UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FACULDADE DE CIÊNCIAS EXATAS
CURSO DE SISTEMAS DE INFORMAÇÃO

HENRIQUE ARRUDA PELUZIO

DESENVOLVIMENTO DE APLICATIVO WEB:
Sistema de divulgação e comercialização de carros e motocicletas em Diamantina

Diamantina
2015

HENRIQUE ARRUDA PELUZIO

DESENVOLVIMENTO DE APLICATIVO WEB:
Sistema de divulgação e comercialização de carros e motocicletas em Diamantina

Trabalho de conclusão de curso de Graduação apresentado à Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Alessandro Vivas Andrade

Diamantina

2015

AGRADECIMENTOS

A Deus por ter me dado saúde e força para superar as dificuldades, por iluminar meu caminho e sempre me colocando nos trilhos para conquistar meus objetivos.

Ao meu pai Sebastião, à minha mãe Maura e minha irmã Érica, pelo amor, apoio e incentivo incondicional, por sempre acreditarem em mim.

À minha namorada Sheila com sua paciência suportando todas minhas manias e por sempre estar ao meu lado nos dias bons e ruins.

Ao meu orientador Alessandro Vivas Andrade pelo suporte, pelas suas correções, conselhos e incentivos.

E a todos que direta ou indiretamente fizeram parte da minha formação.

RESUMO

O objetivo deste trabalho foi a criação de um aplicativo web com interface simples e intuitiva de divulgação e comercialização de carros e motocicletas voltado para cidade de Diamantina. Os usuários podem, desta forma, anunciar suas intenções de vendas ou buscar por um veículo anunciado facilitando suas pesquisas na cidade de Diamantina e região além de ajudar em suas divulgações caso necessitem. Na implementação foi utilizado a linguagem de programação Java por possuir forte reuso de código, ser uma linguagem portável e possuir um amplo grupo de usuários em sua comunidade. Na camada de dados foi utilizado o mais famoso framework de persistência de dados no mundo, o Hibernate, junto ao sistema gerencial de banco de dados mais rápido e leve do mercado, o MySQL. O padrão de arquitetura utilizado foi o MVC por trazer escalabilidade, manutenibilidade e se adequar bem ao framework escolhido, o JSF. O JSF foi utilizado juntamente com o framework Primefaces, dedicado a camada visão da arquitetura MVC, dando uma interface limpa, simples e intuitiva à aplicação. O ambiente integrado para desenvolvimento de software utilizado foi o popular Eclipse em uma distribuição Linux (Ubuntu 14.04). Esta aplicação, por ser desenvolvida para a Web, foi necessário um servidor de aplicação embutido, desta forma foi escolhido o JBoss que é implementado completamente em Java e integra muito bem com o Hibernate. O JBoss também disponibiliza o JAAS para desassociar as aplicações dos controles de acesso a recursos do sistema, trazendo muito mais segurança à aplicação como um todo.

ABSTRACT

The objective of this work was to create a web application with simple and intuitive interface for the dissemination and marketing of cars and motorcycles facing city of Diamantina. Users can thus advertise their sales intentions or check for a vehicle advertised facilitating their research in the city of Diamantina region and beyond to help in their disclosures if they need. In the implementation we used the Java programming language because it has strong reuse code, is a portable language and have a broad user group in your community. In the data layer was used the most famous data persistence framework in the world, Hibernate, with the managerial system faster database and lightest on the market, MySQL. The architectural pattern used was the MVC for bringing scalability, maintainability and fit well to the chosen framework, JSF. The JSF was used along with Primefaces framework dedicated to layer view of MVC architecture, giving a clean interface, simple and intuitive application. The integrated environment for software development used was the popular Eclipse on a Linux distribution (Ubuntu 14.04). This application, to be developed for the Web, a built-in application server is required in this way was chosen JBoss that is completely implemented in Java and integrates nicely with Hibernate. JBoss also offers the JAAS to disassociate the application of access control to system resources, bringing more security to the application as a whole.

LISTA DE SIGLAS

JEE	-	Java Enterprise Edition
POO	-	Programação Orientada a Objetos
API	-	Application Programming Interface
JDBC	-	Java Database Connectivity
JSF	-	Java Server Faces
EJB	-	Enterprise JavaBeans
JAAS	-	Java Authentication and Authorization Service
XML	-	Extensible Markup Language
AJAX	-	Asynchronous Javascript and XML
SGBD	-	Sistema de Gerenciamento de Banco de Dados
JPA	-	Java Persistence API
ORM	-	Object Relational Mapper
MVC	-	Model View Controller
IDE	-	Integrated Development Environment
WWW	-	World Wide Web
HTTP	-	Hypertext Transfer Protocol
IETF	-	Internet Engineering Task Force
RPC	-	Remote Procedure Call

SUMÁRIO

1 INTRODUÇÃO

Pouco tempo atrás as empresa que possuíam um aplicativo web eram bem vistas com um diferencial perante suas concorrentes. Hoje as empresas que não possuem sequer uma página online estão caminhando em direção contrária à competitividade do mercado. Ter uma aplicação executando 24 horas, 7 dias por semana é estar ao alcance de qualquer pessoa com acesso à Internet, pronto para trocar informações a um custo muito baixo, abrangendo assim qualquer camada de renda da população, das mais altas até as mais baixas.

Dessa maneira, os aplicativos Web são muito bem vistos pela sociedade pois trazem enorme comodidade para seus usuários pois podem acessar de qualquer lugar do mundo as suas páginas favoritas e obter informações que lhes sejam convenientes, sejam de um blog ou um comércio eletrônico, por exemplo. O usuário que deseja fazer uma simples pesquisa ou adquirir um lote de produtos do outro lado do globo pode facilmente fazê-lo com poucos cliques.

Dentro desse contexto, não deve ser esquecido que o atendimento eficiente das necessidades do usuário é muito importantes pois a tecnologia e as facilidades da Internet abrem um importantíssimo meio de interação com estes, que deve ser feito por meio de interfaces mais simples e intuitivas. Desta forma pode-se gerar muito mais valor para o cliente de maneira mais fácil e econômica, o que é extremamente saudável na medida em que possibilita agregar valor para ambas as partes, ou seja, se o cliente ganha, toda organização também ganha.

Ainda nessa visão, todas as partes do sistema também devem evoluir, ou seja, não somente a interatividade com o cliente em si fica mais robusta, e o uso de software visando o melhoramento de processos para melhor atender a demanda dos clientes de uma empresa também se intensifica. Isso se deve em grande parte da diminuição dos custos de hardware e refinamento das técnicas de desenvolvimento de sistemas ao longo dos anos, juntamente com a difusão de sistemas baseados na Web.

1.1 Identificação e justificativa

Diamantina é um município com pouco menos de 50 mil habitantes onde o comércio de carros seminovos e motocicletas seminovas se fazem pelas pequenas

concessionárias que situam na cidade ou também pela divulgação individual dos que desejam vender/trocar seus veículos. Os interessados em adquirir ou trocar seus automóveis acabam por ter de procurar informações, preços e contatos pessoalmente, não havendo uma forma mais ágil e econômica para isso.

Das nove concessionárias de seminovos em destaque no município, apenas duas se demonstram forte em vendas e nenhuma delas possuem um sistema online de porte convidativo para manter/atrair clientes. Deste modo, esse setor de comércio em Diamantina carece de uma forma mais simples, viável, ágil e econômica para difundir/divulgar e propagar informações, preços e contatos de todos os que desejam vender/trocar seus automóveis ou suas motocicletas no município.

Tendo em vista o cenário Diamantinense, a criação de uma aplicação Web tendo como base o comércio de carros, motocicletas, automóveis sanará em grande parte essa demanda na cidade, onde o mercado necessita de um serviço de qualidade para o setor.

1.2 Objetivos

- Criar um aplicativo web de divulgação e comercialização de carros e motocicletas voltado para cidade de Diamantina - MG.

1.2.1 Objetivos específicos

- Levantar requisitos do sistema baseado nas necessidades de futuros usuários da cidade de Diamantina - MG.
- Modelar os componentes que irão suprir o motor de persistência.
- Implementar uma aplicação web utilizando a linguagem Java.

2 REVISÃO BIBLIOGRÁFICA

A tecnologia da informação tem se tornado o diferencial em organizações do momento atual, seja na área comercial, seja na área educacional. Os sistemas de informação que são “conjunto de partes, componentes, que interagem entre si, de forma ordenada, a fim de atingir um objetivo comum” (STAIR, 1998; LAUDON & LAUDON, 2004) estão suprindo grande parcela da demanda por serviços na Web.

2.1 Internet

Segundo (ACM, 2009), a Internet surgiu em meados dos anos 60, na época da guerra fria nos EUA. O departamento de defesa americano sentiu a necessidade de criar uma rede de comunicações de computadores em pontos não centralizados, a intenção era justamente descentralizar toda informação para que se houvesse um bombardeiro inimigo, não perdessem informações de grande valor em um único servidor. Anos após, a Internet foi considerada a maior criação tecnológica desde a televisão e viria a alcançar em maiores proporções a população à partir da década de 1990, onde sua relevância para o meio acadêmico e comercial se tornaram marcantes.

Segundo Kurose (2006), a Internet é uma rede que interconecta milhares de dispositivos computacionais ao redor do mundo. Há pouco tempo esses dispositivos eram basicamente os computadores de mesa, as estações de trabalho e os chamados servidores que armazenam e transmitem informações, como páginas da Web e mensagens de e-mail. Para que os computadores possam fazer essas trocas de dados e informações é necessário um conjunto de regras padronizando o formato e a sintaxe das mensagens trocadas. Estas regras são chamadas de protocolos. Por sua vez, protocolos são regras/convenções que dois computadores devem obedecer quando trocam informações. Este conjunto de regras é padronizado e especifica o formato, a sincronização, o sequenciamento e a verificação de erros em comunicações de dados. Um protocolo de comunicação também pode ser definido como um conjunto de regras acordadas que permite a comunicação entre os computadores, este acordo que faz o controle e possibilita a comunicação e consequentemente uma conexão com transferência de dados entre dois ou mais computadores.

Os protocolos são padronizados por um grupo de técnicos e pesquisadores denominado IETF[1] (“Internet Engineering Task Force”), tendo por objetivo sempre a evolução da Internet e seu bom funcionamento, é este grupo que propõe padronização das tecnologias e protocolos. Como exemplos notórios utilizados pela Internet, temos os protocolos listados pela RFC[2] (“Request for Comments”):

A. FTP (RFC 959): O “File transfer protocol” ou protocolo de transferência de dados é um protocolo para transferência de arquivos. É uma forma flexível e rápida para esta tarefa.
B. TCP (RFC 793): O “Transmission Control Protocol” ou protocolo de controle de transmissão é um protocolo altamente confiável e robusto utilizado pela Internet. Este protocolo garante o envio sem erros dos dados transmitidos pela rede.
C. IP (RFC 791): O “Internet Protocol” ou protocolo da Internet é usado entre duas ou mais máquinas em rede para encaminhamento dos dados. Possui vários cabeçalhos inclusive endereço de destino e de origem.
D. IPSec (RFC 6071): O “IP secure” ou IP segure é uma extensão do protocolo IP que visa a confiabilidade, autenticidade e integridade das informações.
E. UDP (RFC 768): O User Datagram Protocol ou Procolo de Datagrama do Usuário é semelhante ao TCP, porém, não garante a entrega correta dos dados, se estes dados chegarão ou não ao destino. Este protocolo escreve um pacote encapsulado de dados e envia.
F. HTTP (RFC 2616): O “Hypertext Transfer Protocol” ou protocolo de transferência de hipertexto é utilizado em alguns sistemas para a transmissão de informação de hipermídia. O HTTP é de suma importância para a comunicação na *World Wide Web*.
G. HTTPS (RFC 2818): Basicamente é o protocolo HTTP seguro. Este protocolo possui uma camada adicional de segurança (HyperText Transfer Protocol Secure ou protocolo de transferência de hipertexto seguro).

[1] IETF Disponível em <https://www.ietf.org/>. Acesso em 18 de mar. 2015
[2] RFC Disponível em <https://www.ietf.org/rfc.html>. Acesso em 18 de mar. 2015

H. SMTP (RFC 2821): O “Simple Mail Transfer Protocol” ou protocolo de transferência de correio simples tem como característica importante a sua capacidade para o transporte ou envio de correio eletrônico através da Internet. É um protocolo simples onde é possível especificar os destinatários.
I. ICMP (RFC 792): O “Internet Control Message Protocol” ou Protocolo de controle de mensagens é utilizado para fornecer relatórios de erros à fonte original. Os computadores que utilizam o protocolo IP aceitam as mensagens ICMP de acordo com o erro recebido.
J. POP (RFC 1939): O “Post Office Protocol” ou Protocolo de correios é um protocolo que permite acessar remotamente uma caixa de correio eletrônico. Quando se fala remotamente, o protocolo permite baixar todas mensagens de e-mail para o computador local e desta forma acessá-los.
K. SSH (RFC 4253): O “Secure Shell” ou Shell Seguro é um programa de computador e também um protocolo de rede que permitem a conexão com outro computador na rede de forma segura, permitindo a execução de comandos de uma unidade remota. É muito utilizado em sistemas operacionais derivados do Unix.

A Internet utiliza esses e outros protocolos não citados, é uma rede de computadores mundialmente interligadas que utilizam deles para se comunicarem e servir os seus usuários. Segundo Ward (2009), a Internet consiste em uma rede de várias outras redes onde podemos encontrar um vasto acervo de recursos e informações, documentos dos mais diversos assuntos acadêmicos, não acadêmicos e serviços. Todo esse conteúdo é disponibilizado através de *hipertextos e hipermídia* da *World Wide Web*.

2.2 Web

Segundo Levy (1998), a Web (“World Wide Web”) é um sistema de documentos em hipertexto e hipermídia que são interligados e executados através da Internet, tais documentos podem ser fornecidos em diferentes formatos como de vídeos, sons, textos e figuras. A Web foi criada por uma equipe dirigida por Tim

Berners-Lee, no CERN[3] (Organização Europeia para a Pesquisa Nuclear), em Genebra, para melhorar a pesquisa cooperativa entre os físicos. Esse sistema permitiu interconectar todos os documentos digitalizados e disponibilizados na Internet e de torná-los acessíveis a partir de qualquer lugar do mundo com acesso à Internet e um computador.

Sobre essa perspectiva, segundo Gosciola (2003, p.34) a Hipermídia é definida como:

> Hipermídia é o conjunto de meios que permite acesso simultâneo a textos, imagens e sons de modo interativo e não-linear, possibilitando fazer links entre elementos de mídia, controlar a própria navegação e, até, extrair textos, imagens e sons cuja sequência constituirá uma versão pessoal desenvolvida pelo usuário.

A Web, segundo Press (1999), provavelmente foi a invenção de maior revolução na história da tecnologia, essa evolução e disseminação do uso da Web não foi nem prevista e nem desejada pelas grandes multinacionais da informática, das empresas de telecomunicações ou da mídia, mas se expandiu como uma epidemia entre os já usuários de computadores. Desta forma, pessoas e grupos que realmente desejavam publicar um texto, uma música ou imagens na Web tiveram grandes oportunidades tornando as informações disponíveis para um vasto público internacional, sem desconsiderar grandes oportunidades de emprego e renda que surgiram no mercado.

Outra vantagem dos Sistemas Baseados na Web é a busca constante de características positivas no uso de hipermídia onde Bieber (1997, p.35) diz:

> Os benefícios de adicionar funcionalidade hipermídia às aplicações de sistemas de informação são que a hipermídia proporciona acesso navegacional, contextual para ver informação e que representa conhecimento em uma forma relativamente próxima das estruturas cognitivas organizacionais que as pessoas usam. Assim, a hipermídia apoia entendimento.

Nessa linha, segundo Gosciola (2003), o Hipertexto pode ser descrito como um termo ligado a um texto, onde se agregam conjuntos de informação na forma de blocos de textos, palavras, imagens ou sons, cujo acesso é realizado por meio de

[3] CERN Disponível em <http://home.web.cern.ch/>. Acesso em 18 de mar. 2015

referências específicas, no meio digital são denominadas hiperlinks ou links. Já a Hipermídia, outra grande responsável pelo conteúdo na web, se define como a reunião de várias mídias num suporte computacional, suportado por sistemas eletrônicos de comunicação.

A difusão da Web acarretou em um grande crescimento em número e complexidade dos sistemas de informação baseados na Web, na qual segundo Press (1999) ainda classificou as aplicações Web como representantes da quarta geração dos sistemas de processamento de dados empresariais, sendo posterior as gerações dos sistemas de processamento em lote, sistemas de compartilhamento de tempo e das aplicações cliente-servidor.

2.3 Comércio Eletrônico

O comércio eletrônico trata-se de uma forma de transação eletrônica para compra e venda de produtos ou serviços especialmente através da Internet, sendo a Internet mais um canal para aquisição de bens disponíveis na rede através de lojas virtuais. A transação comercial é feita através de um equipamento eletrônico, como computadores, tablets, smartphones ou qualquer tipo de aparelho eletrônico que possa se conectar a um servidor Web e fazer um pedido de compra. O varejo virtual tem crescido bastante nos últimos anos e com isso atraído milhares de novas lojas, desde as maiores empresas do mercado até micro e pequenas empresas dos lugares mais distantes (SEBRAE, 2012).

Segundo Albertin (2011), no comércio eletrônico a compra é efetuada através de um dispositivo eletrônico, com realização dos processos de negócio em um ambiente virtual, por meio da aplicação intensa das tecnologias de comunicação e de informação, atendendo aos objetivos do negócio. Esses processos podem ser realizados tanto dentro da empresa, quanto fora da empresa com outras empresas e com outros consumidores, através de uma infraestrutura de informação e comunicação que em sua maior parte é pública, de acesso fácil e de baixo custo. Essa definição permite entender que a realização dessa cadeia de valores deve incluir desde a distribuição de informações de produtos e serviços até a realização de transações entre as partes.

Ainda nesta perspectiva, Albertin (2011, p. 95)

> Comércio eletrônico é a realização de toda a cadeia de valores dos processos de negócio em um ambiente eletrônico, por meio da aplicação intensa das tecnologias de comunicação e de informação, atendendo aos objetivos de negócio. Os processos podem ser realizados de forma completa ou parcial, incluindo as transações negócio-a-negócio, negócio-a-consumidor e intra-organizacional, em uma infra-estrutura de informação e comunicação predominantemente pública, de acesso fácil, livre e de baixo custo

Desta forma, segundo Felipini (2002), o *e-commerce* veio para ajudar as organizações crescerem e a adquirirem mais clientes, conhecerem mais os clientes, proporcionar uma melhor qualidade nas vendas, oferecendo para o cliente aquilo o que ele realmente deseja e necessita, e também aumentando os lucros. E tudo isso, por um pequeno investimento de tecnologia e utilizando a Internet.

Nesta visão, Felipini (2002, p. 41)

> Desde que o homem começou a transacionar, passando pelo surgimento do Marketing, sabe-se que a longo prazo o que efetivamente importa é o atendimento eficiente das necessidades do cliente. Se ele está satisfeito, você tem um negócio vencedor. Pois a Tecnologia e as facilidades da Internet abrem um importantíssimo canal de interação com seus clientes, atuais e futuros. Através da Internet pode-se buscar novos clientes, pode-se conhecer melhor os hábitos e comportamento de seu clientes atuais de forma até a antecipar suas necessidades, pode-se fazer atendimento personalizado a centenas de milhares de consumidores, naquilo que se chama customização em massa. Pode-se atendê-lo melhor. Enfim, pode-se gerar muito mais valor para o cliente de forma mais fácil e econômica, o que, inclusive, é extremamente saudável na medida em que possibilita às pequenas empresas uma disputa mais equilibrada pelo mercado.

E, é nesse meio que se enquadra o comércio eletrônico, que surgiu como um diferencial estratégico de vendas para as organizações utilizando a Web, uma boa estratégia para avançar além do mercado físico e agora também o virtual, sempre adquirindo e conhecendo cada vez mais os seus clientes e podendo levar aquilo o que o consumidor deseja e necessita com a maior comodidade para ele, e assim, vendendo mais, desenvolvendo, ganhando novos mercados além da *Web* e obtendo sucesso.

2.4 Evolução do Comércio Eletrônico

Segundo Albertin (2011), em se tratando do caminho percorrido pelo comércio eletrônico, foi percebido que as organizações do mundo todo inclusive as brasileiras utilizaram as tecnologias de informação e comunicação para interagir com seus clientes, porém de forma meramente informativa alguns anos atrás. Mais adiante essa iteração se tornou mais forte, empresas passaram a interligar suas várias áreas para aproximar de seus clientes e fornecedores. Tendo como base uma maior proximidade com fornecedores e clientes, essas empresas puderam minerar e processar um número maior de transações e atender a uma quantidade maior de clientes de forma rápida, segura e personalizada.

Segundo Felipini (2012), empresas estão efetivamente utilizando de várias artimanhas no ramo do comércio eletrônico, ainda que focando inicialmente em sua maior parte nos seus processos de negócio apenas. O próximo estágio da sua evolução será permear também os processos internos e integrá-los com os externos de forma automática, criando o novo ambiente de negócio. Algumas empresas estão utilizando do comercio eletrônico em diversos setores da economia, estando em um estágio de evolução constante. O cenário atual, mesmo que não uniforme para maioria das empresas, permite entender tal evolução e identificar suas tendências.

De acordo com o levantamento realizado pela Câmara E-net (Câmara Brasileira de Comércio Eletrônico), o segmento com maior participação nas vendas é o dos eletrodomésticos, seguido pelos produtos de saúde, beleza e medicamentos. Na terceira posição vem o segmento de moda, vestuário e acessório. O segmento literário composto por revistas, livros e assinaturas de jornais segue na quarta posição. Há cerca de dez anos a categoria que ocupava o ponto mais alto do ranking era o segmento dos CDs e DVDs, com cerca de 40% das vendas. O setor de serviços também está se alavancando de forma rápida. Tudo isso indica que o ramo tende a crescer mais e mais.

Diante de todos os fatos, não podemos desconsiderar que o comércio eletrônico e sua utilização, com o decorrer dos anos, deixou de ser somente uma característica

diferencial para as empresas e passou a ser uma grande ferramenta para conhecer cada vez mais as necessidades dos clientes e alavancar as vendas.

2.5 Arquitetura Cliente Servidor

Segundo Mendes (2002), a arquitetura cliente servidor é utilizada pela maioria dos aplicativos Web. Esta arquitetura divide a carga de trabalho em duas unidades: o cliente e o servidor. O cliente é responsável pelo encaminhamento de requisições ao servidor, envia a requisição e fica aguardando a resposta do servidor que oferece algum tipo de serviço, isto é, pode ser um servidor de e-mail, de páginas ou qualquer outra função. O princípio básico de uma rede de computadores é o compartilhamento de recursos e o servidor é uma peça fundamental para atingir este objetivo, ele fica sempre aguardando uma ou mais requisições de um ou mais clientes, quando a requisição chega ele faz o processamento da requisição e devolve a resposta para o cliente.

Deste modo, a arquitetura cliente-servidor permite na maioria dos casos que os papéis e responsabilidades de um sistema de computação possam ser distribuídos entre vários computadores independentes que são conhecidos por si só através de uma rede. Isso cria uma maior facilidade de manutenção. É possível substituir, reparar, atualizar ou mesmo realocar um servidor de seus clientes sem que eles saibam, abstraindo a complexidade dos mesmos (Mendes, 2002).

2.6 Arquitetura de Software

Segundo Pressman (1995), a arquitetura de software de um programa ou sistema computacional é toda a estrutura do sistema que abrange os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles. Já segundo Sommerville (2007), em se tratando de arquitetura de software, a arquitetura de software é a definição dos elementos que compõem uma estrutura e como eles se relacionam. Tais elementos citados seriam as classes componentes e serviços relacionados ao software. Ainda pode-se definir como um sistema que consiste na definição dos componentes de software, suas propriedades externas, e os relacionamentos com outros software.

Segundo Mendes (2002), Arquitetura de Software é o estudo da organização global dos sistemas de software bem como do relacionamento entre subsistemas e componentes. Desde suas origens, quando descrições qualitativas de organização de sistemas eram consideradas úteis, a arquitetura de software tem amadurecido ao longo da última década, buscando englobar e exportar notações, ferramentas e técnicas de análise.

Segundo Garlan e Shaw (1994),

> Várias definições são dadas para arquitetura de software, sendo que a maioria delas faz uso dos conceitos de componentes e conexões. Uma Arquitetura de Software descreve os elementos com os quais um sistema é construído, assim como a forma como esses componentes irão interagir.

Um exemplo concreto de Arquitetura de Software é o “Model View Controller” que, inclusive, é a arquitetura utilizada para construção e elaboração deste trabalho.

2.7 Servidor Web

Segundo Moraes (2001), a função principal do servidor Web é dar suporte as requisições feitas pelos usuários da web, sejam essas requisições buscando arquivos ou um simples documento HTML[4]; para isso, os servidores web podem ser integrados a um banco de dados e atender às requisições do usuário, controlando a interação devidamente. Um serviço de correio eletrônico, por exemplo, dispõe de uma série de software e protocolos para fornecer ao usuário esse serviço.

Desta forma, o servidor Web é responsável por disponibilizar um conjunto de páginas para que se comuniquem com usuário ou com outras páginas na Web. Um dos protocolos que um servidor web aceita e processa são as requisições do famoso HTTP (Hypertext Transfer Protocol). Segundo Moraes (2001), as requisições HTTP que se referem geralmente à páginas HTML são usualmente feitos através de “browsers” e o método se inicia com a conexão entre o computador onde está estabelecido o servidor web e o dispositivo do cliente. Como na web é impossível prever a que hora será feito o acesso, os servidores web tendem a estar acessíveis sempre, a qualquer momento. Os servidores web também podem executar programas e scripts.

[4] HTML, Disponível em <http://www.w3schools.com/html/>. Acesso em 19 de mar. 2015.

A figura 1 representa como um servidor web interage com o cliente, mostrando assim a camada da aplicação, de interface e gerenciamento de dados.

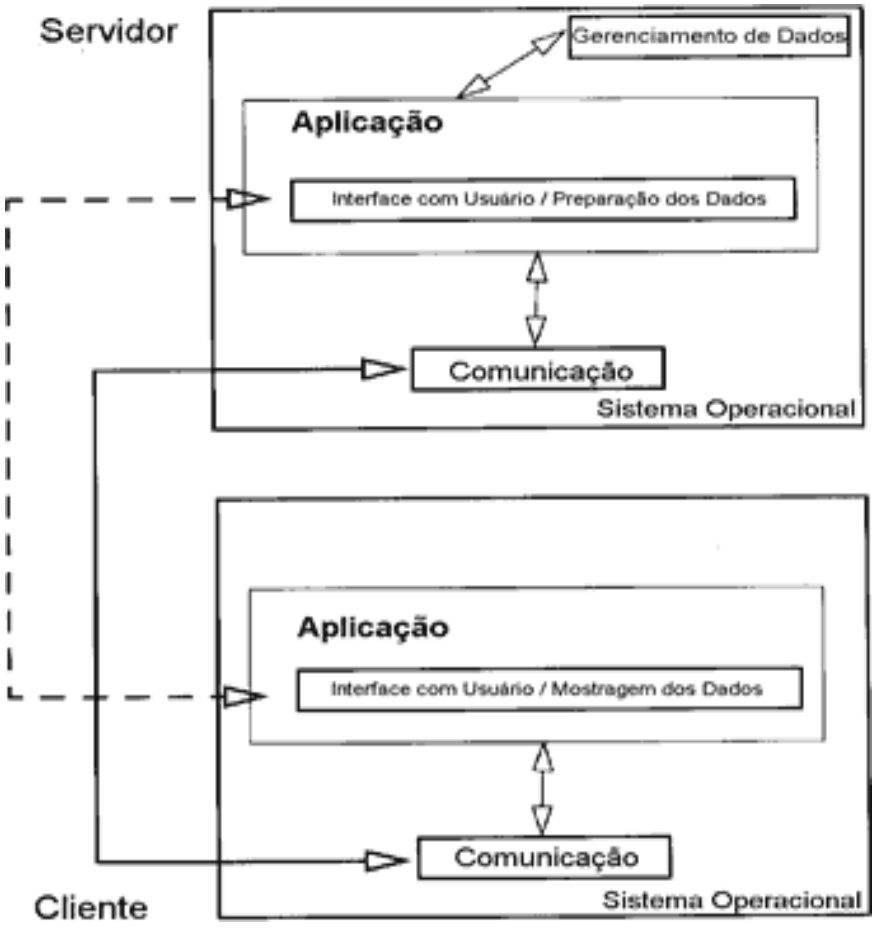

Figura 1 - Interação Cliente e Servidor Web.
Fonte: Moraes (2001).

2.8 Servidor de Aplicação

Segundo Deitel (2005), servidor de aplicação é um ambiente de software que disponibiliza diversas APIs (Application Programing Interface) para a manipulação de uma aplicação. De forma mais detalhada, fornece um ambiente para a instalação e execução de certas aplicações, mantendo tudo centralizado e dispensando a instalação nos computadores clientes.

Parecido com a ideia de um framework, um servidor de aplicação é um conjunto de classes que colaboram para realizar um objetivo, um servidor se diferencia apenas por ser uma plataforma de APIs a fim de realizar este mesmo objetivo, separando assim o desenvolver de complexidades inerentes de um

sistema computacional. Segundo DEITEL (2005), os servidores de aplicação podem ser considerados como um *middleware*, um mediador entre software e demais aplicações.

Desta forma, as vantagens dos servidores de aplicação junto ao modelo cliente servidor está nos serviços implementados por eles e acessíveis aos desenvolvedores, fazendo com que eles possam canalizar a maior parte do tempo no desenvolvimento da lógica de negócio. Deste modo, no desenvolvimento de aplicações, a ênfase dos desenvolvedores pode ser voltado a solucionar problemas relacionados ao sistema, e não de infraestrutura da aplicação. Além das especificações e características já citadas, outros serviços também são disponibilizados pelos servidores de aplicação:

A. Tolerância a Falhas: um atributo que permite que sistemas continuem a operar adequadamente mesmo após falhas em alguns de seus componentes. Se a qualidade de uma operação diminui ou venha a falhar, a queda é proporcional à severidade da falha, trazendo maior confiabilidade.
B. Balanceamento de carga: este promove o aumento de performance através da distribuição da carga de forma balanceada e possui a qualidade de distribuir o tráfego fazendo com que diferentes máquinas que compõe um cluster, por exemplo, funcionem como uma única.
C. Gerenciamento de Componentes: este tem por objetivo gerenciar elementos do servidor através de ferramentas para a manipulação de componentes e serviços, tais como gerenciamento de sessão, notificação, distribuição da lógica de negócios.
D. Gerenciamento de Transações: proporciona garantia e integridade das transações entre os aplicativos e bancos de dados envolvidos.

Existem inúmeras implementações de servidores de aplicação, em sua maioria implementados na plataforma Java. Como exemplo temos:

A. Oracle Oracle 9i Application Server;
B. Red Hat JBoss
C. Sun GlassFish
D. Apache Gerônimo

2.9 Programação Orientada a Objetos

Um dos problemas da antiga programação estruturada é que, muitas vezes, partes do código que servem apenas para tratarem os dados se misturam com partes do código que tratam da lógica do algoritmo. Essa prática não é viável, na medida em que diminui a reusabilidade e dificulta leitura/depuração do sistema.

A modularização da programação estruturada foi um grande avanço na busca pela reusabilidade de código, no entanto, nem se compara ao que trouxe a programação orientada a objetos. Esse novo paradigma reflete bem mais fielmente os problemas atuais, é um paradigma que se baseia na abstração de coisas do mundo real em um sistema.

O paradigma orientado a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas específicas e é pela interação entre objetos que uma tarefa computacional é realizada.

A programação orientada a objetos auxilia na modelagem de sistemas, reduzindo a diferença semântica entre a realidade sendo modelada e os modelos construídos. Um sistema orientado a objetos consiste em objetos em colaboração com o objetivo de realizar as funcionalidades desse sistema e cada objeto é responsável por tarefas específicas e a cooperação entre eles é importante para o desenvolvimento do sistema.

Sommerville (2007, p. 208) afirma que:

> A análise orientada a objetos concentra-se no desenvolvimento de um modelo orientado a objetos do domínio da aplicação. Os objetos nesse modelo refletem as entidades e as operações associadas ao problema a ser resolvido".

A Orientação a Objetos possui alguns conceitos básicos ou pilares fundamentais como: abstração, classes, objetos, atributos, métodos, encapsulamento, herança e polimorfismo.

2.9.1 Abstração

Abstração é a subtração de detalhes, ou seja, quanto mais abstrato, há menos detalhes, e quanto menos abstrato, há mais detalhes. É um processo mental pelo qual os seres humanos se atem aos aspectos mais importantes (relevantes)

de alguma coisa, ao mesmo tempo que ignora-se os aspectos menos importantes. Esse processo mental nos permite gerenciar a complexidade de um objeto, ao mesmo tempo que concentra-se a atenção nas características essenciais do mesmo.

Correia (2006, p. 11) afirma que:

> Pelo princípio da abstração, nós isolamos os objetos que queremos representar do ambiente complexo em que se situam, e nesses objetos representamos someta as características que são relevantes para o problema em questão".

A abstração de algo é dependente da perspectiva (contexto) sobre a qual uma coisa é analisada: o que é importante em um contexto pode não ser importante em outro.

2.9.2 Classes e Objetos

Segundo Correia (2006), objetos são coisas (carro, foto, bola, etc) e classes são um agrupamento de coisas. A classe é a descrição dos atributos e serviços comuns a um grupo de objetos, logo pode-se dizer que é um modelo/template a partir do qual objetos são construídos. Podemos dizer que objetos são instâncias de classes.

São componentes de um objeto: identidade, estado e comportamento. A identidade é responsável por distinguir um objeto dos outros, ou seja, eles são únicos, mesmo que sejam instâncias de uma mesma classe e que tenham os mesmos valores. O estado reflete os valores correntes dos atributos do objeto em um determinado momento.

Para Ambler (1998, p. 5):

> Um objeto é qualquer indivíduo, lugar, evento, coisa, tela, relatório ou conceito que seja aplicável ao sistema". Todo objeto pertence a uma determinada classe e possui atributos próprios. Os atributos são mutáveis e podem receber diferentes valores de acordo com as características do objeto.

Já o comportamento se refere a como os objetos reagem em relação a mudança de estado e troca de mensagens, ou seja, é um conjunto de atividades externamente observáveis do objeto. Identidade é o que torna o objeto único;

Resumindo, estado se refere aos seus atributos e comportamento se refere aos seus métodos e procedimentos.

Alguns conceitos importantes:

- Variável/Método de Classe (também chamados estáticos): similar a uma variável global, é uma variável cujo valor é comum a todos os objetos membros de uma classe.
- Variável/Método de Instância: é uma variável cujo valor é específico ao objeto e, não, à classe. Em geral, ela possui um valor diferente para cada instância.

2.9.3 Atributos

Consiste em uma informação de estado, para o qual cada objeto de uma classe tem seu próprio valor. Há dois tipos: atributos de objetos e de classes. O primeiro descreve valores mantidos em um objeto. Diferentes objetos de uma mesma classe não compartilham os atributos de objetos, ou seja, cada um possui sua própria cópia do atributo. O segundo é aquele cujo valor todos os seus objetos devem compartilhar.

As mensagens enviadas a um objeto (a chamada de um método) podem mudar o valor de um ou mais atributos, alterando o estado de um objeto. Um atributo é um dado para o qual cada objeto tem seu próprio valor. Basicamente são a estrutura de dados que representam a classe.

2.9.4 Métodos

Similares a procedimentos e funções, são descrições das operações que um objeto executa quando recebe uma mensagem, esta mesma mensagem pode resultar em métodos diferentes quanto enviada para objetos diferentes.

Segundo Ricarte (2001), os métodos construtores são métodos especiais, que são chamados automaticamente quando instâncias. Seu objetivo é garantir que o objeto será instanciado corretamente. Ele tem exatamente o mesmo nome da classe que está inserido, não possui tipo de retorno e não é necessário declará-lo.

Através da criação de construtores, pode-se garantir que o código que eles contêm será executado antes de qualquer outro código em outros métodos. Eles

geralmente são usados para preparar um objeto, inicializando as variáveis do objeto. Pode existir mais de um método construtor em uma classe através da sobrecarga de construtores.

2.9.5 Encapsulamento

O mecanismo de encapsulamento é uma forma de restringir o acesso ao comportamento interno de um objeto. Um objeto que precise da colaboração de outro objeto para realizar alguma operação simplesmente envia uma mensagem a este último.

Segundo o mecanismo do encapsulamento, o método que o objeto requisitado usa para realizar a operação não é conhecido dos objetos requisitantes. Em outras palavras, o objeto remetente da mensagem não precisa conhecer a forma pela qual a operação requisitada é realizada, tudo o que importa a esse objeto remetente é obter a operação realizada, não importando como.

Correia (2006, p. 13) afirma que:

> As pessoas que usam os objetos não precisam se preocupar em saber como eles são constituídos internamente acelerando o tempo de desenvolvimento.

Em geral, utilizam-se especificadores de acesso para privar o acesso direto a atributos/métodos e obrigar o usuário a fazê-lo por meio de métodos públicos, efetivando assim, o mecanismo de encapsulamento.

- Private: Somente a classe tem acesso;
- Protegido: Somente a classe e subclasses tem acesso;
- Público: Todos tem acesso;
- Pacote: Todos do pacote tem acesso menos o global.

2.9.6 Herança

Segundo Ricarte (2001), a herança é utilizada na orientação a objetos e pode ser vista como um nível de abstração acima da encontrada entre classes e objetos. Na generalização, classes semelhantes são agrupadas em uma hierarquia. Cada nível dessa hierarquia pode ser visto como um nível de abstração.

Cada classe em um nível de hierarquia herda as características e o comportamento das classes às quais está associada nos níveis acima dela.

Ademais, essa classe pode definir características e comportamento particulares. Dessa forma, uma classe pode ser criada a partir do reuso da definição de classes preexistentes. A generalização facilita o compartilhamento de comportamento comum entre classes.

Podemos dizer que se trata do mecanismo que permite que classes compartilhem atributos e métodos, com o intuito de reaproveitar o comportamento generalizado ou especializar operações e atributos. Quando uma classe herda de várias classes, é conhecido como Herança Múltipla e quando herda de apenas uma classe, é conhecido como Herança Simples.

2.9.7 Polimorfismo

O polimorfismo indica a capacidade de abstrair várias implementações diferentes em uma única interface, ou seja, permite que a mesma mensagem seja enviada a diferentes objetos e que cada objeto execute a operação que é mais apropriada a sua classe. Há uma relação estreita com o conceito de abstração, na medida em que um objeto pode enviar a mesma mensagem para objetos semelhantes, mas que implementam a sua interface de formas diferentes.

Segundo Sintes (2002), “o polimorfismo é o distúrbio das múltiplas personalidades do mundo do software, pois um único nome pode expressar muitos comportamentos diferentes”.

O polimorfismo pode ser estático ou dinâmico. O primeiro é também conhecido como sobrecarga ou “overloading”, é representado com o nome do método igual e argumentos diferentes. A decisão do método a ser chamado é tomada em tempo de compilação de acordo com os argumentos passados.

O segundo é também conhecido como sobrescrita, redefinição ou “overriding”. Ele está associado ao conceito de herança e é representado com o nome e argumentos do método iguais. A subclasse redefine o método da superclasse e a decisão do método a ser chamado é tomada em tempo de execução.

2.10 FrameWork

Gamma (2000) esclarece que o framework dita a arquitetura de uma aplicação. É o framework que define sua estrutura geral, sua divisão em classes e objetos e em consequência as responsabilidades das classes e objetos, como estas colaboram, e o fluxo de controle. Um framework predefine estes parâmetros de projeto, de maneira que o projetista ou programador da aplicação possa concentrar-se nos aspectos específicos da aplicação.

Um framework é um conjunto de classes que colaboram para realizar um objetivo. Pode ser definido como um projeto abstrato orientado a objetos que pode ser adaptado segundo as necessidades da aplicação fornecendo um grau de reutilização bem alto, servindo como moldes para a construção de aplicações (Figura 2).

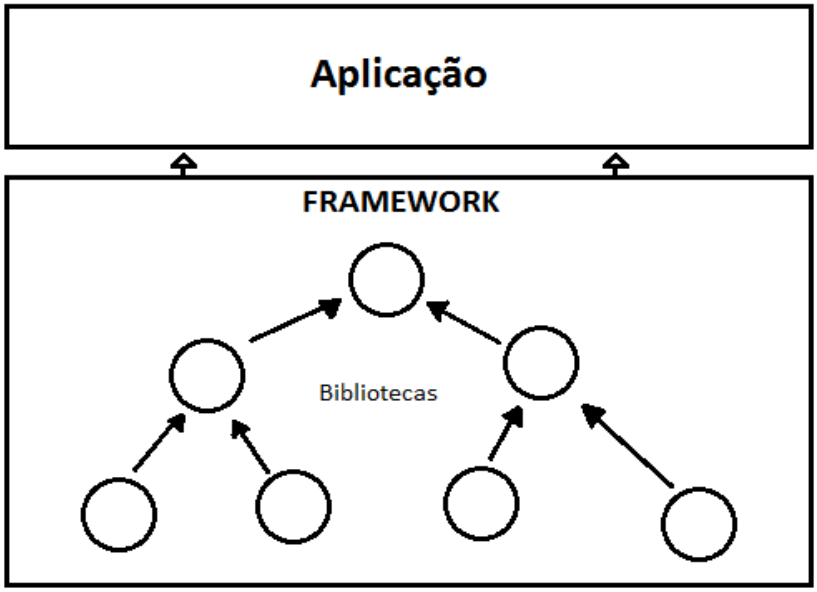

Figura 2 - Estrutura de um framework.

Um framework é visto com bons olhos pois serve para aumentar a produtividade das equipes de desenvolvimento trazendo maior organização separando as responsabilidades de apresentação, controle e negócio e da existência de diversas funcionalidades prontas.

Abstraindo um pouco mais, o framework, também conhecido como arcabouço, em desenvolvimento de software, é uma abstração que utiliza de

códigos comuns que vários software poderão utilizar como arcabouço, fazendo uma funcionalidade mais comum. Desta forma ele trabalha com uma funcionalidade específica durante a programação de uma aplicação. É o framework quem mostra o fluxo de controle do sistema. O mesmo atua onde há funcionalidades em comum a várias aplicações.

Segundo Goldberg (1995), os frameworks orientado a objetos disponibilizam um vasto acervo para aumentar a produtividade e a qualidade no desenvolvimento de software. Basicamente, aplicações especificas são construídas especializando as classes do framework para fornecer a implementação de alguns métodos, enquanto a maior parte da funcionalidade da aplicação é herdada. Esta característica permite a reutilização do código e do projeto das aplicações em questão, o qual representa um benefício importante em comparação a outras tecnologias de reutilização. Possuem vantagens por serem peças mais certas de software, tais como maior facilidade para a detecção de erros, concentração na abstração de soluções, eficiência e otimização de recursos.

2.11 Padrão de arquitetura MVC

Conforme Quicoli (2007), o padrão de projeto MVC separa a aplicação em três camadas distintas, “model”, “view” e “controller”, que podem ser definidas da seguinte forma:

a) “Model” é a representação da informação que a aplicação utiliza. Em um sistema orientado a objetos o model pode ser composto pelas classes de entidades do sistema, como por exemplo classes Pessoa, Cliente e NotaFiscal;
b) “View” é a apresentação gráfica do model. Por exemplo, um formulário expondo as 50 informações de um cliente. Uma view está sempre ligada a um model e sabe como exibi-lo ao usuário. Para um determinado model podem existir várias views. Além disso, uma view também pode ser composta por várias views;
c) “Controller” é responsável por mapear as ações do usuário em ações do sistema, por exemplo se um botão ou item do menu é clicado é ele que deve definir como a aplicação responderá.

A vantagem da separação da aplicação nestas três camadas é torná-las independentes e desacopláveis, além de possibilitar a reutilização do mesmo “model” em várias “views”. A Figura 2.11 mostra como é feita a comunicação entre as três camadas do MVC, na qual as linhas contínuas representam métodos e as linhas pontilhadas representam eventos.

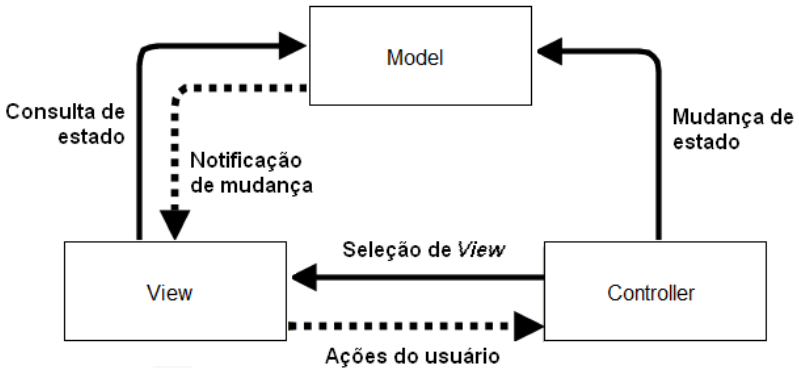

Figura 3 - A comunicação dentro do MVC.
Fonte: Quicoli (2007).

Segundo Marcoratti (2008), a arquitetura MVC nos traz uma forma de dividir as funcionalidades envolvidas na manutenção e apresentação dos dados de uma aplicação. A arquitetura MVC não é recente e foi inicialmente desenvolvida para mapear as tarefas tradicionais de entrada, processamento e saída para o modelo de interação com o usuário. Usando o padrão MVC fica fácil mapear esses conceitos no âmbito das aplicações multicamadas.

Conforme Quicoli (2007, p. 26-34):

> O modelo MVC se torna mais atrativo devido ao crescimento e difícil entendimento das aplicações desenvolvidas hoje em dia, a programação orientada a objeto junto ao modelo se torna mais necessária, desta forma fica relevante a separação entre os dados e a visão das aplicações. É importante salientar que esta arquitetura permite alterações feitas na View sem afetar a manipulação de dados, e estes poderão ser redefinidos e organizados sem alterar a forma de mostrá-lo, como exemplo, podemos mostrar tipos de dados em formas diferentes como em tabelas, gráficos, diagramas ou mais formas que forem necessárias utilizando os mesmos dados.

Desta forma, o padrão MVC resolve vários problemas que vários desenvolvedores enfrentam através da separação das tarefas de acesso aos dados

e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um controlador entre os dois para facilitar e melhorar a complexidade de todo o negócio, não somente o sistema.

3 FERRAMENTAS UTILIZADAS

Na implementação deste sistema com a finalidade de divulgação e comercialização de carros e motocicletas no município de Diamantina foi utilizado a linguagem de programação Java que vem acompanhada com seu paradigma de programação orientada a objetos. A plataforma utilizada foi a JEE[5] (Java Enterprise Edition). Na camada de persistência de dados de acordo com as especificação JEE foi utilizado o framework Hibernate[6] junto ao sistema gerencial de banco de dados MySQL[7]. O padrão de arquitetura utilizado foi o MVC[8] (Model-View-Controller). O framework escolhido para este trabalho foi o JSF[9] (Java Server Faces), que foi utilizado junto ao framework Primefaces[10], este dedicado a camada visão do MVC. A IDE (ambiente integrado para desenvolvimento de software) utilizada foi o Eclipse[11], instalado em uma distribuição Linux (Ubuntu 14.04[12]). Por se uma aplicação desenvolvida para a Web foi necessário um servidor de aplicação embutido, desta forma foi escolhido o JBoss[13], o qual disponibilizou o JAAS[14] para desassociar as aplicações dos controles de acesso a recursos do sistema.

3.1 Java

Segundo Campione (2004), Java é uma linguagem computacional completa de baixo custo de desenvolvimento, robusta, multi-plataforma, possui diversas bibliotecas que aumentam a produtividade, é adequada para o desenvolvimento de aplicações Web, redes fechadas ou ainda qualquer programa que trabalhe sozinho, os chamados “stand alone”.

Segundo Wutka (1997) Java é uma linguagem poderosa com inúmeros recursos em ambientes distribuído. Mas sua versatilidade permite ao programador ir além, oferecendo uma poderosa linguagem de programação de uso geral, com

[5] JEE, Disponível em <https://jcp.org/en/jsr/detail?id=317>. Acesso em 19 de mar. 2015.
[6] HIBERNATE, Disponível em <http://hibernate.org>. Acesso em 19 de mar. 2015.
[7] MYSQL, Disponível em <http://www.mysql.com>. Acesso em 19 de mar. 2015.
[8] MVC, Disponível em <https://jcp.org/en/jsr/detail?id=371>. Acesso em 19 de mar. 2015.
[9] JSF, Disponível em <https://jcp.org/en/jsr/detail?id=372>. Acesso em 19 de mar. 2015.
[10] PRIMEFACES, Disponível em <http://www.primefaces.org >. Acesso em 19 de mar. 2015.
[11] ECLIPSE, Disponível em <https://www.eclipse.org/>. Acesso em 19 de mar. 2015.
[12] UBUNTU, Disponível em <http://ubuntu-br.org/>. Acesso em 19 de mar. 2015.
[13] JBOSS, Disponível em <http://www.jboss.org/>. Acesso em 19 de mar. 2015.
[14] JAAS, Disponível em <https://www.jcp.org/en/jsr/detail?id=196/>. Acesso em 19 de mar. 2015.

recursos suficientes para a construção de uma variedade de aplicativos que podem ou não depender do uso de recursos de conectividade.

Diferentemente de algumas linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada e também interpretada. Depois de escrever um programa utilizando Java, em um editor de textos qualquer por exemplo, salva-se o programa como código fonte. Após, pode-se compilar esse código-fonte, com objetivo de produzir um tipo de arquivo binário chamado de arquivo de classe. Esses arquivos não são executados em seguida porque eles não contêm instruções que são entendidas pelos processadores que estão no mercado. Os programas Java são compilados em um formato intermediário chamado bytecodes. Desta forma, esses programas podem ser executados em qualquer sistema operacional por um interpretador Java, a chamada JVM (Máquina Virtual Java). Assim, o código precisa ser escrito e compilado apenas uma vez, pois os bytecodes gerados serão executados da mesma forma em qualquer plataforma de hardware e software.

A linguagem Java foi projetada para ser orientada a objetos, para ter portabilidade e independência de plataforma, ou seja, "escreva uma vez e execute em qualquer lugar". É uma linguagem simples, de fácil aprendizado ou migração, pois possui um reduzido número de construções. A diminuição das construções mais suscetíveis a erros de programação, tais como ponteiros e gerenciamento de memória via código de programação também faz com que a programação em Java seja mais eficiente. Contém um conjunto de bibliotecas que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica.

Também foi projetada para obter largos recursos de rede, possuindo uma extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP, HTTP e FTP. É segura, pode executar programas via rede com restrições de execução. Além disso, podemos destacar outras vantagens apresentadas pela linguagem como ter uma sintaxe parecida com a do C/C++, suporta nativamente caracteres unicode, tem simplicidade na especificação, tanto da linguagem como no "ambiente" de execução, a máquina virtual (JVM, “Java Virtual Machine”). Também é distribuída

com um vasto conjunto de bibliotecas. Possui facilidades para criação de programas distribuídos e multitarefa (múltiplas linhas de execução num mesmo programa), possui desalocação de memória automática por processo de coletor de lixo e carga dinâmica de código, onde programas em Java são formados por uma coleção de classes armazenadas independentemente e que podem ser carregadas no momento de utilização.

A linguagem java em si possui três ambientes de desenvolvimento:

A. O Java 2 Standard Edition (J2SE), que é um ambiente de desenvolvimento mais utilizado em Servidores e PCs;
B. O Java 2 Enterprise Edition (JEE), amplamente usado para Internet e intranets, redes em geral;
C. O Java Micro Edition (J2ME) um ambiente de desenvolvimento para dispositivos móveis.

A Figura 4 exemplifica como é o funcionamento de uma arquitetura de uma aplicação com JEE.

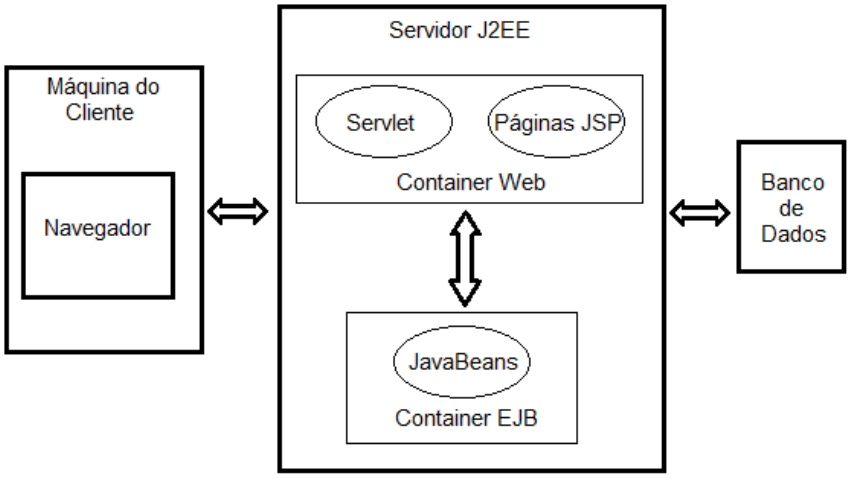

Figura 4 - Arquitetura de uma aplicação com JEE.

Atualmente existem dois exemplos de Maquinas Virtuais Java muito conceituados, a Oracle[15] JVM (*Java Virtual Machine*) e OpenJDK[16] JVM que implementam tais otimizações.

[15] ORACLE, Disponível em <http://www.oracle.com/br>. Acesso em 19 de mar. 2015.
[16] OPENJDK, Disponível em <http://openjdk.java.net/>. Acesso em 19 de mar. 2014

3.1.1 Java Enterprise Edition

Java EE (Java Enterprise Edition) é uma plataforma para desenvolvimento de aplicações Java que oferece a funcionalidade para o desenvolvimento de aplicações de multicamadas baseadas na Web. O Java EE que apresenta um conjunto de especificações com alto nível de detalhamento, que descreve convenções e padronizações de implementações de software bem como usos dos recursos computacionais tais como conexão Banco de Dados, utilização de componentes de Web, acesso remoto, gerenciamento de threads, gerenciamento de conexões HTTP, gerenciamento da sessão Web dentro outros que são utilizadas por desenvolvedores de sistemas Web.

O Modelo de Aplicações Java EE define uma arquitetura para implementação de serviços como aplicações multicamadas que fornecem escalabilidade, acessibilidade e gerenciabilidade necessários para aplicações corporativas. Dessa forma, a lógica de apresentação e a lógica de negócio são implementadas pelo desenvolvedor e os outros serviços são fornecidos pela plataforma Java EE.

Basicamente existem duas aplicações multicamadas Java EE divididas em níveis como:

1. Camada do Cliente: componentes são executados na Máquina Cliente;
2. Camada Web: componentes são executados no Servidor Java EE;
3. Camada de Negócio: componentes são executados no Servidor Java EE;
4. Camada EIS: software executado no Servidor EIS.

3.2 JBoss 4.0

O servidor de aplicação JBoss é baseado na plataforma JEE e implementado na linguagem de programação Java, e por isso ele pode ser utilizado em todos sistema operacionais. Uma das características de grande relevância do JBoss é ter código fonte aberto (JBoss Developers, 2015).

A ideia básica por trás do código aberto é simples: possibilitar que os programadores possam ler, redistribuir e modificar o código-fonte da maneira que for necessária, inclusive ajudar o software a evoluir. Pessoas também podem melhorá-lo, adaptá-lo a suas necessidades e ajudar a corrigir bugs.

O JBoss Application Server (JBoss AS) 4.0 é pertencente ao Java Enterprise Edition (JEE). É sucessor do JBoss 3.2 que teve grande sucesso. A nova versão obteve melhoras no cumprimento das normas e obteve grandes aprimoramentos de recursos. JBoss AS 4.0 oferece o mesmo nível de qualidade e estabilidade para seus usuários (JBoss Developers, 2015). As principais características do JBoss AS 4.0 incluem:

A. Foi oficialmente certificado para ser totalmente compatível com a especificação JEE 1.4. JBoss AS 4.0 é o primeiro servidor de aplicações JEE pronto para a produção de 1,4 na indústria.
B. Suporte bem completo para JEE Web Services e Arquitetura Orientada a Serviços (SOA).
C. Suporta o modelo de Programação Orientada a Aspectos (AOP) para o desenvolvimento de soluções de middleware. JBoss AOP melhora muito a produtividade do desenvolvedor.
D. Integra muito bem com o Hibernate, o mais popular framework de persistência objeto do mundo desenvolvido pela JBoss, no interior do recipiente servidor de aplicativos.
E. Melhora clustering e suporte de armazenamento em cache distribuído com uma nova arquitetura de cache interno.

3.3 JSF 2

O JSF como dito anteriormente, é um framework MVC baseado em Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, fazendo uma abstração dos detalhes da manipulação dos eventos e organização dos componentes, e desta forma deixando que o programador se concentre na lógica da aplicação.

O JSF foi projetado para ser flexível, é uma tecnologia que aproveita a interface de usuário padrão existente do Java e conceitos da Web sem limitar os desenvolvedores a uma língua menos conhecida, nem mesmo a protocolos diferentes (ORACLE, 2015).

As classes de componentes de interface do usuário são incluídos com a tecnologia do JSF que encapsula as funcionalidades dos componentes, permitindo

assim que várias interfaces possam ser prestadas a vários dispositivos cliente, uma tipicidade abstraída pela arquitetura MVC. Ao combinar a funcionalidade dos componentes de interface do usuário com renderizadores personalizados, que definem os atributos de renderização para um componente de interface específica, os desenvolvedores podem construir marcas personalizadas para um dispositivo cliente particular. Além disso, a tecnologia JavaServer Faces oferece uma biblioteca de tags JSP[17] para renderizar de forma personalizada a cada tipo de cliente de HTML, permitindo que os desenvolvedores de Java Platform, Enterprise Edition (Java EE) possam usar a tecnologia JavaServer Faces em suas aplicações sem se preocuparem com adaptações.

Segundo Luckow e Melo (2010), o principal objetivo do JSF é a facilidade de uso, a arquitetura JavaServer Faces define claramente uma separação entre a lógica da aplicação e apresentação, enquanto torna mais simples a conexão com a camada de apresentação do aplicativo. Este projeto permite que cada membro de uma equipe de desenvolvimento de aplicações web possam se concentrar em sua parte do processo de desenvolvimento, e também fornece um modelo de programação simples para ligar as peças. Por exemplo, os desenvolvedores de páginas web sem conhecimentos de programação utilizando das tags de interface do usuário fornecidas pelo JSF podem construir o código do aplicativo sem escrever qualquer script.

O JSF foi desenvolvido através da comunidade Java estabelecendo, a partir daí, o padrão para a construção de interfaces de usuário. Com as contribuições dessa comunidade, o JavaServer Faces foi sendo projetado de modo que possa ser aproveitados por ferramentas que irão tornar o desenvolvimento de aplicações web ainda mais fácil. Como exemplo, as visões do JSF são apresentadas, utilizando arquivos XML[18] chamados de modelos de visão ou Facelets Views. Nestes, os pedidos são processados pelo FacesServlet, que carrega o modelo de visão correto, define os componentes, processa os eventos e mostra a resposta, em linguagem HTML, para o cliente. O estado de componentes de interface do usuário e outros

[17]JSR 245, Disponível em <https://jcp.org/en/jsr/detail?id=245>. Acesso em 19 de mar. 2015.
[18] XML, Disponível em <http://www.w3schools.com/xml/>. Acesso em 19 de mar. 2015.

objetos de interesse de escopo é salvo no final de cada pedido em um processo e restaurado na próxima criação desta visão. Objetos e estados podem ser salvos ou no cliente ou no servidor, outra característica importante que deixa o JSF com maior portabilidade.

Algumas características se sobressaem no JSF, a permissão que o desenvolvedor crie interfaces de usuário através de um conjunto de componentes UIs (Interface de usuário) pré-definidos é uma delas, temos outras como o conjunto de tags JSP fornecidas para acessar os componentes e esses mesmos componentes da página podem ser reutilizados. O JSF associa os eventos do lado cliente com os manipuladores dos eventos do lado do servido, ou seja, os componentes de entrada possuem um valor local representando o estado no lado do servidor. Traz separação de funções que envolvem a construção de aplicações Web e utiliza AJAX[19] em alguns de seus componentes tornando alguns processos mais rápidos e eficientes, algo importante em aplicações web onde os recursos da Internet exigem maiores avanços dos aplicativos web.

O JSF é atualmente considerado pela comunidade Java como um framework mais completo em termos de desenvolvimento de aplicações Web utilizando Java, resultado da experiência e maturidade adquiridas com o JSP/Servlet, Struts, etc. Não se pode deixar de evidenciar que o JSF é um conjunto de "facilitadores" criados em cima de servlet’s e JSP.

3.4 Servlet

Segundo (JCP COMMUNITY, 2015), Servlet (Pequeno servidor em português) é uma classe Java usada para estender as funcionalidades de um servidor. Apesar desses pequenos servidores poderem responder a quaisquer tipos de requisições, são usados para ampliar as aplicações hospedadas por servidores web, desta forma eles se parecem com Applets Java que executam em servidores em vez de executarem nos navegadores web. Estes tipos de servlets são equivalentes à outras tecnologias de conteúdo Web dinâmico, como PHP[20] e

[19] AJAX, Disponível em <http://www.w3schools.com/ajax/>. Acesso em 19 de mar. 2015.
[20] PHP, Disponível em <http://php.net/>. Acesso em 19 de mar. 2015.

ASP.NET[21]. Servlets geram dados HTML e XML para a camada de apresentação de uma aplicação Web. Ele processa dinamicamente requisições e respostas.

Essa tecnologia disponibiliza ao programador da linguagem Java uma interface para o servidor web, através de uma API (Interface de Programação de Aplicativos). As aplicações baseadas no Servlet geram conteúdo dinâmico (normalmente HTML) e interagem com os clientes, utilizando o modelo requisição-resposta. Normalmente utilizam o protocolo HTTP, apesar de não serem restritos a ele.

Outro fato é que um Servlet necessita de um container Web para ser executado, desta forma ele é frequentemente usado para processar ou armazenar dados que foram enviados de um formulário HTML fornecendo conteúdo dinâmico, como os resultados de uma consulta a um banco de dados.

3.5 JSP

Uma das tecnologias utilizadas pelo JSF 2.0 é o JSP (Java Server Pages), que foi difundido em 1999 pela Sun Microsystems. O JSP é uma tecnologia usada no desenvolvimento de aplicações para Web, parecido com as tecnologias Active Server Pages (ASP) da Microsoft ou PHP. Esta tecnologia permite o uso de códigos Java em páginas HTML.

Quando uma página JSP é requisitada pelo cliente através de um Navegador, esta página é executada pelo servidor, e a partir daí será gerada um texto em HTML que será enviada de volta ao navegador do cliente (Figura 5).

[21] ASP, Disponível em <http://www.asp.net/>. Acesso em 19 de mar. 2015.

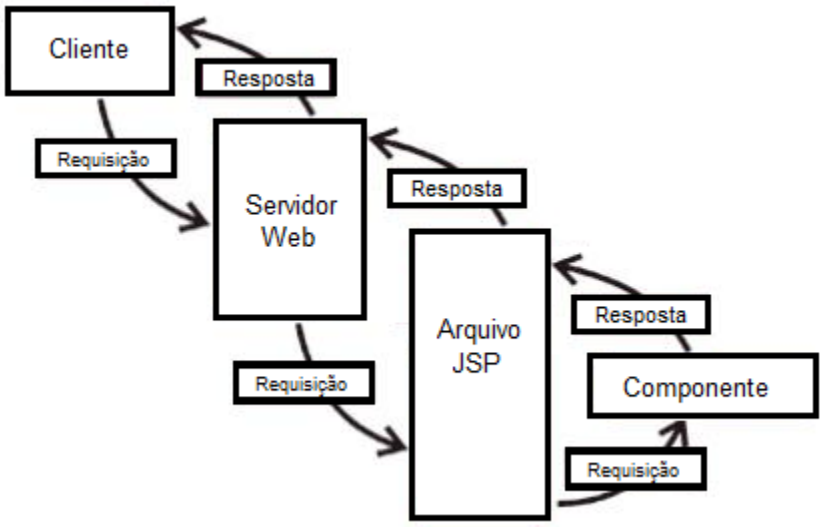

Figura 5 - Exemplo do funcionamento de um arquivo JSP.

A tecnologia JSP tem a vantagem da portabilidade de plataforma por ser baseada na linguagem de Java, permitindo a sua execução em qualquer sistema operacional com portabilidade para esta linguagem. Esta tecnologia também permite ao programador web produzir aplicações que necessitem de acesso ao banco de dados, manipulação de arquivos no formato texto, captura de informações a partir de formulários e captação de informações sobre o visitante e sobre o servidor.

3.6 PrimeFaces

O PrimeFaces é um framework para a criação de interfaces amplamente utilizado no mundo pelos desenvolvedores Java. Segundo Venturini e Marchi (2011, citado por Carmisini e Vahldick1(2012)):

> O PrimeFaces oferece um conjunto de componentes com versões estáveis e de código aberto para o JSF 2.0 e permite que sejam inseridos em seu conjunto, outros componentes através de especificações em JSF. Ele está organizado em três módulos:
>
> a) UserInteface (UI) Components – compreende os componentes que contém as funcionalidades encapsuladas de AJAX, Javascript e gráficos animados;
>
> b) Optimus – módulo que oferece soluções para facilitar desenvolvimento com JSF. Também contém componentes de extensões de segurança;

c) FacesTrace – módulo encarregado das funções relacionadas ao desempenho das aplicações baseadas em JSF.

As vantagens de se usar o PrimeFaces para a criação de interfaces são claras, segundo PrimeFaces (2012 ,citado por Carmisini e Vahldick1(2012)):

a) Simplicidade e desempenho: PrimeFaces é leve, todas as decisões tomadas são baseadas em manter o PrimeFaces tão leves quanto possível.

b) Facilidade de uso: os componentes do PrimeFaces são desenvolvidos com um princípio de design que afirma que “Uma boa UI deve ocultar a complexidade, mas manter a flexibilidade”;

c) Possui o Mobile UI kit para criar aplicações web móveis para dispositivos portáteis baseados em navegadores web kit, por exemplo, IPhone, Palm, AndroidPhones, Nokia S60 entre outros;

d) Possui um rico conjunto de componentes de interface: DataTable, AutoComplete, HTMLEditor, gráficos e etc.;

e) Não é necessária nenhuma configuração extra de XML e não há dependências;

f) Os componentes são construídos com AJAX no padrão JSF 2.0; g) Possui mais de 30 temas de templates;

h) Possui boa documentação com exemplos práticos.

Ainda, a biblioteca componente PrimeFaces para o JSF, desenvolvido pela Prime Technology é disponibilizada pelo servidor de aplicações JBoss, fornecendo um conjunto de componentes visuais baseadas em AJAX para deixar as páginas mais interativas com o usuário.

3.7 Banco de Dados

Software usualmente necessitam de armazenar os dados em que trabalham, ou seja, necessitam de persisti-los para futuras consultas, inserções, exclusões e alterações dos mesmos, além dessas necessidades esses dados devem se apresentar de maneira organizada para facilitar essas operações, para isso, em sua grande maioria, os software utilizam os chamados banco de dados.

Os bancos de dados, por sua vez, são gerenciados por software chamados de SGBDs (Sistema Gerenciador de Banco de Dados), que tem o objetivo de prover maneiras para que um banco de dados possa realizar as operações de criação, inserção, leitura e deleção de forma transparente para o usuário, sempre

respeitando o famoso acrônimo ACID (Atomicidade, Consistência, Integridade e Disponibilidade) em suas aplicações.

A maioria dos SGBD que existem no mercado fazem uso de um modelo de estruturação e organização de seus dados chamado modelo relacional. O modelo relacional surgiu com revolução e é largamente utilizado até os tempos de hoje pelos SGBDs. No Modelo Relacional os dados são separados e organizados de acordo com suas correlações em tabelas que permitem relacionamento de registros para que se possa realizar as operações de consultas. Uma das grandes vantagens do uso dos bancos de dados é a sua independência em relação as aplicações, ou seja, a existência banco de dados não é necessariamente atrelado a existência de uma aplicação. No contexto atual dos SGBDs existem tanto pagos, quanto gratuitos. Abaixo alguns do mais utilizados atualmente:

A. SQLServer[22]
B. MySQL[23]
C. PostGresSQL[24]
D. FireBirdSQL[25]
E. mSQL[26]
F. Oracle Database[27]

3.7.1 MySQL

O MySQL é um SGBD que utiliza a linguagem SQL[28] (Structure Query Language - Linguagem de Consulta Estruturada) e segundo (DB-Engines-Rank[29], 2015), é atualmente um dos bancos de dados mais populares do mercado com mais de 10 milhões de instalações pelo mundo. O MySQL foi criado em 1996 pela a empresa TcX[30] com o intuito de se ter um banco de dados que apresentasse segurança, agilidade e que necessitasse de requisitos de hardware com baixo

[22] MICROSOFT, Disponível em <http://www.microsoft.com/>, Acesso em 19 de mar. 2015.
[23] MYSQL, Disponível em <http://www.mysql.com/>. Acesso em 19 de mar. 2015.
[24] PostGresSQL, Disponível em <http://www.postgresql.org/>. Acesso em 19 de mar. 2015.
[25] FIREBIRDSQL, Disponível em <http://www.firebirdsql.org/>. Acesso em 19 de mar. 2015.
[26] mSQL, Disponível em <http://xoops.net.br/docs/ref.msql.php>. Acesso em 19 de mar. 2015.
[27] ORACLE, Disponível em<http://office.microsoft.com/pt-br/access/>. Acesso em 19 de mar. 2015.
[28] SQL, Disponível em <http://www.w3schools.com/sql/>. Acesso em 19 de mar. 2015.
[29] DB-Engines, Disponível em <http://db-engines.com/en/ranking>. Acesso em 19 de mar. 2015.
[30] TcX, Disponível em <http://www.tcx.se/>. Acesso em 19 de mar. 2015.

custo, hoje é atualmente desenvolvido pela MySQL AB, na qual apresenta uma versão gratuita e uma versão paga, é um SGBD multiplataforma, sendo compatível com os sistemas operacionais Windows e sistemas baseados em Linux. O MySQL é usado na comunicação entre as aplicações e os dados disponibilizando métodos de consulta, gravação e alteração dos dados que se apresentam sobre diversos tipos de campos.

Os maiores atrativos do MySQL é ser rápido, leve, simples, prático, seguro, além de ter código fonte aberto e possuir versão gratuita. Essas características o colocam bem próximo a grandes SGBDs como o Oracle Database.

3.7.2 Java Database Connectivity

O JDBC (Java Database Connectivity) segundo (Glossário[31] JDBC, 2015), é uma especificação para conexão entre os SGBDs e sistemas programados em Java, basicamente ele é formado por um conjunto de classes e interfaces (API) que fazem o envio de instruções SQL para qualquer banco de dados relacional, ou seja, para cada banco de dados há uma conectividade JDBC específica, tornando-o importante para portabilidade entre sistemas gerenciadores de banco de dados diferentes.

3.7.3 Java Persistence API

A JPA[32] é uma API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados, definindo um meio de mapeamento objeto relacional para objetos Java simples e comuns. Vários frameworks de mapeamento objeto relacional como o famoso Hibernate implementam a JPA.

Desta forma, segundo (OpenJPA[33], 2015), para que um sistema possa utilizar as informações presentes em um banco de dados é necessário existir comunicação com uma determinada sintaxe de consulta entre o banco de dados e o sistema, evitando, então, a diversidade de tipos de sintaxes entre a linguagem

[31] Glossário JDBC, Disponível em <http://mindprod.com/jgloss/jdbc.html>. Acesso em 19 de mar. 2015.
[32] JPA, Disponível em <https://jcp.org/en/jsr/detail?id=317>. Acesso em 19 de mar. 2015.
[33] OpenJPA, Disponível em <http://openjpa.apache.org/>. Acesso em 19 de mar. 2015.

Java e os vários tipos de SGBDs. A especificação além de padronizar os diferentes tipos de sintaxe ainda provê recursos para problemas relacionados a incompatibilidade de paradigmas.

3.7.4 Hibernate

O Hibernate é um framework que age basicamente na camada de persistência de um sistema pois é uma implementação da especificação JPA, é uma ferramenta ORM, ou seja, que implementa o mapeamento objeto relacional para a linguagem Java, e também provê recursos para a consulta, gravação e alteração de dados, sendo um mediador entre a aplicação e o banco de dados. É muito utilizado para mapear classes Java em tabelas do banco e vice-versa.

Desta forma, o Hibernate concede um forte mecanismo para manipulação de dados, permitindo uma redução considerável no tempo de desenvolvimento da aplicação.

3.8 JAAS

JAAS (Java Authentication and Authorization Service) é um conjunto de API's que servem para desassociar as aplicações dos controles de acesso a recursos do sistema. Na prática retira-se do desenvolvedor a responsabilidade de ter que controlar o acesso a recursos por perfis, todo o controle passa a ser feito de maneira declarativa no descritor da aplicação. O JAAS é mais um importante recurso disponibilizado pelo servidor de aplicações JBoss.

Desenvolver uma aplicação que restringe o acesso de usuários ao sistema e implementar controles de segurança não é uma tarefa muito fácil, e fazer isso de um jeito que a aplicação seja totalmente independente da tecnologia de segurança utilizada é ainda mais difícil. É exatamente isso que o JAAS tem por objetivo, tirar do desenvolvedor a grande responsabilidade de ter que projetar previamente todo o sistema com base no controle de acesso a recursos do sistema e passar de maneira declarativa no descritor da própria aplicação.

Deste modo, a grande vantagem do JAAS é fazer com que o desenvolvedor passe a se preocupar somente com a sua aplicação, as restrições a determinados recursos não vão mais interferir na maneira como se desenvolve a aplicação, o

desenvolvedor pode fazer as devidas declarações de quais recursos serão protegidos e quem poderá acessá-los após concluir o sistema. Isso deixará o código do sistema mais claro e ligado nas regras de negócio. Sem contar que qualquer modificação nas restrições implicará na mudança de apenas um arquivo, e não mais em diversos pontos espalhados pela aplicação como é feito em sistemas legados.

3.9 Eclipse

Para Santos (2007), as IDEs (Integrated Development Environment - Ambientes de Desenvolvimento Integrado) são ferramentas para auxiliar os desenvolvedores de software, garantindo diminuição de erros, aumento na produtividade e consequentemente melhores produtos.

Para Luckow (2010) a IDE Eclipse:

> Não é apenas uma das principais IDEs Java, bem além disso, o projeto é uma referência e um caso de sucesso no desenvolvimento de software de um modo geral. Atualmente com pouco mais de 10 anos de vida, o Eclipse hoje é um ecossistema composto por vários projetos de diversas características. Um ponto extremamente importante relacionado ao Eclipse é o modelo open source. O Eclipse atingiu o patamar atual, principalmente pela força e influência da comunidade open source. Indo um pouco além, o Eclipse é a prova real de que desenvolvimento de software open source funciona, e muito bem.

Uma das grandes vantagens da utilização da IDEs Eclipse é a sua adaptação a diversas maneiras e metodologias de desenvolvimento através dos vários plugins que podem ser incorporados a ele.

4 SISTEMA DESENVOLVIDO

De acordo com os objetivos do sistema descritos no capítulo 1 e utilizando as ferramentas apresentadas no capitulo 3, foi desenvolvido o sistema denominado **SNDTNA** que conta com telas simples e intuitivas dentro de um escopo a fim de trazer uma melhor experiência para o usuário.

A figura 6 ilustra como o escopo do sistema é dividido.

Figura 6 - Escopo do Aplicativo Web SNDTNA.

A. Cabeça (Parte superior): Composta pelo logotipo do aplicativo web e menus que mapeam e possibilitam trafegar entre as páginas trazendo uma melhor visão como um todo para o usuário.
B. Conteúdo (Parte medial): Nesta parte é onde cada página (View) é renderizada sempre acompanhando os menus que indicam a função das mesmas. É a única parte do escopo que muda totalmente sua estrutura e conteúdo de acordo com as requisições do usuário.
C. Rodapé (Parte inferior): No rodapé é disponibilizado informações úteis sobre o aplicativo web.

4.1 Funcionalidades

O sistemas possui funcionalidades que se auto incrementam, ou seja, o usuário logado possui permissão para realizar todas as funcionalidades que o usuário comum possui, assim como o administrador também possui permissão para realizar todas as funcionalidades que o usuário comum e logado possuem.

1- As funcionalidades dos usuários comuns são:

A. O usuário corrente pode utilizar das funcionalidades de busca através da página inicial (Home) por veículos fornecendo os filtros como fabricante, modelo, ano do veículo e mais.
B. O usuário não cadastrado pode se cadastrar.
C. O cadastro de usuário é necessário para ter acesso a outras funcionalidades do sistema.
D. Após cadastrado, o usuário pode fazer “login”.

2- As funcionalidades para os usuários logados são:

A. O usuário cadastrado e logado pode cadastrar um veículo afim de anunciar a sua venda fornecendo dados, como preço, fotos e mais.
B. O usuário cadastrado e logado pode editar os veículos que por ele foi cadastrado.
C. O usuário pode editar seus veículos de acordo com sua preferência, ou caso tenha fornecido algum dado errado.

3- As funcionalidades para os administradores são:

A. O administrador logado pode editar qualquer veículo.
B. Criar, atualizar e remover fabricante.
C. Criar, atualizar e remover modelo.
D. Criar, atualizar e remover usuário.
E. Criar, atualizar e remover cores a serem selecionadas no cadastro de veículos.

4.2 Diagrama de casos de uso

O diagrama de casos de uso do SNDTNA tem por finalidade mostrar toda interação possível do usuário com o sistema, com objetivo maior o entendimento em alto nível do mesmo.

A figura 7 mostra o diagrama de casos de uso levantados na aplicação.

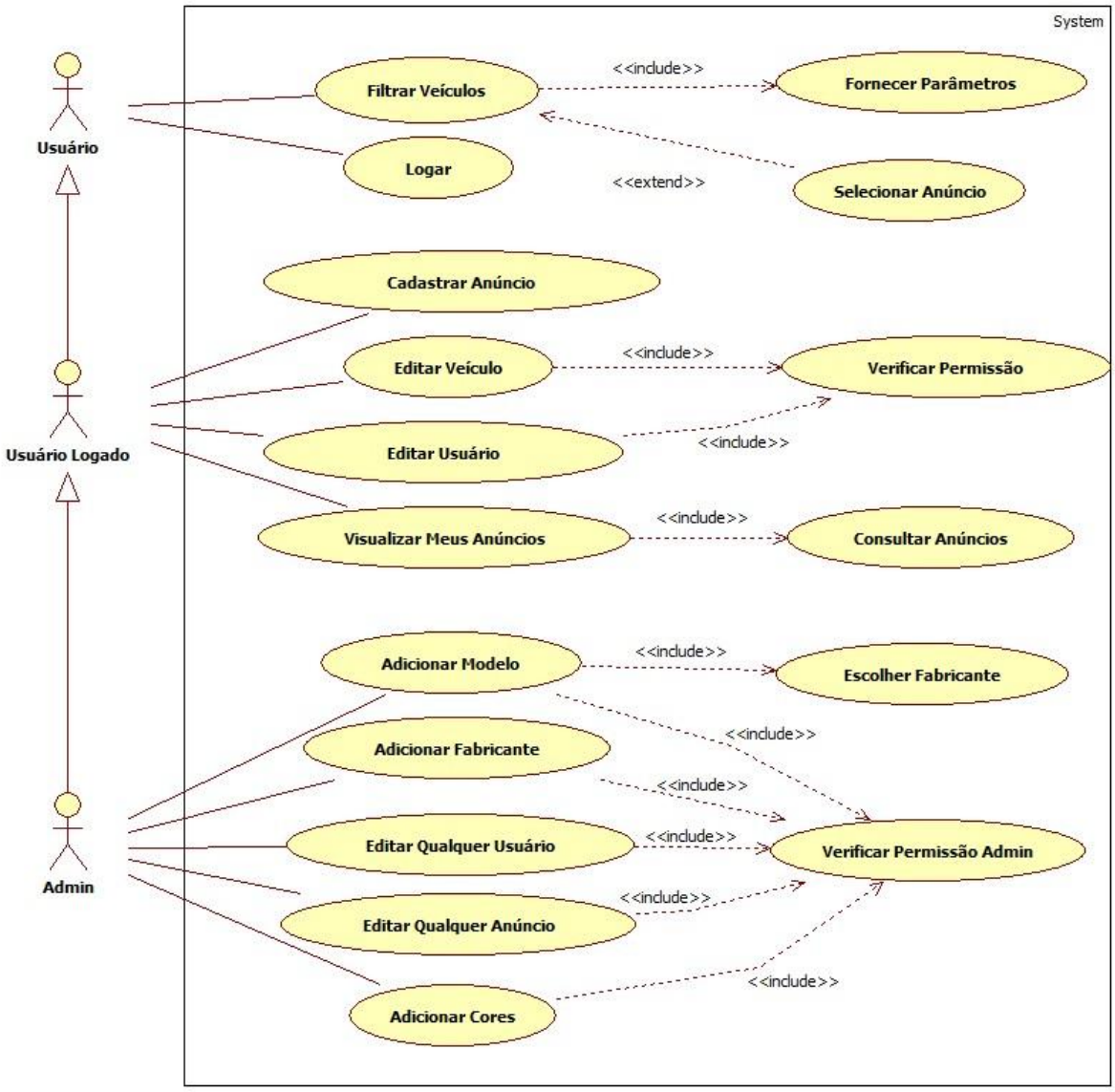

Figura 7 - Diagrama de casos de uso.

4.3 Diagrama do modelo relacional do banco de dados

O modelo relacional do banco de dados nos mostra todas as tabelas e seus relacionamentos e seus campos. Através deste modelo é possível obter informações importantes para o desenvolvedor caso haja necessidade de uma manutenção no banco de dados.

A figura 8 mostra o modelo relacional dessa aplicação.

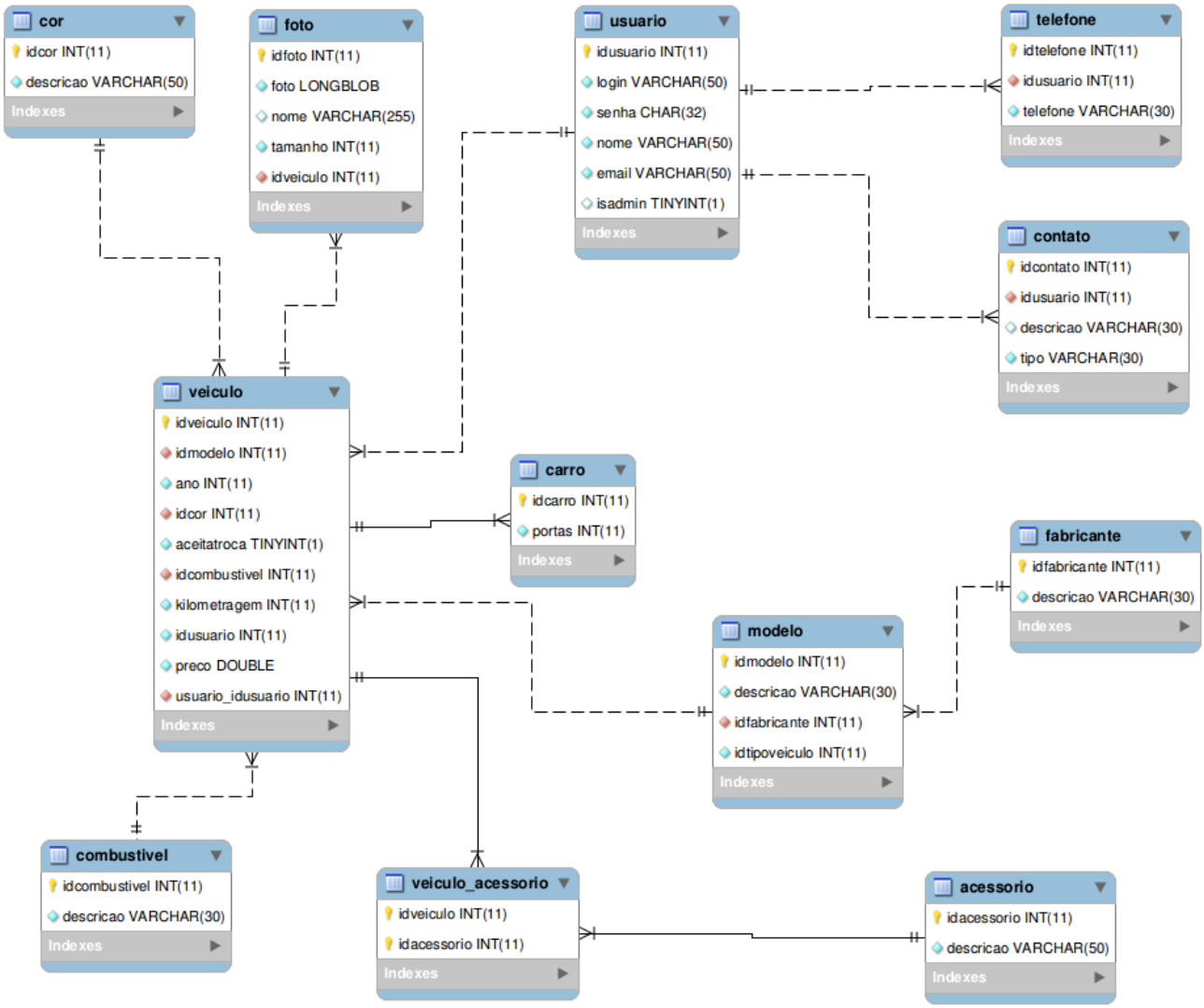

Figura 8 - Modelo Relacional.

4.4 Diagrama de Classe

Diagrama de classes do sistema ocultando métodos “getters and setters” para uma melhor visualização do diagrama como um todo. O diagrama de classe tem como objetivo o melhor entendimento de todas as classes implementadas no sistema assim como seus métodos mais importantes.

A figura 9 mostra o diagrama de classes da aplicação.

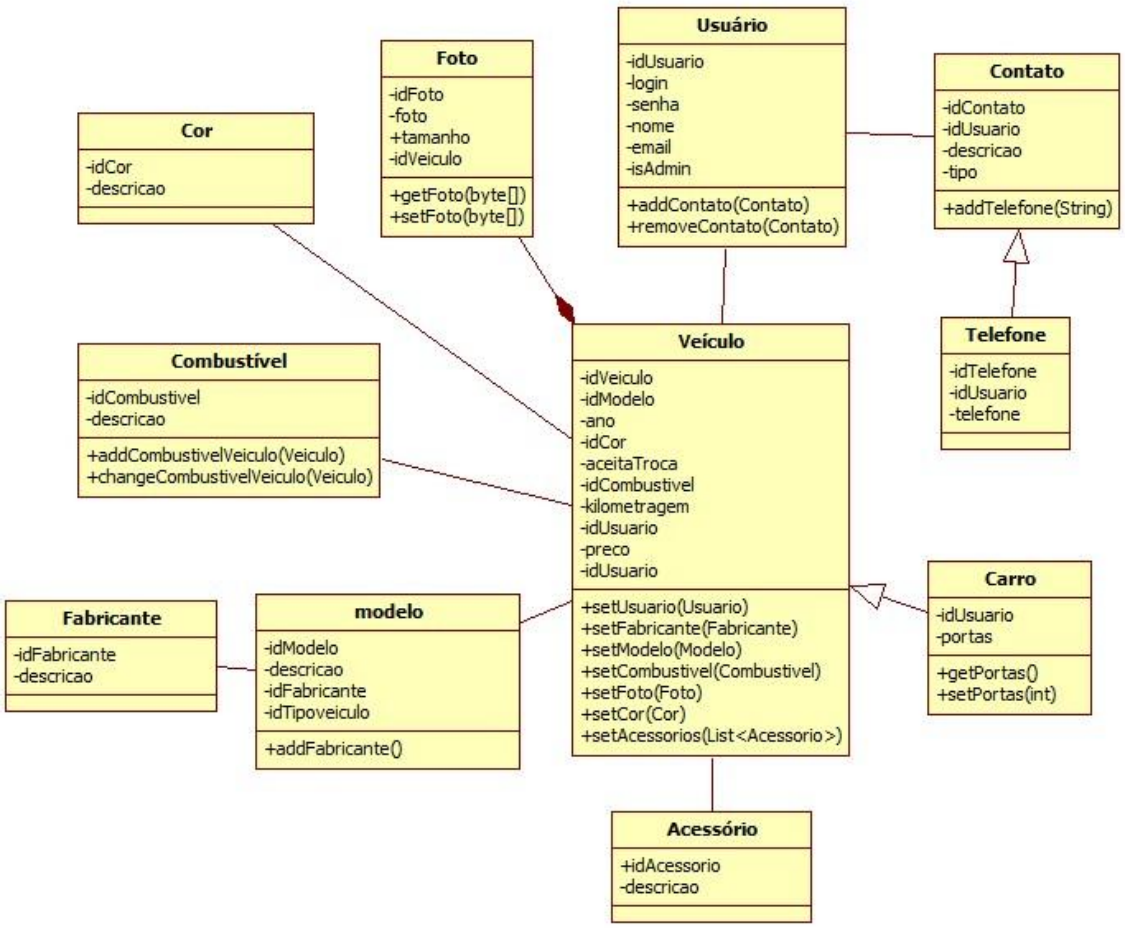

Figura 9 - Diagrama de classes.

4.5 Telas do sistema

Nesse capitulo será apresentado de forma detalhada o funcionamento do sistema através de imagens do próprio sistema.

4.5.1 Página Inicial:

Para atender a especificidade deste trabalho esta página é dividida em duas partes horizontalmente. A lateral esquerda conta com filtros de diversos tipos que atendem à demanda maior sobre os anúncios. A lateral direita é atualizada de acordo com as requisições do filtro e exibem os anúncios filtrados.

A figura 10 mostra a página inicial.

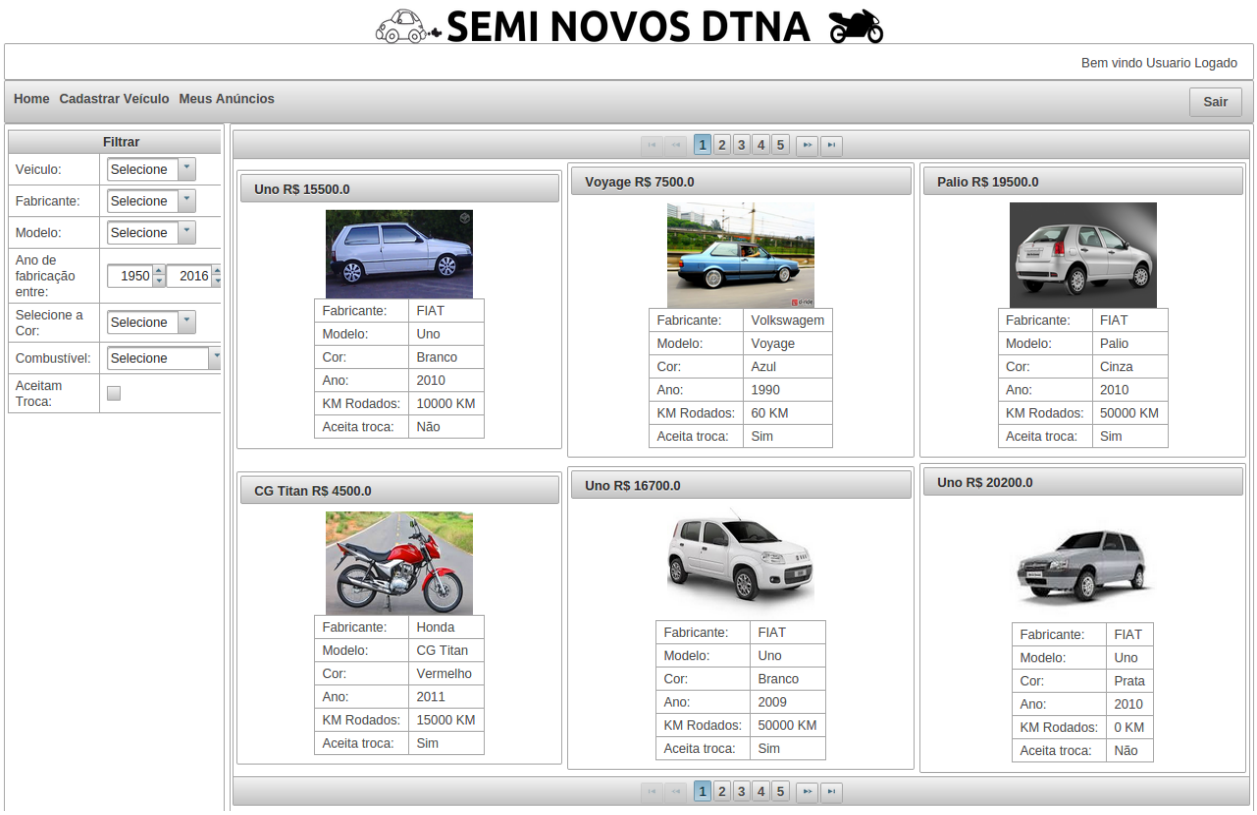

Figura 10 - Página inicial.

Tabela 1: Descrição dos filtros da página inicial.

Nome	**Descrição**	**Tipo de entrada**
Veículo	Tipo do veículo a ser filtrado	Caixa de seleção
Fabricante	Fabricante do veículo a ser filtrado	Caixa de seleção
Modelo	Modelo do veículo a ser filtrado	Caixa de seleção
Cor	Cor do veículo a ser filtrado	Caixa de seleção
Ano	Ano do veículo a ser filtrado	Caixa de seleção
Combustível	Combustível do veículo a ser filtrado	Caixa de seleção
Aceita Troca	Não filtrar veículos que aceitam troca	Caixa de marcação

4.5.2 Cadastro de usuário:

Esta página dá ao usuário a opção de se cadastrar aceitando os termos, condições de uso e após pode informar seus dados.

A figura 11 mostra a tela de cadastro de usuário.

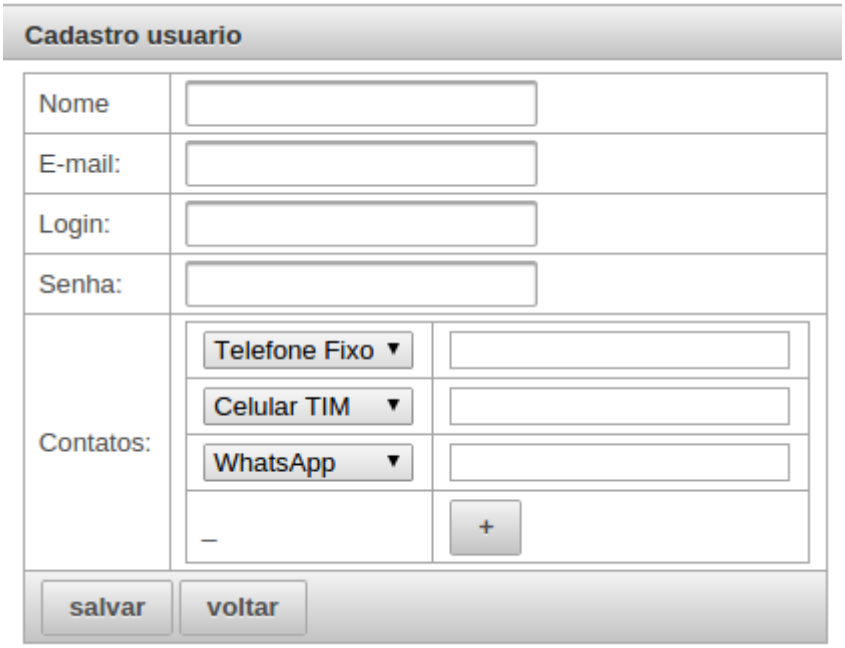

Figura 11 - Tela de cadastro de usuário.

Tabela 2 - Descrição de campos da tela de cadastro de usuário.

Nome	**Descrição**	**Tipo de entrada**
Nome	Nome do usuário a ser cadastrado	Alfanumérico
E-mail	E-mail do usuário a ser cadastrado	Alfanumérico
Login	Login do usuário a ser cadastrado	Alfanumérico
Senha	Senha do usuário a ser cadastrado	Alfanumérico
Tipo do contato	Tipo do contato do usuário a ser cadastrado	Caixa de seleção
Contato	Contato do usuário a ser cadastrado	Alfanumérico
+	Adicionar campo para inserir mais contatos	-
Salvar	Realiza o cadastro do usuário	-
Voltar	Volta para tela anterior	-

4.5.3 Cadastro de veículo:

Após ter logado com seu nome de usuário e senha, o usuário pode acessar os recursos para anunciar seu veículo especificando os detalhes do mesmo nesta página.

A figura 12 mostra a tela de cadastro de veículo.

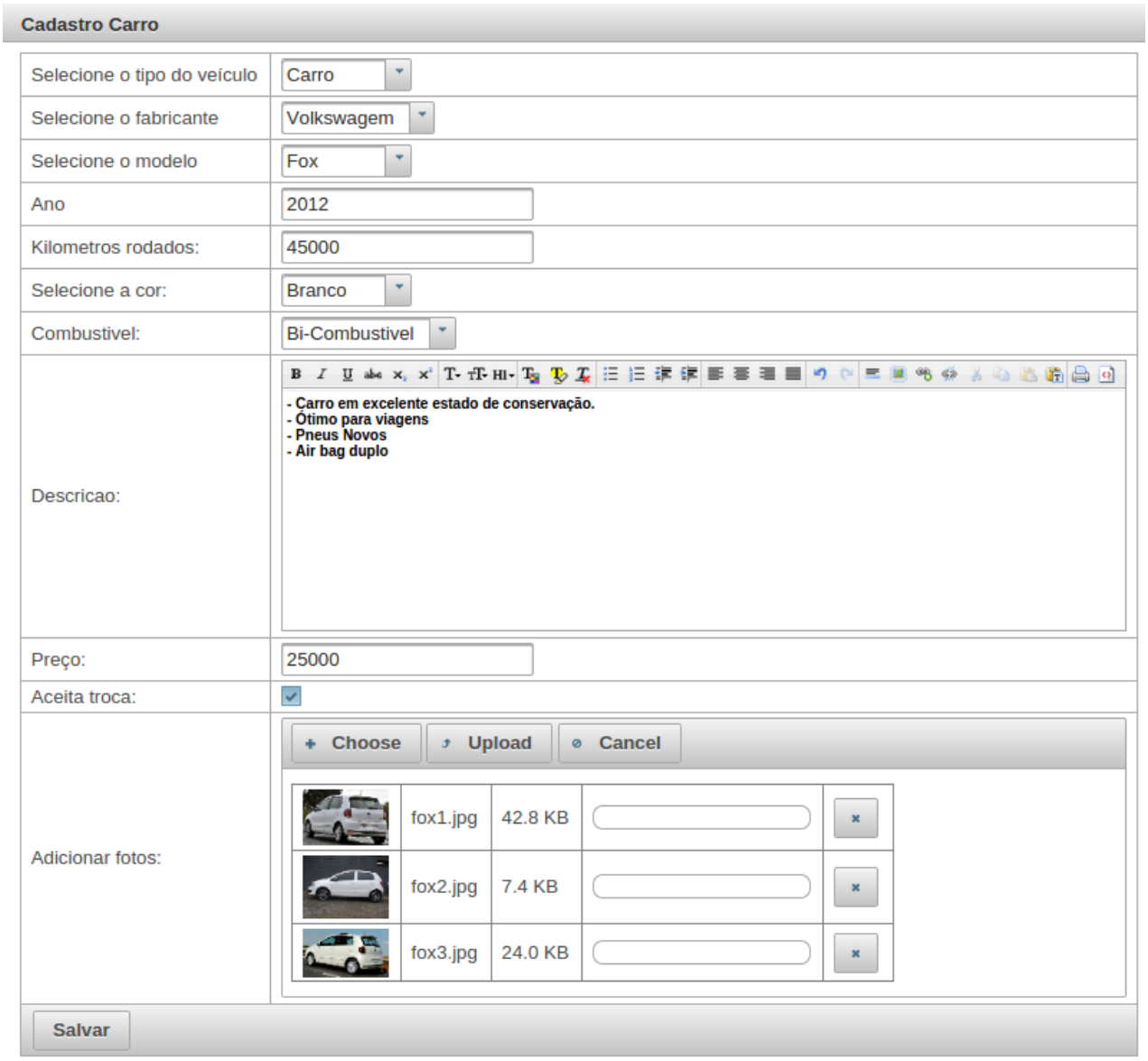

Figura 12 - Tela de cadastro de veículo.

Tabela 3 - Descrição de campos da tela de cadastro de veículo.

Nome	**Descrição**	**Tipo de entrada**
Tipo do veículo	Tipo do veículo a ser cadastrado	Caixa de seleção
Fabricante	Fabricante do veículo a ser cadastrado	Caixa de seleção
Modelo	Modelo do veículo a ser cadastrado	Caixa de seleção
Ano	Ano do veículo a ser cadastrado	Caixa de seleção
Kilometragem	Kilometragem do veículo a ser cadastrado	Número Inteiro
Cor	Cor do veículo a ser cadastrado	Caixa de seleção
Combustível	Combustível do veículo a ser cadastrado	Caixa de seleção
Preço	Preço do veículo a ser cadastrado	Número flutuante
Aceita troca	Veículo aceita ou não troca	Caixa de marcação
Fotos	Fotos do veículo a ser cadastrado	Arquivo de imagem
Salvar	Realiza o cadastro do veículo	-

4.5.4 Meus anúncios:

Aqui o usuário tem acesso aos seus anúncios podendo selecionar para edição caso necessário.

A figura 13 mostra a tela meus anúncios.

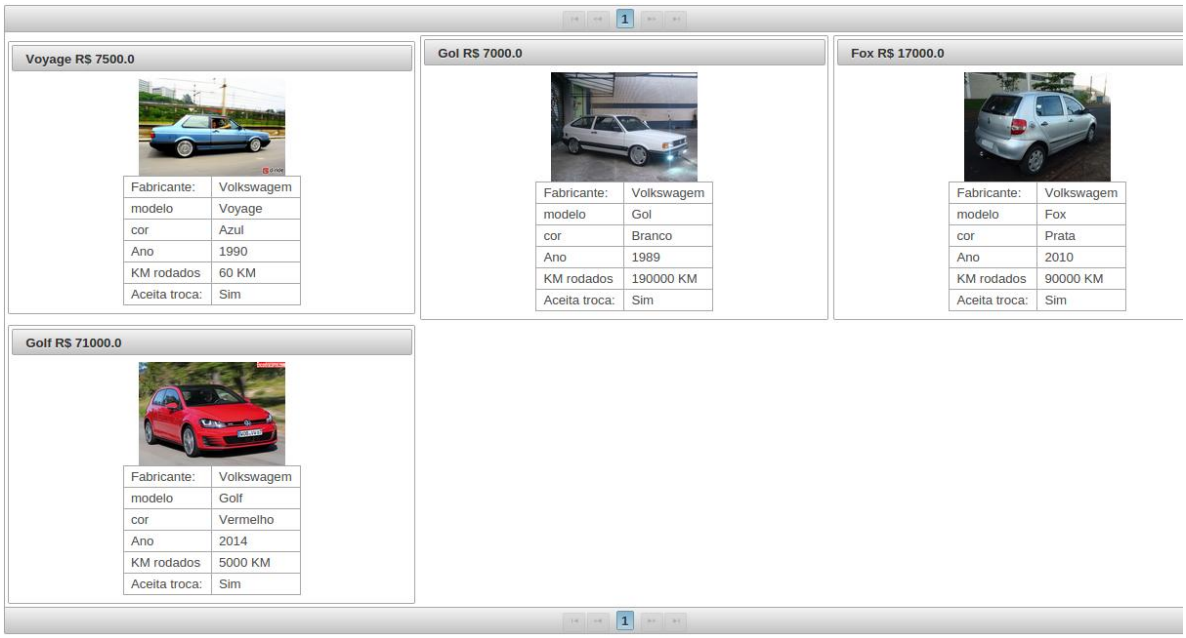

Figura 13 - Tela meus anúncios.

Tabela 4 - Descrição de campos da tela meus anúncios.

Nome	**Descrição**	**Tipo de entrada**
Anúncio	Direciona para o anúncio do veículo	-

4.5.5 Detalhes do veículo:

Nesta página o usuário tem acesso a maiores detalhes sobre o veículo procurado bem como contato do vendedor e tipos de negociações aceitas pelo vendedor. O usuário pode editar o veículo caso tenha sido anunciado por ele próprio. A figura 14 mostra a tela de detalhes do veículo.

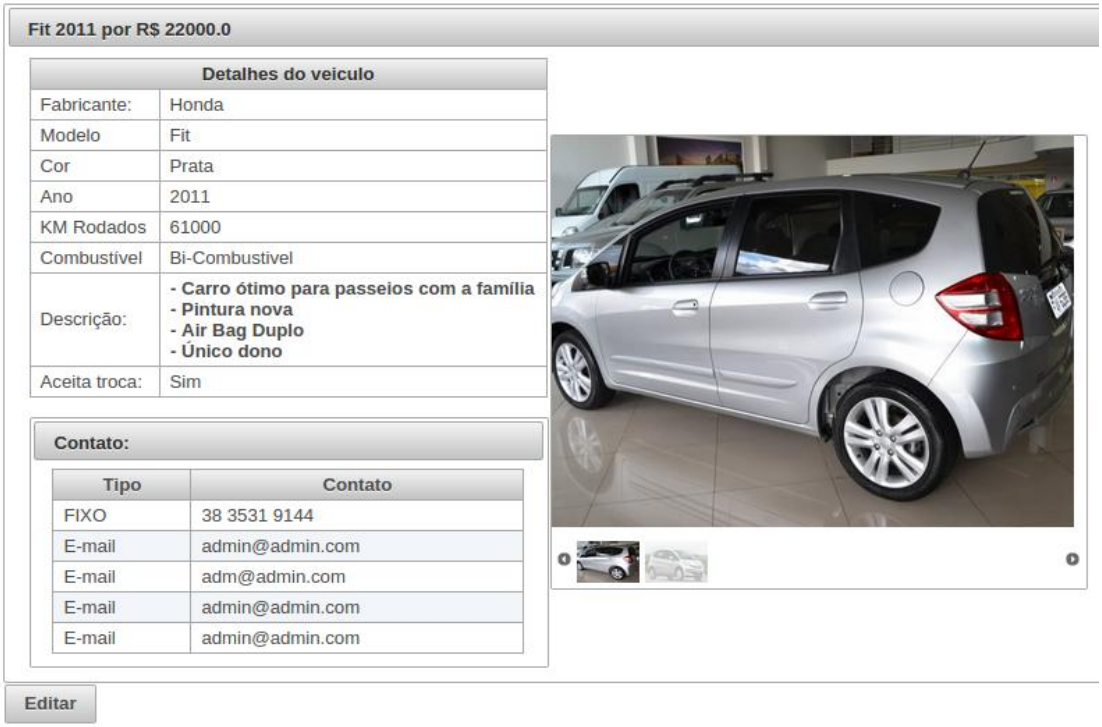

Figura 14 - Tela de detalhes do veículo.

Tabela 5 - Descrição de campos da tela meus anúncios.

Nome	Descrição	Tipo de entrada
Imagens	Troca a imagem do anúncio do veículo	-

5 TESTES REALIZADOS

Para o sistema SNDTNA realizou-se testes para verificar o seu correto funcionamento, com foco nas relações de persistência de dados e entrada de dados em cadastros funcionais. Neste capitulo são descritos todos os testes bem como os resultados de cada um.

Tabela 6 – Pontos de função dos testes realizados.

Ponto de função	**Operação de teste**	**Resultado**	**Avaliação**
Filtragem de veículos	Preenchimento de todos os campos de filtro.	Filtragem concluída com sucesso	Positiva
Cadastro de usuário	Preenchimento de todos os campos seguindo do clique no botão “Salvar”.	Usuário Cadastrado com Sucesso.	Positiva
Cadastro de veículo	Preenchimento de todos os campos seguindo do clique no botão “Salvar”.	Veículo Cadastrado com Sucesso.	Positiva
Edição de usuário	Alteração de todos os campos seguindo do clique no botão “Salvar”.	Usuário Alterado com Sucesso.	Positiva
Edição de veículo	Alteração de todos os campos seguindo do clique no botão “Salvar”.	Veículo Alterado com Sucesso.	Positiva

6 CONCLUSÃO

Tendo em vista as necessidades da população em Diamantina por um ambiente voltado para divulgação e comercialização de carros e motocicletas, o SNDTNA se mostrou eficaz para atender esta demanda, todos os requisitos levantados nos casos de uso para uma melhor interação com o usuário foram atendidos e estão em funcionamento juntamente com uma interface simples e intuitiva através dos seus recursos visuais.

A filtragem através do fornecimento dos parâmetros no filtro da página inicial ocorreu de forma rápida com fluidez, possibilitando aos usuários buscarem pelos veículos anunciados de acordo com o tipo de veículo, fabricante, modelo, cor, tipo de combustível, ano e se aceitam troca ou não. Essa interface com o usuário obteve sucesso com o dinamismo dos filtros.

O cadastro de usuário se mostrou completo deixando com que o usuário selecione suas formas de contato de acordo com suas necessidades. Além dos campos usuais de nome, e-mail, login e senha, o usuário anunciante também pôde fornecer os detalhes do contato, ou seja, o WhatsApp[34], Skype[35] ou Facebook[36], ou ainda o seu número de telefone fixo ou número de celular com as respectivas operadoras. Tudo isso para que o usuário que busca por um veículo pudesse contactar de forma mais conveniente o anunciante.

O cadastro de veículo teve sucesso em todos campos de cadastro permitindo que o usuário inserisse todos os dados do veículo, sendo que cada veículo podia ter vinculado mais de uma foto.

A tela dos anúncios do usuário logado se mostrou eficiente para que o anunciante possa navegar de forma mais precisa, aumentando a agilidade de acesso ao seu próprios anúncios, ou seja, o anunciante pôde buscar em uma tela todos os veículos por ele cadastrado.

[34] WhatsApp, Disponível em <http://www.whatsapp.com/>. Acesso em 20 de mar. 2015.
[35] Skype, Disponível em <http://www.skype.com/>. Acesso em 20 de mar. 2015.
[36] Facebook, Disponível em <https://www.facebook.com/>. Acesso em 20 de mar. 2015.

A tela de detalhes do veículo se mostrou simples, exibindo todos os dados do veículo selecionado pelo usuário. Nesta tela também foi exibida a galeria de imagens que o anunciante cadastrou, para que, desta forma, o usuário que buscasse por mais detalhes do veículo pudesse fazer esta consulta.

Tendo em vista todos os casos de uso que foram implementados podemos concluir que este aplicativo web será bastante útil para a cidade de Diamantina, a demanda por um serviço desse tipo terá um impacto positivo para sua população diminuindo a dificuldade de pesquisar e anunciar um veículo dentro deste nicho de mercado na cidade.

Neste trabalho a linguagem de programação Java se mostrou bastante eficiente, a orientação a objetos forneceu um alto grau de reaproveitamento de código e sua vasta comunidade de usuários ajudaram bastante com a troca de informações inerentes ao processo de desenvolvimento.

A framework ORM Hibernate foi de grande ajuda para o mapeamento de classes a partir do modelo relacional do banco de dados, assim como seu forte mecanismo para manipulação de dados, permitindo uma redução considerável no tempo de desenvolvimento da aplicação.

A biblioteca de componentes PrimeFaces ofereceu simplicidade e desempenho para camada de visão do sistema, seu rico conjunto de componentes de interface somados aos recursos embutidos do AJAX ofereceram grande facilidade de uso para interação com usuário.

A flexibilidade do framework JSF 2 se mostrou bastante eficaz neste trabalho, toda construção das camadas de modelo e controladora do sistema tomou por base a linguagem Java, ou seja, não houve limitações em conhecer outra linguagem de desenvolvimento, nem mesmo a protocolos diferente.

As características de leveza, rapidez, segurança e praticidade do MySQL utilizado neste trabalho foram de suma importância. O famoso acrônimo ACID (Atomicidade, Consistência, Integridade e Disponibilidade) obteve êxito trabalhando junto ao Hibernate.

Os maiores desafios encontrados neste trabalho foram relacionados à configurações do ambiente de desenvolvimento que tomou parte do tempo na

integração das várias tecnologias e ferramentas utilizadas, especificamente relacionado às bibliotecas do modelo de persistência de dados.

7 BIBLIOGRAFIA

ACM SIGCOMM, Computer Communication, A Brief History of the Internet, Volume 39, Number 5, October 2009

ALBERTIN, Alberto Luiz. O comércio eletrônico evolui e consolida-se no mercado brasileiro. São Paulo: EAESP/FGV, 2011.

AMBLER, Scott W. Análise de projeto orientado a objeto. 2. ed. Rio de Janeiro: Infobook, 1998.

BIEBER, A. Transportation planning and system analysis em urban transport plannin process. Paris: OECD, 1971.

CAMPIONE, M; WALRATH, K. The Java Tutorial: Object-Oriented Programming for the Internet. SunSoft Press, 2004.

CARMISINI, A.; VAHLDICK, A. Comparativo Entre Frameworks de Javaserver Faces: Apache Tobago, Primefaces e Richfaces. SC. CONTE, 2012.

CORREIA, Carlos; TAFNER, Malcon. Análise orientada a objetos. 2. ed. Florianópolis: Visual Books, 2006.

CORRÊA, Fernando Antônio; VEIRA, Guilherme Alves. Construa uma aplicação 100% OO – Parte 1. Ano 05 – 61 ed. Grajaú-RJ: .Net Magazine, 2009.

DAILTON, Felipini. Empreendedorismo na Internet, o momento é agora. Disponível em: <http://www.e-commerce.org.br/artigos/empreendedor-sucesso.php>. Acesso em: 20 de mar. de 2015.

DB-ENGINES, Disponível em: <http://db-engines.com/en/ranking>. Acesso em: 20 de maio. 2015.

DEITEL, H. M. Java Como Programar, 6ª. Edição. Pearson Education. São Paulo. 2005.

ECLIPSE, Disponível em <https://www.eclipse.org/>. Acesso em 20 de mar. 2015.

FIELDING, R., "Relative Uniform Resource Locators", RFC 1808, UC Irvine, June 1995.

GAMMA, Erich et al. Padrões de projeto: soluções reutilizáveis de software orientado a objetos. Tradução Luiz A. Meirelles Salgado. Porto Alegre: Bookman, 2000.

GARLAN, David; SHAW, Mary. An introduction software architecture. School of Computer Science Carnegie Mellon University Pittisburgh, 1994.

GOLDBERG, A.; RUBIN, K. Succeeding with Objetcs; Decision Frameworks for Project Management. Reading: Addison-Wesley, 1995.

GOSCIOLA, Vicente. Roteiro para as novas mídias: do game à TV interativa. São Paulo, Editora Senac São Paulo, 2003.

IETF, Disponível em <https://www.ietf.org/>. Acesso em 20 de mar. 2015.

JBOSS, Disponível em <http://www.jboss.org/>. Acesso em 20 de mar. 2015.

LAUDON, Kenneth C. e LAUDON, Jane P. Management information systems: New Approaches to Organization & Tecnology.5ªed. New Jersey: Pretince Hall, 1998.

LUCKOW, Décio Heinzelmann; MELO, Alexandre Altair de. Programação Java para a Web: Aprenda a desenvolver uma aplicação financeira pessoal com as ferramentas mais modernas da plataforma Java. 1 ed. São Paulo: Novatec, 2010.

MACORATTI, J. C. Padrões de projeto: O modelo MVC. Disponível em <http://imasters.uol.com.br/artigo/2592/java/j2ee_x_net/> Acesso em 20 de mar. 2015.

McGARRY, K. O conceito dinámico da informação: uma análise introdutória. Brasilia: Briquet de Lemos/Livros, 1999. p.111-142.

MENDES, Antonio. Arquitetura de Software: desenvolvimento orientado para arquitetura. Editora Campus. Rio de Janeiro - RJ, 2002.

MORAES, Geraldo Leite. Aplicações Web. Processamento de Dados - UNIT. Aracajú, 2001.

OFICINA DA NET, “MVC - O padrão de arquitetura de software”. Disponível em: <http://www.oficinadanet.com.br/artigo/1687/mvc_o_padrao_de_arquitetura_de_software>. Acesso em: 20 de maio. 2015.

OPENJDK, Disponível em <http://openjdk.java.net/>. Acesso em 20 de abr. 2015.

OPENJPA, Disponível em <http://openjpa.apache.org/>. Acesso em 20 de abr. 2015.

ORACLE, Disponível em <http://www.oracle.com/br>. Acesso em 20 de abr. 2015

PARAMESWARAN, M., SUSARLA, A., WHINSTON, A., “P2P Networking: An Information-Sharing Alternative”. IEEE Computer, Julho de 2001.

LÉVY, Pierre. A Revolução contemporânea em matéria de comunicação. Revista Famecos, Porto Alegre, 1998. Semestral. Disponível em <http://revistaseletronicas.pucrs.br/ojs/index.php/revistafamecos/article/viewFile/3009/2287>. Acesso em 20 mar. 2015.

PRESSMAN, R.S. Engenharia de Software. São Paulo: Makron Books, 1995.

QUICOLI, Paulo R. MVC: Model-View-Controller. Clube Delphi, RJ, 2007.

RFC, Disponível em <http://www.ietf.org/rfc.html>. Acesso em 20 mar. 2015.

RICARTE, Ivan Luiz Marques. Programação Orientada a Objetos: Uma Abordagem com Java, 2001.

SANTOS, L. M. “Conceitos Básicos de Programação Orientada a Objetos”. Disponível em <http://pt.slideshare.net/leonardomelosantos/conceitos-bsicos-de-programacao-orientada-a-objetos> Acesso em 20 mar. 2015.

SEBRAE, Disponível em <http://www.sebrae.com.br/sites/PortalSebrae/>. Acesso em 20 de abr. 2015.

SIERRA, K.; BATES, B. “Use a cabeça. Java. 2. Ed. RJ: Alta Books, 2007.

SOMMERVILLE, Ian. Engenharia de Software.8.ed. São Paulo. Pearson Addison-Wesley, 2007.

STAIR, Ralph M. Princípios de Sistemas de Informação: Uma Abordagem Gerencial. Rio de Janeiro: LTC, 1998.

WARD, Mark. "Celebrating 40 years of the net", BBC News, 29 de outubro de 2009.

WEB DEVELOPERS, Disponível em <http://www.webdevelopersnotes.com/>. Acesso em 20 de abr. 2015.

WUTKA, M. Java: Técnicas Profissionais. Berkeley, 1997.