica, da entrega ao cliente ao acompanhamento no que se refere a prazos, quantidade e qualidade. Também responde por todo o processo de logística de transporte e cumprimento dos pagamentos. Portanto, assim como o vendedor, o gestor de vendas necessita de velocidade na comunicação, juntamente com a armazenagem segura de todos os dados coletados diariamente.

c) setor administrativo – responsável por assuntos fiscais, jurídicos, financeiros e de recursos humanos. A administração deve ter um banco com os dados obtidos pelo setor de “gestão de vendas” já filtrado e com todas as informações acessíveis e guardadas de forma segura.

Esta subdivisão pode variar e é definida conforme os processos diários. Um dos setores fundamentais para qualquer empresa representante é o de “gestão de vendas”, pois é através dele que a empresa representada tem conexão com os vendedores, e estes com seus clientes.

3. **ABORDAGEM TECNOLÓGICA**

A maleta do representante de vendas estava ficando cada vez mais pesada. Atualmente no lugar da papelada habitual, ele carrega o laptop. Porém, por conta do peso, tamanho e pouca autonomia esta tecnologia já está sendo substituída aos poucos por dispositivos móveis, como *tablets* e *smartphones.*

Esse profissional necessita de *software* específico que organize e otimize todas as informações e tarefas pertinentes ao trabalho de representação. Isto torna a relação com a representada e com o cliente mais ágil e profissional. As informações manipuladas por uma empresa deste ramo possuem três principais formas de entrada, através do fornecedor, através do cliente ou ainda pelos funcionários da própria empresa. Por serem setores diferentes e indivíduos com conhecimentos distintos é primordial ter uma interface amigável e de fácil manuseio de forma que seja tudo altamente intuitivo para cada característica. Neste setor existe o contato com uma grande massa de dados e por isso é de suma importância possuir formas ágeis de interação com o seu armazenamento.

Neste ponto é preciso pensar em quais tecnologias serão aplicadas no desenvolvimento de um aplicativo voltado para este setor. Essas tecnologias devem fazer com que o desenvolvimento se torne mais simples e produtivo.

Por conta da intercomunicação de diversos setores da empresa e seus contatos é de suma importância pensar em uma forma onde o acesso à informação seja feito de forma fácil.

Ao se tratar de desenvolvimento de softwares, algumas etapas devem ser seguidas antes do inicio do projeto, de modo a entender como estruturar o processo de desenvolvimento, para isso essas etapas, derivadas do site Wikipédia (artigo Processo de desenvolvimento de *software*), serão brevemente descritas abaixo:

3.1. ANÁLISE DE REQUISITOS DE *SOFTWARE*

A extração dos requisitos é a primeira tarefa na criação. Embora todo cliente, acredite saber como o *software* deva se comportar é neste ponto onde o profissional consegue obter as informações necessárias para desenhar o projeto e apresentar ao cliente qual a real necessidade e como realmente o *software* deve funcionar, esta

tarefa requer habilidade para a percepção de brechas e formas de facilitar ao máximo o acesso às informações.

3.2. ESPECIFICAÇÃO

Parte onde será descrito e documentado precisamente o funcionamento e as abordagens do *software*. Especificações são extremamente importantes e detalhadas ao máximo, são elas que representam exatamente o que o cliente quer após o término do projeto.

3.3. CODIFICAÇÃO

Nesta etapa, após as especificações terem sido apresentadas e aceitas pelo cliente, é iniciada a implementação do software. Esta fase requer um tempo significativo, onde o profissional ou o grupo será exigido ao máximo, sendo normalmente a etapa de maior cobrança por agilidade e precisão, não podendo o profissional se dar ao luxo de cometer erros. Normalmente este processo possui um prazo mais amplo, contemplando correções e modificações no projeto, mesmo assim não podendo deixar de reservar um espaço maior para correção de erros não previstos.

a) testes e correções – é nesta parte do projeto que serão, estressantemente, testadas as situações que ocorrem no dia-a-dia do usuário final, mas que durante o processo de codificação não foram previstas. Desta forma normalmente são encontrados erros assim como adaptações e melhoramentos solicitados pelo cliente. Estes devem ser analisados e corrigidos antes da entrega do programa final.
b) documentação – neste ponto o *software* já deve estar pronto ou em fase de entrega, é aqui que é feita a documentação de cada parte do software descrevendo todos os pontos e seu funcionamento de forma detalhada, mostrando telas, menus e aspectos técnicos. Esta etapa deve ser feita em duas fases de forma a elaborar uma explicação simplificada para os usuários do sistema e uma documentação mais técnica para alterações futuras.

c) entrega – após apresentar os testes e o software ao cliente é feita a entrega. Para cumprir os passos do desenvolvimento de software, descritos, temos ainda que analisar as necessidades de todos os processos que envolvem o programa e optar pela melhor forma de aplicar esses conceitos. Assim temos: local (*desktop*), *web* (*Internet*), móvel (dispositivos móveis) ou ainda a combinação de duas ou mais dessas tecnologias.

Tendo em vista o desenvolvimento de um *software* bem específico com foco em representações comerciais é necessário buscar o máximo de informações possíveis sobre esta área de atuação. De forma a analisar todos os processos que estão envolvidos na manipulação destes dados, seja por parte do cliente, das representações ou do cotidiano da própria empresa.

A análise apresentou três focos principais:

a) As solicitações feitas aos fornecedores, essas comumente possuem padrões próprios fornecidos por cada empresa representada, devem ser estabelecidas de formas diferentes utilizando tipos diversos de comunicação, tais como: fax, e-mail, telefone, sms e correspondências;

b) Clientes, os pedidos feitos são uma parte fundamental para o sucesso neste setor, facilitar e se adaptar as preferências de cada cliente, não é fácil. Por isso é imprescindível oferecer muitas possibilidades de comunicação, como: presencial (através de um vendedor), fax, cadastro online (através de sites ou chats com vendedores), telefone (através de atendentes e vendedores) ou ainda por e-mail (através de um e-mail central que deve ser dividido por setores e encaminhado ao funcionário responsável).

c) Controle interno, o financeiro e Recursos Humanos são setores determinantes da empresa. O controle desta área deve abranger todos os funcionários, suas funções e responsabilidades, assim como controlar a segurança com acesso específico de acordo com as funcionalidades do sistema que serão responsáveis.

Esta interatividade entre a empresa, clientes e fornecedores, deixa claro que a utilização de uma solução que combine diversas tecnologias é o ideal. Por tanto com um foco concentrado em soluções online desenhadas para um acesso fácil e

dinâmico. Para abranger essas situações, a aplicação quando utilizada na empresa terá acesso local tornando-a extremamente rápida e produtiva, por outro lado, quando o acesso for externo existe a facilidade de comunicação através da *web*.

Trabalhar com a *web* tornou-se mais fácil, pois com diversos dispositivos em expansão tecnológica já é possível encontrar em abundância comunicação com a *Internet* através do sistema 3g, já oferecido por todas as operadoras de telefonia celular, em diversos desses aparelhos móveis tais como celulares e *tablets*, tendo a possibilidade de acesso em qualquer lugar que estejam. Hoje com a tecnologia *wireless* totalmente disseminada, é possível encontrá-la inclusive em locais públicos como aeroportos, praças e etc., os aparelhos que a dispõem tem a possibilidade de acesso facilitada. Já máquinas locais têm este acesso simplificado através de qualquer tipo de conexão com a *Internet* seja banda larga, sistema 3g ou qualquer outra tecnologia aplicada.

Para atender a todas essas especificações é necessário escolher uma forma apropriada para o desenvolvimento da aplicação. O fato de optar por linguagens mais difundidas, onde muitos profissionais já a dominem, trás à solução uma forte redução de tempo e gastos. Para tal e pensando em atender todos os requisitos encontrados foi escolhida a linguagem de programação *Java*.

4. **TECNOLOGIAS**

Neste capítulo serão abordadas as formas e tipologias tecnológicas necessárias para a informatização das empresas de representações comerciais.

4.1. *JAVA*

Java é uma linguagem de programação orientada a objeto desenvolvida na década de 90, pela empresa *Sun*. Teve seu maior reconhecimento no ano de 1995 quando foi lançada uma versão projetada para se mover por meios das redes de dispositivos heterogêneos, tais como a *Internet*. Atualmente é utilizada pelas principais empresas do mundo, assim como é suportada por mais de 3,0 bilhões de dispositivos que abrangem desde celulares, computadores de bordo até mesmo eletrodomésticos como geladeiras. Esse alcance de mercado atrai cada vez mais profissionais capacitados para a área, tornando esta tecnologia muito popular (Wikipédia – Java Linguagem de Programação).

Portanto, com a grande quantidade de profissionais disponíveis fica mais fácil obter mão-de-obra especializada a um custo menor. Também, é possível encontrar diversas linhas de pesquisas e projetos já elaborados que vem a servir como exemplo e acabam por facilitar o processo de codificação.

Sua heterogênia abrange tantos meios que permite a manipulação de um número impressionante de dispositivos. Possui ferramentas adequadas para uma melhor adaptação com cada tipo de equipamento, estas estão distribuídas e disponíveis em diversas bibliotecas que fazem a conexão entre a linguagem e os mais diversos dispositivos.

Com a crescente utilização com foco em *web,* trás o benefício de mobilidade, podendo ser acessada de qualquer dispositivo que tenha conexão com a *Internet*, seja um *smartphone* ou até mesmo as atuais smartTV (televisões com funcionalidades extras como acesso a internet). A grande maioria desses aparelhos já dispõe da máquina virtual *Java* pré-instalada, sendo possível a comunicação com qualquer sistema desenvolvido para a linguagem *Java*.

4.2. *JAVASCRIPT*

Com o advento do *JavaScript* junto à tecnologia AJAX (*Asynchronous JavaScript and* XML), as aplicações ficaram cada vez mais dinâmicas e com opções multimídia avançadas. Esse progresso permitiu que a chamada *web* 2.0 viesse à tona. Então, com o auxílio do *JavaScript* novas tecnologias começaram a se destacar, dentre elas o *Adobe Flex*, o AJAX e mais recentemente o GWT (*Google Web Toolkit*). Todas elas voltadas a facilitação e aprimoramento a experiência do usuário ao acessar um site, essas melhorias estão direcionadas, em grande parte, para as interfaces e comunicações assíncronas com o servidor (Wikipédia – JavaScript).

4.3. *FRAMEWORKS & TOOLKITS*

Um *framework* é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica. Também, poderá prover uma funcionalidade específica, por configurações personalizadas, durante a programação de um aplicativo (Wikipédia – Framework).

Ao contrário das bibliotecas, o *framework* dita o fluxo de controle completo da aplicação. Segundo Fayad e Schimidt (1997) este processo é chamado de Inversão de Controle *framework* que é “um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um sistema ou subsistema”.

O *framework* não se trata de um *software* executável, mas sim de um modelo de dados para uma função única, e é composto de um conjunto de classes implementadas em uma linguagem de programação específica, usadas para auxiliar o desenvolvimento de *software*. O modelo *framework* atua onde há funcionalidades em comum as várias aplicações.

Especificamente em orientação a objeto, o *framework* é um conjunto de classes com o objetivo de reutilização de arquitetura de *software*, provendo uma solução para um problema específico.

A biblioteca se diferencia de um modelo de *framework*, pois está direcionada apenas para oferecer funcionalidades específicas, sem definir a reutilização de uma

solução de arquitetura e, além disso, o *framework* poder ser composto por um conjunto de bibliotecas.

O framework é uma ferramenta avançada que proporciona o reuso de métodos e até bibliotecas inteiras. Ele não apenas reutiliza parcelas de códigos, ou uma porção de classes, consegue também o reaproveitamento de todo um design genérico para determinadas aplicações.

Um exemplo típico do uso desta ferramenta é um *framework* para gerar janelas de uma interface gráfica (comum em aplicações *Java*), onde terá sempre um *Jframe*, com botões, barras de rolagem, etc. Aplicar um *framework* para esta tarefa garante que os *software*s que o usam terão sempre uma interface uniforme, consistente e de fácil intercomunicação. Isso poupa o programador da reconstrução de um desenho que tende a se repetir por diversas vezes no código.

Frameworks são de grossa granularidade, logo são menos reutilizáveis. Em geral, eles são usados em determinadas aplicações específicas, e podem ser encontrados prontos ou ainda serem montados da maneira que o programador quiser até mesmo tendo como base um *framework* já existente (MELO, 1998).

Atualmente são duas as formas principais de interpretar os *frameworks*, “caixa-transparente” e “caixa-preta”. *Framework* caixa-transparente é aquela que o programador pode customizar estendendo e implementando subclasses para atender suas necessidades. Já no *framework* caixa-preta, você não enxerga o código, nem tem como acessá-lo apenas utilizar seus objetos. Os *frameworks* caixa-transparente são complicados de empregar, pois necessitam que o programador tenha mais experiência e um profundo conhecimento da plataforma escolhida, porém é uma ferramenta muito poderosa. Em contrapartida, o caixa-preta é de fácil utilização, pois não exige experiência com a linguagem e nem mesmo com a ferramenta, bastando o emprego dos métodos já disponibilizados, tornando-o mais específico.

Toolkits, ou “kits de ferramentas”, são aplicações em alto nível, geralmente baseados em *framework*s caixa preta. Foram desenvolvidos para que os programadores possam construir aplicações de forma mais simples, apenas customizando ou interligando objetos de um *framework*. Um exemplo típico é um *toolkit* para criação de interfaces gráficas.

O *toolkit* não programa a parte gráfica (design) em suas classes e subclasses, enquanto o *framework* na maioria das vezes já vem com uma implementação, mesmo que básica, que contem modelos já implementados que servem de base para o desenvolvedor, sendo possível utilizar da mesma forma que já está previamente desenvolvido ou é possível fazer alterações personalizando de acordo com a necessidade do projeto. Desta forma pode ser visto que o grau de acoplamento se torna maior nos *framework*, por já conter componentes prontos a serem utilizados.

O uso de *toolkits* e *framework*s é uma forma de facilitar e agilizar o trabalho do programador, tornando mais fácil e produtivo o desenvolvimento de projetos. Seu uso já é tomado como rotineiro, uma vez que uma grande parte das empresas de *software* está adotando como padrão de desenvolvimento.

4.4. INTRODUÇÃO AO GWT

Atualmente, criar aplicativos para a *web* é um processo com alta incidência de erros. Os desenvolvedores podem passar até 70% do tempo trabalhando para contornar peculiaridades de cada navegador. Além disso, a criação, a reutilização e a manutenção de grandes bases de código *JavaScript* e componentes AJAX é difícil e delicada. O *Google Web Toolkit* (GWT) facilita essa tarefa permitindo que os desenvolvedores construam e mantenham aplicativos *front-end* complexos em *Java*Script, mas com ótimo desempenho na linguagem de programação *Java* (Google Developers).

O *Google Web Toolkit* é um *toolkit* de código-fonte aberto permitindo aos desenvolvedores codificarem aplicativos com tecnologia AJAX em linguagem de programação *Java*. O GWT, como é comumente chamado, suporta programação cliente-servidor, e *debuging* em qualquer IDE *Java*. Exceto por algumas bibliotecas nativas, todo o fonte *Java* pode ser construído em qualquer plataforma com o GWT que possua o *Ant*. Atualmente está registrado sob a Licença Apache versão 2.0, tornando-o uma opção robusta e economicamente viável (Google Developers).

Como um bom toolkit enfatiza a reutilização, tornando-se uma solução eficiente para os desafios recorrentes do AJAX, ou seja, chamadas assíncronas de

procedimentos remotos, gerenciamento de históricos, favoritos, internacionalização e portabilidade entre os mais diversos navegadores (Google Developers).

4.4.1. Compilação e Debugging

Durante a compilação, o GWT transformará o código fonte desenvolvido em *Java*, em arquivos independentes e otimizados em *JavaScript*, montando compilações, chamadas de permutações, separadas por navegadores e idiomas, dando a liberdade para o desenvolvedor optar em quais navegadores e em que idiomas o sistema deve executar.

Para cada navegador adicionado é acrescentada uma etapa, que gera uma carga extra no processo de compilação, neste processo o compilador se encarrega de fazer permutações separadas e em paralelo para cada idioma e navegador, otimizando com as particularidades de cada. Todo este processo é separado em threads distintos, para aproveitar ao máximo os computadores que possuam múltiplos núcleos ou múltiplos processadores, entretanto como a memória é de uso comum é compreensível que seja exigido uma quantidade maior.

Diferente dos *minifiers JavaScript* que funcionam somente em nível textual, o compilador GWT executa rotinas de análises mais abrangentes e otimizações em todos os fontes desenvolvidos, produzindo um novo código *JavaScripts* que tem como foco a otimização de rotinas que auxiliam ao carregar e executar mais rapidamente independente do navegador. Durante o processo de compilação o GWT elimina códigos em desuso com segurança, removendo classes, métodos, campos e mesmo parâmetros de métodos que não estejam sendo utilizado, para garantir que o script compilado seja o menor e mais performático o possível. A compilação permutada permite que sejam mantidas as abstrações e a modularidade necessárias para o desenvolvimento, sem prejudicar o desempenho durante o tempo de execução (Run Time).

Para o *debugging*, durante o processo de codificação, o GWT fornece o “*Host mode*”, que proporciona uma espécie de pré-compilação que em conjunto com sua IDE (*Integrated Development Environment*) simula o comportamento do código como se tivesse ocorrido a compilação. Deste modo o desenvolvedor tem a liberdade de testar seus métodos e procedimentos de forma prática de dentro da própria IDE, dando um ganho de produtividade (Wikipédia – Debugger).

Entretanto, como dito anteriormente, se trata apenas de uma pré-compilação, sendo que nem todo o código é convertido. Isso pode gerar diferenças na forma de execução e principalmente na velocidade de execução de alguns métodos, uma vez que neste modo a conversão de código é realizada em tempo de execução, trazendo um custo agregado ao desempenho, e não são feitas as otimizações nem limpezas de códigos.

4.4.2. Compatibilidade

Tanto a J2SE (*Java* 2 *Standard Editon*) como a J2EE (*Java* 2 *Enterprise Edition*) são muito extensas, e muitas de suas classes dependem de funcionalidades não disponíveis em um navegador web. Por isso, GWT suporta apenas um subconjunto da plataforma *Java*, contido nos pacotes *Java.lang* e *Java.util*.

As versões destas classes emuladas pelo GWT irão geralmente se comportar da mesma forma como elas se comportam em *Java*, salvo algumas poucas exceções. Por exemplo, a sintaxe de expressões regulares em *Java* não é exatamente a mesma das expressões regulares em *JavaScript*. Portanto, ao invocar métodos que usam expressões regulares da classe *String* (por exemplo, os métodos "*replaceAll*(*String, String*)" e "*split*(*String*)"), tenha certeza que as expressões regulares utilizadas têm o mesmo significado tanto em Java como em JavaScript (Google Developers).

À medida que a aplicação é codificada, é necessária certa precaução ao usar no código *Java* somente as características da linguagem e as bibliotecas de classe suportadas pelo GWT.

Para o auxílio na função de encontrar incompatibilidades, quando o código roda no modo "*Hosted*", ele é verificado comparativamente a uma biblioteca de emulação da JRE (*Java Runtime Enviroment*), que captura os principais usos de funcionalidades no momento em que a aplicação é executada.

UIs (*User Interfaces*) GWT são compostas de *widgets* e *panels. Widgets* que fornecem os modelos para elementos da interface comumente usados, como botões, caixas de texto, e árvores. *Panels*, como *DockPanel*, *HorizontalPanel* e *RootPanel* que servem como containers para os *widgets*, sendo usados para a definição do layout das páginas (Google Developers).

Widgets e *Panels* funcionam da mesma forma em todos os navegadores. Ao usá-los, elimina-se a necessidade de escrever código específico para cada tipo de navegador. O *toolkit* do GWT possui um rico conjunto de *widgets*, mas não é necessário ficar limitado a eles, há várias formas de criar *widgets* customizados. Por padrão, aplicações GWT usam o tema *Standard*, o que pode ser facilmente alterado para um mais apropriado em cada aplicação. Também é possível encontrar uma boa variedade de bibliotecas online de *widgets* e temas escritos por terceiros (Google Developers).

Cada *widget* está relacionado a uma ou mais regras de CSS (*Cascading Style Sheets*) que lhes provêem estilos diversos. Cada tipo de *widget* possui um nome de estilo *default*, derivado do nome de sua classe. Por exemplo, o *widget Button* que possui o nome de estilo *gwt-Buttton*, então por padrão, todos os *widgets Button* herdam o estilo da classe CSS do *gwt-Button*. Isso facilita, pois para fazer uma alteração em todos os botões da aplicação, como alterar o tamanho da fonte, basta modificar o estilo que será aplicado a todos os que herdam do mesmo (Google Developers).

4.4.2.1. Lado Cliente

O coração do GWT é um compilador que converte código *Java* em *Java*Script, transformando sua aplicação *Java* numa aplicação equivalente em *Java*Script. Porém, devido às limitações do *Java*Script e do navegador na qual ele é executado, haverão certos aspectos do *Java* que o GWT não suportará. É necessário manter atenção conforme o código é escrito (Google Developers).

O GWT suporta boa parte da semântica de Java 1.5. É possível usar quaisquer dos tipos intrínsecos da linguagem, bem como as classes de sua própria aplicação. Porém, JavaScript não suporta inteiros de 64 bits, nem garante a precisão de operações de ponto flutuante (números decimais) com os tipos primitivos. Portanto deve-se evitar o uso de variáveis do tipo long em operações de alta complexibilidade, contudo existem meios de contornar esta situação, um desses meios é o desenvolvedor implementar a classe BigDecimal (Google Developers).

4.4.2.2. Lado Servidor

Toda aplicação GWT roda como código *JavaScript* no navegador *web* do usuário final. Frequentemente é preciso criar mais do que aplicações *stand-alone* no lado do cliente. Será necessário que a aplicação se comunique com o servidor *web*, enviando pedidos e recebendo atualizações. Aplicações *web* tradicionais, precisam recarregar a página HTML inteira toda vez que elas se comunicam com o servidor. Em oposição, aplicações AJAX removem esse fardo da UI, fazendo chamadas assíncronas para o servidor, trocando com ele apenas os dados em si. Isso permite que as UIs sejam mais leves e rápidas, ao mesmo tempo em que diminui os requisitos de largura de banda e de carga no servidor (Google Developers).

A comunicação com o servidor é feita via RPC (*Remote Procedure Calls*), que é uma tecnologia popular para a implementação do modelo cliente-servidor. Nessa comunicação é executada uma chamada de procedimento remoto iniciada pelo cliente, enviando uma mensagem para um servidor remoto para que esse execute um procedimento específico. Logo, uma resposta é retornada ao cliente, desta forma o programador não se preocupa com detalhes da interação remota, tornando-a transparente (Google Developers).

A *framework* GWT RPC facilita para os componentes do cliente e do servidor da aplicação *web* trocarem entre si objetos *Java* via HTTP. Os serviços GWT são baseados na conhecida arquitetura *Java Servlets*. No código-cliente, as classes *Proxy* geradas automaticamente são usadas para realizar chamadas aos serviços. O GWT lidará com a serialização dos argumentos das chamadas, bem como dos respectivos valores de retorno (Google Developers).

É importante salientar que, de acordo com o guia do desenvolvedor GWT, “serviços GWT não são a mesma coisa que os *Web Services* baseados em SOAP (*Simple Object Access Protocol*) ou REST (*Representational State Transfer*)”. Eles foram criados para ser simplesmente uma forma mais leve de transferir dados entre o servidor e a aplicação GWT executada no cliente (Google Developers).

Todas as chamadas RPC GWT são assíncronas, assim elas não bloqueiam a UI enquanto aguardam o retorno do servidor. Essa abordagem traz benefícios com relação às chamadas síncronas:

a) UIs síncronas: As *engines JavaScript* são geralmente *single-threaded*, e chamadas síncronas provocam travamentos na página enquanto ela aguarda a chamada completar. Se a rede estiver lenta, ou o servidor não

estiver respondendo, isto pode arruinar a experiência para o usuário final. Chamadas síncronas ao servidor vão contra os princípios básicos de AJAX.

b) Múltiplas chamadas assíncronas: As chamadas podem ser feitas ao mesmo tempo. Porém os navegadores tipicamente limitam o número de conexões em duas por vez, limitando o paralelismo que pode ser feito usando chamadas assíncronas. É possível iniciar outras tarefas enquanto se aguarda por uma chamada pendente. Por exemplo, pode-se simultaneamente montar o restante da tela, ao mesmo tempo em que os dados estão sendo recuperados do servidor. Isto encurta o tempo total em que o usuário teria de esperar para interagir com a interface.

O lado do servidor não é colocado em *JavaScript*, por este motivo é possível utilizar todo o potencial da linguajem *Java*. Desta forma o servidor recebe as chamadas contendo os objetos e requisições, vinda do cliente via RPC, processa através de métodos Java e retorna da mesma forma. Assim, o RPC GWT se responsabiliza pela tradução entre as estruturas de dados.

5. **BANCO DE DADOS**

Esse capítulo aborda as funções do banco de dados e estabelece os motivos que levaram a escolha do banco que será utilizado na aplicação. Para esta escolha será levado em consideração o desempenho, facilidade de comunicação, bibliotecas disponíveis, otimizações e robustez.

5.1. HIBERNATE

Hibernate é um software livre de código aberto distribuído com a licença LGPL (*Lesser General Public License*).

O Hibernate não é um banco de dados, e sim um *framework* para o mapeamento objeto-relacional desenvolvido em *Java*. Foi planejado para facilitar o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante ao uso de notações em arquivos (XML (*Extensible Markup Language)*) para estabelecer esta relação (Wikipédia - Hibernate).

O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo de orientação a objeto, que precisa trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGBDs (Sistema de Gerenciamento de Banco de Dados)) (Wikipédia - Hibernate).

Sua principal característica é a transformação de classes *Java* em tabelas de dados. O Hibernate gera os códigos SQL (*Structured Query Language*) para fazer a comunicação com o banco, mantendo o programa portável para quaisquer bancos de dados SQL, deixando ao desenvolvedor a liberdade de escolha.

O Hibernate pode ser utilizado em aplicações *Java stand-alone* ou em aplicações *Java* EE (*Enterprise Edition*), utilizando *servlet* ou sessões *Enterprise Java Beans*. No caso de aplicações construídas para serem executadas em servidores de aplicação, o gerenciamento das transações é realizado segundo o padrão JTA (*Java Transaction API*). Já nas aplicações *stand-alone*, o programa delega o tratamento transacional ao driver JDBC (*Java Database Connectivity*).

5.2. BANCOS DE DADOS

Bancos de dados são softwares responsáveis por gerenciar e armazenar informações que se relacionam entre si. Após se tornarem mais confiáveis e robustos, são utilizados como a principal peça dos sistemas de informação da computação moderna (Wikipédia – Sistema de Gerenciamento de Banco de Dados).

O primeiro Sistema Gerenciador de Banco de Dados (SGBD) comercial surgiu no final de 1960 com base nos primitivos sistemas de arquivos disponíveis na época, os quais não controlavam o acesso concorrente por usuários ou processos. Os SGBDs evoluíram desses sistemas de armazenamento em arquivos no disco local, criando novas estruturas de dados com o objetivo de armazenar informações. Com o tempo, os SGBDs passaram a utilizar diferentes formas de representação, ou modelos de dados, para descrever a estrutura das informações contidas em seus bancos de dados (Wikipédia – Sistema de Gerenciamento de Banco de Dados).

Atualmente, os modelos de dados normalmente utilizados pelos SGBD’s são: modelo hierárquico, modelo em redes, modelo relacional (amplamente usado) e o modelo orientado a objetos.

O modelo hierárquico foi o primeiro a ser reconhecido como um modelo de dados. Nesse modelo de dados, as informações são estruturadas em hierarquias ou árvores. Os nós das hierarquias contêm ocorrências de registros, onde cada registro é uma coleção de campos (atributos), cada um contendo apenas uma informação. O registro da hierarquia que precede a outros é o registro-pai, os outros são chamados de registros-filhos (TAKAI, O.K.; ITALIANO, I.C.; FERREIRA, J.E 2005).

O modelo em redes surgiu como uma extensão ao modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registro estivesse envolvido em várias associações. No modelo em rede, os registros são organizados em grafos onde aparece um único tipo de associação (*set*) que define uma relação 1:N entre dois tipos de registros, proprietário e membro.

O modelo relacional apareceu devido às necessidades de aumentar a independência de dados nos sistemas gerenciadores de banco de dados, prover um conjunto de funções apoiadas em álgebra relacional para armazenamento e recuperação de informações. O modelo relacional revelou-se ser o mais flexível e adequado ao solucionar os vários problemas que se colocam no nível da concepção e implementação da base de dados. A estrutura fundamental do modelo relacional é a relação (tabelas). Uma relação é constituída por um ou mais atributos (campos) que traduzem os tipos de dados que serão armazenados. Cada instância do esquema (linha) é chamada de tupla (registro). O modelo relacional não tem caminhos pré-definidos para o acesso aos dados como nos modelos que o precederam. O modelo relacional implementa estruturas de dados organizadas em relações (SANTOS 2010).

Os bancos de dados orientados a objeto começaram a se tornar comercialmente viáveis em meados dos anos 80. A motivação para seu surgimento está em função dos limites de armazenamento e representação semântica impostas no modelo relacional. A habilidade para criar os tipos de dados necessários, é uma característica das linguagens de programação orientadas a objeto, que foi trazida para os bancos, contudo, estes sistemas necessitam guardar representações das estruturas de dados que utilizam no armazenamento permanente. O termo “modelo orientado a objetos” é usado para documentar o padrão que contém a descrição geral das facilidades de um conjunto de linguagens de programação orientadas a

objeto e a bibliotecas de classes que podem formar a base para o Sistema de Banco de Dados (TAKAI, O.K.; ITALIANO, I.C.; FERREIRA, J.E 2005).

Existem diversas empresas que implementam bancos de dados, cada uma com suas características próprias e um modelo próprio. Dessas empresas com certeza as mais famosas são a *Microsoft* e a *Oracle*, ambas contendo ferramentas excelentes, porém pagas. Entretanto existem aplicações de uso livre que estão disponíveis a qualquer usuário de forma gratuita, dentre esses bancos o MY-SQL é o mais famoso, mas não é o único, o *Firebird* e até uma solução da própria *Oracle*, o *Oracle Express*, que apesar de suas limitações também pode ser utilizada tendo em vista pequenas aplicações.

5.3. MS-SQL

Microsoft SQL Server é um servidor de banco de dados relacional, em parceria com a Sybase em 1988 e inserido como produto complementar do Windows NT. Ao fim da parceria a Microsoft deu continuidade ao produto criando o atual MS-SQL Server. Assim como os demais sua principal função é armazenar e recuperar dados, conforme solicitado por outras aplicações tanto em aplicações locais como em aplicações remotas, através de uma rede local ou Internet.

Existem diversas edições diferentes do *Microsoft* SQL *Server*, cada uma destinadas a públicos diferentes e para diferentes cargas de trabalho (variando de pequenas aplicações que armazenam e recuperam dados no mesmo computador, a aplicações de grande porte com milhares de acessos simultâneos). Fiel ao seu nome e princípios, o *Microsoft* SQL *Server* suporta os principais idiomas de consulta como T-SQL (Transact - *Structured Query Language*) e SQL ANSI (*American National Standards Institute*).

O MS SQL *Server* tem seu gerenciamento baseado em diretivas que possui suporte a uma estrutura para o gerenciamento de uma ou mais instâncias do MS-SQL *Server*, essa funcionalidade foi adicionada em sua versão 2008. Através do LINQ (*Language Integrated Query*), permite que os desenvolvedores utilizem objetos para endereçar as consultas em relação aos dados usando uma linguagem de programação gerenciada, dando preferência e maior compatibilidade ao C# ou VB.NET (linguagens proprietárias da *Microsoft*), ao invés das declarações SQL.

Principais características do MS-SQL *Server*:

a) *Trigger;*

b) *Stored procedure;*

c) SQL *User Function;*

d) *Extended Stored Procedure;*

e) Suporte online, telefônico 24h.

5.4. FIREBIRD

Firebird é um sistema gerenciador de banco de dados. A Fundação *Firebird*SQL coordena a manutenção e desenvolvimento do Firebird, sendo que os códigos fonte são disponibilizados sob o CVS (*Concurrent Versioning System*) da *SourceForge*.

Baseado no código do *InterBase* da *Borland*, quando da abertura de seu código na versão 6.0. O *Firebird* é totalmente gratuito, pois não há limitações de uso, tamanho ou número de conexões, seu suporte amplamente discutido em listas e fóruns na Internet facilita a obtenção de ajuda técnica. O *Firebird* possui também uma versão *mobile* para *Android*, sistema operacional da *Google* presente na maioria dos dispositivos móveis (Firebird - Conheça o Firebird em 2 minutos).

Ele possui seu código aberto, portanto você pode utilizá-lo em qualquer tipo de aplicação, seja ela comercial ou não, sem ter um investimento financeiro alto focando apenas em mão de obra qualificada (Firebird - Conheça o Firebird em 2 minutos).

A tecnologia usada no Firebird tem mais de 20 anos, fazendo com que ele seja um produto confiável e estável.

Segundo o site Firebird - Conheça o Firebird em 2 minutos o *Firebird* é um SGBD completo e poderoso. Ele pode gerenciar bancos de dados com bom desempenho e baixa manutenção. Seus principais recursos são:

a) Suporte a *Stored Procedures e Triggers;*

b) Suporte a Transações compatíveis com ACID;

c) Suporte a Integridade Referencial;

d) Suporte a *Multi Generational Architecture;*

e) Baixo consumo computacional;

f) Suporte a linguagem nativa para PSQL *(Stored Procedures e Triggers);*

g) Suporte a acesso ao banco de dados: nativo/API*, dbExpress,* ODBC (*Open Database Connectivity*), OLEDB (*Object Linking and Embedding, Database), .Net provider,* JDBC nativo tipo 4 e outros;

h) Multiplataforma (sistemas operacionais: *Windows, Linux, Solaris, MacOS*);

i) Suporte a 64bits;

j) Suporte a *triggers* de conexão e transação;

k) Suporte a tabelas temporárias.

5.5. MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações ao redor do mundo (Wikipédia - MySQL).

O MySQL foi criado na Suécia por volta da década de 1980. Hoje seu desenvolvimento é de responsabilidade da *Sun Microsystems*, que em 2008 comprou em uma negociação bilhonária, entretanto no ano seguinte a *Oracle* compra a *Sun Microsystems* e todos o seus produtos, incluindo o MySQL.

O sucesso do MySQL não veio por acaso e deve-se em grande parte à fácil integração com a linguagem de programação PHP, incluído na maioria dos pacotes de hospedagem de sites da Internet, além de suportar *Unicode*, *Full Text Indexes*, replicação, *Hot Backup*, GIS (*Geographic information system*), OLAP (*Online Analytical Processing*) e muitos outros recursos que o tornam esse sistema um banco de dados eficiente.

Além disso possui características que o fortalecem, como informa a site Wikipédia - MySQL

a) Portabilidade (sendo multiplataforma);

b) Compatibilidade (drivers ODBC, JDBC e .NET e interface para diversas linguagens de programação)

c) Desempenho e estabilidade;

d) Suporte a controle transacional;

e) Suporte a *Triggers*;

f) Suporte a *Stored Procedures* e *Functions*;

g) Suporte a replicação;

h) É um Software livre com base na GPL (*General Public License*).

5.6. ORACLE

Uma combinação quase imbatível para a computação, baseada nos mesmos padrões desde 1995, o sistema de gerenciamento de banco de dados (SGBD) Oracle garante hoje o topo das estatísticas quando se fala de um banco de dados confiável e eficiente. Detentor de um desempenho invejável ele implementa rotinas que chegam a ser até 4 vezes mais rápido que a concorrência (SILVA 2006).

A *Oracle* tem apoiado a plataforma *Java* com entusiasmo desde seu surgimento em 1995. Hoje, oferece os ambientes de desenvolvimento e implantação de J2EE mais completos e produtivos disponíveis. Como membro executivo do *Java Community Process*, a *Oracle* participa de mais de 80 solicitações de especificações *Java* (JSRs, pelas iniciais em inglês), e gurus da *Oracle* atuam como líderes de especificações para importantes JSRs, como a API XQuery para *Java*, ligação de dados padrão e recurso de acesso a dados para J2EE, e Metadados de tempo de design para componentes JSF (*JavaServer Faces*).

A *Oracle* também está à frente de importantes projetos de criação de ferramentas para BPEL (*Business Process Execution Language*), JSF e EJB 3.0 (*Enterprise JavaBeans*), além de estar inclusa na *Eclipse Foundation*, uma comunidade que trabalha com código aberto e desenvolve um dos melhores IDE Java do mercado (SILVA 2006).

Assim como muitos outros o *Oracle* possui diversas distribuições que permite acesso a empresas de pequeno e grande porte. Dentre as diversas distribuições existe a *Express*, versão inicialmente planejada para universitários que estariam a aprender sobre o sistema, hoje tornou-se o produto de entrada da empresa, apesar de muitas limitações como capacidade de dados e funções mais otimizadas, a versão *Express* tem sido muito difundida por usuários de aplicações pequenas.

Outro produto da *Oracle* que veio a chamar a atenção de empresas de grande porte foi à distribuição na versão *Oracle* RAC (*Oracle Real Application Clusters*), que é uma das opções que a Oracle introduziu pela primeira vez com o Oracle 9i.

O RAC é hoje uma tecnologia usada por milhares de clientes em todos os setores, em todos os tipos de aplicação. O *Oracle* RAC fornece opções para dimensionar aplicações além da capacidade de um único servidor, isso permite que os clientes tirem proveito de hardwares padronizados de baixo custo para reduzir seu custo total de propriedade e fornecer um ambiente de computação redimensionável que suporte a carga de trabalho de suas aplicações. O *Oracle* RAC permite que o Banco de dados *Oracle* execute todos os tipos de aplicações corporativas de base em clusters, incluindo produtos empacotados conhecidos (como o *Oracle Applications*, *Peoplesoft*, SAP (*Systems, Applications, and Products*)), aplicações desenvolvidas internamente, que podem ser OLTP (*Online Transaction Processing*) ou Processamento de Transações em Tempo Real ou uma carga de trabalho mista.

Graças a essas diversas versões a seu desempenho extraordinário o Oracle mesmo sendo uma ferramenta paga, possui grande popularidade e abrangência, principalmente no mercado corporativo.

6. **CONSIDERAÇÕES FINAIS**

Durante a explanação foram apresentadas diversas soluções, que devem ser utilizadas no processo de desenvolvimento do sistema estabelecido, para essas cada qual se encaixa em determinadas circunstâncias, tendo seus recursos melhor utilizados.

Para o desenvolvimento deste sistema foram escolhidas uma relação de tecnologias tendo em vista o melhor desempenho possível e facilidade na análise, desenvolvimento e implementação. Por tanto, como base de todo o projeto, a linguagem de programação *Java* foi escolhida.

Foi optado pelo *Java* por sua abrangência atual no mercado, o fácil acesso a informações técnicas (API) e a quantidade de profissionais que a dominam, além de possuir diversos *frameworks* e *toolkits* disponíveis, o que torna sua implementação mais fácil, rápida e robusta.

Em conjunto com o *Java*, uma vez que o sistema será baseado em *browsers*, será utilizada a linguagem *JavaScript*, tendo foco que as novas aplicações *web*, a dita *web* 2.0, já se baseiam no uso de AJAX, que possui o *Javascript* como seu principal componente, tornando o sistema mais dinâmico, atraente, flexível e atual.

Para a ligação entre ambas as linguagens foram analisadas algumas ferramentas como *frameworks* e *toolkits*, dentre elas os *frameworks Vaddin* e *Ext* se destacaram, pois ambos cumprem todos os requisitos necessários para a construção de um sistema com estabilidade, desempenho, acessível e apresentando um visual simples, intuitivo e atrativo.

O *Vaddin* e o *Ext*-GWT executam em conjunto com o GWT (*Google Web Toolkit*) e são programados diretamente em *JAVA*, podendo ter instruções *HTML* e *JavaScript* (em situações adversas) através de *tags* de comando, fazendo posteriormente, durante a compilação e a permutação do projeto, a tradução para AJAX, permitindo compatibilidade com a maioria dos navegadores disponíveis para os principais sistemas operacionais.

Para esta aplicação será utilizado o *Ext*-GWT (conhecido por GTX), que entre suas muitas características possui uma ampla API de informações sobre cada componente, além de aplicações demonstrativas e métodos semelhantes à já conhecida tecnologia *desktop swing*, tornando mais simples e intuitivo aos

programadores mais experientes. Essas características auxiliam em todo o processo de análise e desenvolvimento do *software*, tornando esta a melhor opção.

Para concluir a integração das tecnologias e sendo este um dos itens de maior importância em qualquer projeto de *software*, a escolha do banco de dados tem de ser feita de forma cuidadosa, avaliando as possibilidades e necessidades do cliente e desenvolvedores envolvidos e foi por este motivo que o *Oracle* se apresentou como melhor opção.

Ao escolher o *Oracle* foram levados em consideração fatores como compatibilidade com a linguagem (sendo possível estabelecer métodos internos ao banco programados diretamente em *Java*, linguagem já utilizada no projeto), facilidade no manuseio, levando em consideração que ele atua com uma linguagem própria já bem difundida e tendo um desempenho de alto nível, o PL/SQL e que além disso aceita interações em linguagem SQL padrão além de aceitar comando *Java* nativamente, portabilidade entre plataformas, sendo possível sua implementação nos sistemas operacionais mais utilizados por empresas, possui instâncias tanto para *Windows* (exigindo uma maior utilização de memória), quanto em *Linux* (este tendo uma distribuição da própria empresa, totalmente otimizada o que o torna mais rápido e eficiente), formas de acesso, integridade dos dados e o ponto forte deste banco, que o faz um destaque dentre os outros estudados, seu desempenho no acesso às informações nas tabelas de dados.

Para facilitar a integração do banco de dados e a linguagem de programação, foi utilizado o intermediador *Hibernate*. Sendo esta uma poderosa ferramenta que traz como seus principais benefícios a comunicação com múltiplos bancos de dados, assim caso futuramente a base de dados venha a ser migrada para outro banco, não será necessário a reescritura de código por parte dos desenvolvedores, apenas a estruturação da nova instância de Banco de dados através de um DBA (*Data Base Administrator*). Também, possui uma linguagem própria o HQL, que acessa diretamente os objetos do projeto e traduz de forma eficaz para a comunicação com o banco de dados através do SQL, esse processo é feito em tempo de execução e de forma transparente tornando o desenvolvimento mais simples.

Durante o processo de produção deste trabalho foi analisado que ao combinarmos as tecnologias descritas acima com uma boa análise conceitual de todo o processo produtivo de uma empresa da área de representações, é possível

desenvolver uma aplicação robusta e otimizada para os desafios de um representante comercial, facilitando ao máximo o processo de comunicação setorial e o processo de relacionamento com o cliente final.

As tecnologias descritas foram testadas e analisadas, em geral com foco em sua performance, até chegar a uma combinação que agradasse em questões de desempenho, facilidade de codificação e agilidade no processo de produção.

Todo o processo de análise e escolha das ferramentas foi feito visando os desafios dos diversos setores de uma empresa de representação comercial. Tendo em vista que este setor de mercado possui um diferencial frente a outras áreas do comercia, em relação a sua mobilidade.

Uma vez que seus representantes têm de estar a todo o momento em deslocamento para atendimento direto e personalizado de seus clientes e representados, é crucial que a ferramenta a ser desenvolvida supra situações de deslocamento a longas distâncias através de diversos meios. Como não é claro que o local a onde o atendimento será feito possuirá acessibilidade a um meio de comunicação como a *Internet*, é preciso estabelecer outras formas de armazenagem e envio de dados.

A combinação escolhida para o desenvolvimento deste *software* não se prende ao simples fato de estar online, apesar de ser totalmente estruturada para uma interação com a *web*, é possível usufruir de outros benefícios da tecnologia para driblar situações de atendimentos em locais sem acesso ao meio.

Como estas tecnologias já foram pensadas e direcionadas para atender a dispositivos móveis tais como *smartphones* , *tablets* e outros, é visto que qualquer dispositivo capaz de suportar a tecnologia da HTML 5 está apto a executar o sistema, tanto online como *off-line*.

A combinação de todas as ferramentas vistas com o HTML 5 nos permite desenvolver meios que permitam ao cliente fazer algumas inteirações com o sistema, mesmo ele estando *off-line*. Interações como cadastro de novos clientes ou fornecedores, cadastro de pedidos, cálculos de preços e qualquer cadastro necessário para se obter um atendimento eficaz são feitos de forma simples, fácil e rápida, tendo o mesmo visual como se estivesse em um modo online.

Isso ocorre pois é possível fazer o armazenamento dessas informações dentro do dispositivo móvel em questão e desta forma quando o mesmo fizer uma

conexão com o meio enviar tais informações para o banco de dados (na cede da empresa), desta forma sincronizando os dados e mantendo todas as informações atualizadas. Isso é possível graças às tecnologias de comunicação, como o 3g, que hoje atingem localizações mais afastadas dando um apenas um curto período onde o funcionário estará *off-line* e mantendo assim todas as informações sempre atualizadas.

É apresentado que o setor em questão carece de um *software* de alto nível que suporte as tecnologias fixas e móveis hoje já presentes em todos os locais. Também é apresentado que com as soluções adotadas é possível atingir esses objetivos.

REFERÊNCIAS BIBLIOGRÁFICAS

BASHAM, B.; SIERRA, K.; BATES, B. **Use a Cabeça! JAVA.** Rio de Janeiro: Alta Books 2009.

BASHAM, B.; SIERRA, K.; BATES, B. **Use a Cabeça! Servlets & JSP.** Rio de Janeiro: Alta Books 2008

CONFERE. **Conselhos Regionais dos Representantes Comerciais.** Disponível em: <http://www.confere.org.br/lei%204886.HTML>. Acesso em: 20/12/2011.

FAYAD, M. E., SCHMIDT, **D. C. Object-oriented Application frameworks.** Communications of the ACM, Vol. 40 1997.

FERNANDES, Lúcia. **Oracle para desenvolvedores.** Rio de Janeiro: Axcel Books do Brasil 2000.

FIREBIRD.ORG. **Conheça o Firebird em 2 minutos**. Disponível em: <http://www.firebirdnews.org/docs/fb2min_ptbr.html>. Acesso em: 03/01/2012.

GOOGLE. **Google Developers**. Disponível em: <http://developers.google.com />. Acesso em: 03/01/2012.

HIBERNATE. **Hibernate**. Disponível em: <http://www.hibernate.org/>. Acesso em: 03/01/2012.

MELO, Euclides Pinheiro de. **Designing Reusable Classes.** Santa Catarina: UFSC 1998.

MYSQL. **MySQL**. Disponível em: <http://www.mysql.com/>. Acesso em: 03/01/2012.

ORACLE. **Oracle**. Disponível em: <http://www.*Oracle*.com/br/products/database/index.html>. Acesso em: 03/01/2012.

REVISTA EMPRESA & DINHEIRO. **Como ser representante comercial**. Disponível em: <http://www.empresasedinheiro.com/como-ser-representante-comercial/>. Acesso em: 22/12/2011.

SANTOS, Nei Fábio Piedade dos. **Bancos de Dados II**. Itaberaba: Universidade Norte do Paraná 2010.

SEBRAE. **Representação comercial: definição e cuidados legais.** Disponível em: <http://www.mundosebrae.com.br/2010/03/representacao-comercial-definicao-e-cuidados-legais/>. Acesso em: 10/01/2012.

SILVA, Thiago Racca da. **Oracle e Java**. São Paulo 2006.

WIKIPÉDIA. **Debugger.** Disponível em: <http://en.wikipedia.org/wiki/Debugger>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Firebird.** Disponível em: <http://pt.wikipedia.org/wiki/Firebird_(servidor_de_base_de_dados)>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Framework**. Disponível em: <http://pt.wikipedia.org/wiki/Framework>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Hibernate**. Disponível em: <http://pt.wikipedia.org/wiki/*Hibernate*>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Java (Linguagem de programação)**. Disponível em: <http://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o)>. Acesso em: 03/01/2012.

WIKIPÉDIA. **JavaScript**. Disponível em: <http://pt.wikipedia.org/wiki/JavaScript>. Acesso em: 03/01/2012.

WIKIPÉDIA. **MySQL**. Disponível em: <http://pt.wikipedia.org/wiki/MySQL>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Oracle**. Disponível em: <http://pt.wikipedia.org/wiki/*Oracle*_(banco_de_dados)>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Processo de desenvolvimento de software**. Disponível em: <http://pt.wikipedia.org/wiki/Processo_de_desenvolvimento_de_software>. Acesso em: 03/01/2012.

WIKIPÉDIA. **Sistema de Gerenciamento de Banco de Dados**. Disponível em: <http://pt.wikipedia.org/wiki/Sistema_de_gerenciamento_de_banco_de_dados)>. Acesso em: 03/01/2012.

ZEIS, Chris; ZEIS, Ruel. **Oracle 11G para Leigos (for Dummies).** Rio de Janeiro: Alta Books 2009.