

Desenvolvimento de um sistema web para gerenciamento de estoque veicular

Pedro Henrique Rodrigues Soares¹

Prof. Ricardo de la Rocha Ladeira (orientador)¹

¹Instituto Federal Catarinense – Campus Blumenau

Blumenau, SC – Brasil

{p-henriquesoares@live.com, ricardo.ladeira@ifc.edu.br}

Abstract. *This paper describes study and development of a web application to manage the stock of a vehicle resale, replace simple controls for storing information and improve the security of business-critical data. The proposed solution is presented as a free web application available to resellers who wants to improve their inventory control. All requirements and needs have been met and a system is ready for use.*

Resumo. *Este trabalho descreve o estudo e o desenvolvimento de uma aplicação web destinada a gerenciar o estoque de uma revenda de veículos, substituir controles simples de armazenamento das informações e melhorar a segurança dos dados cruciais para o negócio. A solução proposta é apresentada como uma aplicação web gratuita disponível para revendedores de veículos, que desejam melhorar seu controle de estoque. Todos os requisitos e necessidades levantadas foram atendidos e, com isso, tem-se um sistema pronto para uso.*

1. Introdução

Diariamente, diversos veículos entram e saem de revendas em negociações de venda e troca, de forma que manter um controle ágil e fiel da relação do estoque pode se tornar um problema quando são utilizadas formas simples para armazenar as informações, propiciando margem para erros que surgem a todo momento, de maneira que a segurança dessas informações é basicamente inexistente.

Durante visitas realizadas a lojas de veículos, em Blumenau, foi possível verificar que revendedores utilizam de variadas formas para armazenar as informações dos veículos: papéis escritos à mão, controles de textos simples, planilhas e, no melhor

dos casos, algum *software* simples sistematização de algumas das informações cadastrais.

Possuir conhecimento completo a respeito do número de veículos e de suas características é algo crucial para o negócio. Sabendo disso, pode-se atribuir a responsabilidade de gerenciar todas essas informações a um sistema desenvolvido especificamente para isso, garantindo a segurança e a agilidade no armazenamento e no uso das informações que sustentam o negócio.

O fato de serem utilizadas formas precárias para armazenamento e gerenciamento de informações é justificado pelo custo das aplicações disponíveis no mercado. Diante do exposto, o trabalho em questão relata o desenvolvimento do VendaAuto, sistema voltado para o controle do estoque de veículos de uma revenda.

O trabalho está dividido em oito seções. A primeira seção introduz ao leitor o contexto do projeto, apresentando o cenário no qual se verificou a oportunidade de aplicar uma solução, os objetivos propostos para o trabalho, o escopo do projeto, a viabilidade de sua realização, ferramentas, recursos e tecnologias utilizadas e o método de trabalho. A segunda seção apresenta as soluções já existentes, em forma de trabalhos correlatos, ou seja, soluções semelhantes voltadas a atender o mesmo problema. A terceira seção apresenta o levantamento de requisitos funcionais e não funcionais utilizados no decorrer do projeto. A quarta seção apresenta o diagrama de caso de uso e suas descrições. A quinta seção apresenta a modelagem do banco de dados. A sexta seção apresenta os projetos de interface com o usuário. A sétima seção apresenta os resultados obtidos com a realização do trabalho. A oitava seção apresenta as considerações finais e as ideias para trabalhos futuros.

1.1. Tema/Problema

Conhecer todos os veículos presentes na revenda é algo fundamental para se apresentar de forma ágil durante a visita de um cliente. Exibir as opções certas, no momento exato, pode vir a definir o rumo da conversa com o cliente. Com isso, pode ocorrer um ganho para a revenda, em formato de negócio ou, do contrário, o cliente acabará optando por

buscar opções nas revendas concorrentes.

Quando a revenda possui poucos veículos em estoque é simples conhecer todas as informações a respeito destes, o problema surge com o crescimento do negócio e, conseqüentemente, a presença de um estoque maior de veículos. A simplicidade inicial em controlar o estoque desaparece, surgindo cenários de desatualização de informações, situações indesejadas para o negócio e que por ocorrerem, com maior frequência, podem deixar clientes insatisfeitos, bem como frustrar vendas.

Atualmente, a maioria dos vendedores de veículos carrega consigo um celular, necessário para possibilitar um contato próximo entre vendedor e clientes interessados nos veículos. Estes dispositivos podem ser utilizados para propiciar informações de todos os veículos disponíveis e as informações a estes vinculadas, provocando maior agilidade para a loja como um todo.

1.2. Objetivos Propostos/Solução dos Problemas

Este trabalho possui como objetivo geral desenvolver um sistema *web* para gerenciamento do estoque de veículos de uma revenda fictícia.

Os objetivos específicos são:

- a) Pesquisar o referencial teórico para o desenvolvimento do sistema.
- b) Levantar os requisitos funcionais e não funcionais do sistema.
- c) Desenvolver o projeto do banco de dados relacional.
- d) Implementar o sistema.
- e) Realizar testes.

1.3. Escopo

O projeto possui como foco a entrega de um sistema *web* voltado para o gerenciamento do estoque de veículos de uma revenda. O sistema deverá oferecer os módulos de clientes, veículos, vendas e configurações. No módulo de configurações existem os módulos de: usuários, grupos de usuários, países, estados, cidades, marcas, modelos, combustíveis, cores, portas, opcionais e adicionais.

Em todos os módulos oferecidos, com exceção ao de vendas, foram entregues as funcionalidades envolvendo: cadastro, consulta e alteração dos registros. No módulo de vendas estão presentes as funcionalidades de cadastro e consulta de vendas.

Não está contemplada no projeto a integração com serviços de terceiros, como, por exemplo, consultar o valor médio de veículos e a realização de emissão de notas fiscais. O módulo de vendas oferece apenas o registro das informações da pessoa e do veículo utilizados no negócio, nada mais.

A solução proposta não prevê a cópia segura de dados, tal funcionalidade deve ser configurada separadamente.

1.4. Viabilidade do Projeto

Devido à aplicação ser desenvolvida para *web*, o dispositivo cliente deverá contar com, no mínimo, 1GB de memória RAM para ser executado, tendo em vista que todo o processamento será realizado pelo servidor da aplicação, sendo requisitado do dispositivo apenas acesso à *internet* e um navegador.

1.5. Método de Trabalho

Para o desenvolvimento deste se optou por utilizar conceitos do *Unified Process* (UP) ou processo unificado, no entanto, este processo não foi seguido fielmente, como um todo.

O processo unificado prevê um conjunto de atividades necessárias para transformar requisitos de usuário em um *software*. Tal escolha ocorreu pela principal vantagem oferecida por este processo, ou seja, a de não ser necessário aguardar uma versão final do *software* para visualizar um resultado, tendo em vista que estes resultados são vistos, constantemente, devido ao modelo de desenvolvimento ser interativo e incremental. Cada interação realizada durante a etapa de desenvolvimento do projeto resulta em um incremento, e este significa uma nova funcionalidade no sistema (PRESSMAN, 2011).

De modo a compreender melhor a respeito do desenvolvimento deste trabalho,

inicialmente, se faz necessário esclarecer o funcionamento da *internet*.

A *internet* é uma rede largamente utilizada para realizar a comunicação entre todos os computadores e dispositivos com acesso a esta rede.

Os computadores conectados à *internet* interagem através de *requests* e *responses* (requisições e respostas) e, assim, cada requisição efetuada por um dispositivo na rede indica, além dos endereçamentos dos dispositivos do solicitante e do destinatário, também qual o conteúdo desejado.

A *web* é mantida por computadores que apresentam um processamento e disponibilidade superiores, e estes são chamados de *servers* (servidores). Dessa forma, os servidores são responsáveis por receberem as requisições efetuadas pelos computadores convencionais, chamados de *clients* (clientes), bem como em retornar na forma de respostas.

A comunicação entre cliente e servidor é realizada utilizando o protocolo *HyperText Transfer Protocol* (HTTP) ou Protocolo de Transferência de Hipertexto, através de um navegador *web*, responsável por enviar requisições, receber respostas e as traduzir, de modo que o usuário visualize no formato de documentos, de fotos, de vídeos e de imagens.

Conhecendo a respeito do funcionamento da comunicação entre computadores através da *web*, pode-se entender o conceito de *web app* (aplicação *web*), foco do trabalho em questão.

Uma aplicação *web* é como um *software*, porém ao invés de ser instalada, em um computador convencional, esta aplicação é configurada no servidor, de modo que seja possível acessá-la através da *internet*, em que todo o processamento da aplicação é realizado no servidor, enquanto a apresentação visual é realizada no navegador de *internet* utilizado pelo usuário.

O funcionamento da aplicação está descrito no decorrer desta seção, que além de exibir as ferramentas utilizadas neste trabalho, também aborda os recursos que compõem a aplicação, de modo que a mesma esteja disponível para acesso pelo usuário.

1.6. Ferramentas

1.6.1. Eclipse Neon 3

Entre as diversas ferramentas que se podem aplicar é apresentado o *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, ou ainda IDE, que se trata de um *software* gratuito e de código fonte aberto, que foi criado com a finalidade de fornecer um ambiente para o desenvolvimento de aplicações.

Entre as aplicações existentes se expressa que o Eclipse fornece suporte a várias linguagens para desenvolvimento de aplicações, tanto *desktop* quanto *web*, de forma que este foi utilizado no projeto para o desenvolvimento de arquivos nas linguagens Java, HTML, CSS e XML.

O *software* foi utilizado para o desenvolvimento de toda a aplicação desde a definição das classes principais do sistema e a sua publicação no servidor *web* (ECLIPSE, 2017).

1.6.2. MySQL Server 5.7

O MySQL é um Sistema de Gerenciamento de Banco de Dados ou SGBD, utiliza, em suas consultas, a Structured Query Language (SQL) ou Linguagem de Consulta Estruturada.

O MySQL foi utilizado para gerenciar o banco de dados, ou seja, foi aplicada para realizar operações no banco de dados, utilizando a linguagem de consulta estruturada ou SQL (ORACLE, 2017).

1.6.3. Workbench 6.3 CE

O Workbench é uma ferramenta visual gratuita para *design*, desenvolvimento e administração da base de dados MySQL, utilizada para definir, criar e realizar consultas, usando a linguagem SQL no banco de dados da aplicação (ORACLE, 2017).

A ferramenta foi utilizada para gerenciar o banco de dados de forma visual, e nesta foram criadas todas as tabelas que compõem o sistema, bem como as inserções de informações, de consultas e de atualizações necessárias para o resultado final do

sistema, sendo também desenvolvido o Modelo Entidade Relacionamento - MER, apresentado na seção 5.1 deste documento.

1.6.4. Astah Community 7.1

O Astah é uma ferramenta que possibilita a criação e modelagem dos diagramas UML - *Unified Modeling Language* ou Linguagem de Modelagem Unificada do projeto (ASTAH, 2017).

A ferramenta foi utilizada para o desenvolvimento do diagrama de Caso de Uso apresentado na seção 4.0 deste documento.

1.6.5. Notepad ++ 6.9.2

O Notepad é um editor gratuito, contendo diversas ferramentas que facilitam o trabalho com textos, tendo em vista que este editor oferece suporte e reconhecimento para várias linguagens de programação (HO, 2017).

Nesse sentido, este editor foi utilizado durante o projeto com enfoque na realização de trabalhos pequenos e rápidos, como alterações em classes por exemplo, o que permitiu que se alcançasse uma melhor produtividade.

1.6.6. Apache Tomcat 8.0

O Apache Tomcat é um *container* de *servlets*, ou seja, um conjunto de servidores menores, nos quais cada servidor é visto como uma aplicação gerenciada pelo Tomcat, o *container* é responsável por mapear Uniform Resource Locator (URL) ou Localizador Uniforme de Recursos para os *servlets* Java e retornar ao requisitante a resposta gerada pela aplicação (APACHE, 2017).

O Apache Tomcat foi utilizado para hospedar, localmente, a aplicação de modo que se possa executá-la, bem como realizar os testes necessários durante o desenvolvimento das funções.

1.7. Front-end

1.7.1. HTML 5

HyperText Markup Language – Linguagem de Marcação de Texto ou ainda HTML, é uma linguagem utilizada na construção de páginas *web* para publicação de conteúdo, na qual os documentos criados na linguagem são interpretados pelos navegadores e apresentados ao usuário, no formato de letras diferenciadas em cores, em imagens e em vídeos.

Em sua versão 5 foram incluídas novas *tags* para realização de marcações responsáveis por identificar o conteúdo presente na página, novos elementos para vídeos e imagens, elementos para desenhos, controles para formulários e suporte ao CSS3 (W3SCHOOLS, 2017).

1.7.2. CSS3

A ferramenta denominada de *Cascading Style Sheets* (CSS) ou Folhas de Estilo em Cascata é uma linguagem utilizada para definir a aparência de documentos, que adotam as linguagens de marcação como, por exemplo, o HTML, tendo em vista que o CSS define como serão exibidos os elementos existentes em uma página da *web*.

Em sua versão 3 foram incluídas novas propriedades, de forma a facilitar a criação de estilos para os componentes, realizar avanços na utilização das cores e textos e, ainda, possibilitar selecionar atributos HTML com os seletores entre outros (W3SCHOOLS, 2017).

1.7.3. jQuery 3.2.1

O jQuery é entendido como Biblioteca de JavaScript, tendo sido criada com a finalidade de ser mais leve e fácil de se utilizar, no sentido de escrever menos códigos e fazer mais na aplicação.

Foi utilizado no projeto para realizar requisições do tipo Ajax, responsável por transitar as requisições entre a página HTML e o Java (JQUERY, 2017).

1.7.4. UIKit Upload 3

O UIKit é um *framework* utilizado, também, para facilitar o desenvolvimento do *front-end* da aplicação, contém recursos pré-compilados assim como o Bootstrap.

Contendo várias funcionalidades, foi utilizada apenas uma delas chamada *upload* que, juntamente com o Javascript foi a responsável por cuidar da parte de envio de fotos dos veículos (UIKIT, 2017).

A Figura 1 demonstra um exemplo de uso do *framework*.

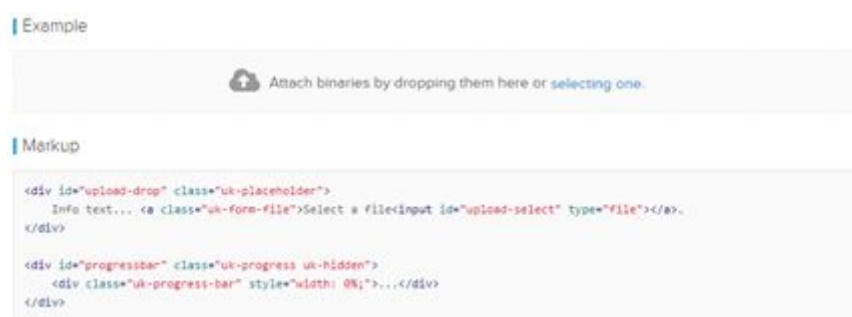


Figura 1. UIKit Upload 3. Fonte: UIKIT.

1.7.5. Font Awesome 4.7

A ferramenta nomeada de Font Awesome se mostra como um conjunto de arquivos pré-compilados, que possibilita a utilização de variadas fontes e ícones personalizados no projeto (FONTAWESOME, 2017).

Esta ferramenta foi utilizada para atender a todos os ícones, bem como algumas fontes do projeto.

1.7.6. JavaScript

Considerada como uma linguagem de programação, o JavaScript executa no lado do usuário, ou seja, é processada pelo navegador, quando o usuário abre a página contendo a chamada do *script*.

Com JavaScript se pode criar desde efeitos na página como, por exemplo, mostrar um *pop-up* em que se exibe uma mensagem ou conteúdo ou até mesmo coletar

informações de itens de formulários e enviar para o *back-end* efetuar o processamento dessa informação e retornar com um determinado resultado (JAVASCRIPT, 2017).

1.7.7. Bootstrap 3

O Bootstrap é um *framework*, cuja finalidade envolve facilitar o desenvolvimento do *front-end* da aplicação, sendo este baseado em jQuery por possuir vários componentes pré-compilados, responsáveis por implementar funcionalidades de forma fácil ao projeto, sendo a seguir exemplificado na figura que segue (BOOTSTRAP, 2017).

A Figura 2 demonstra um exemplo de uso do Bootstrap.

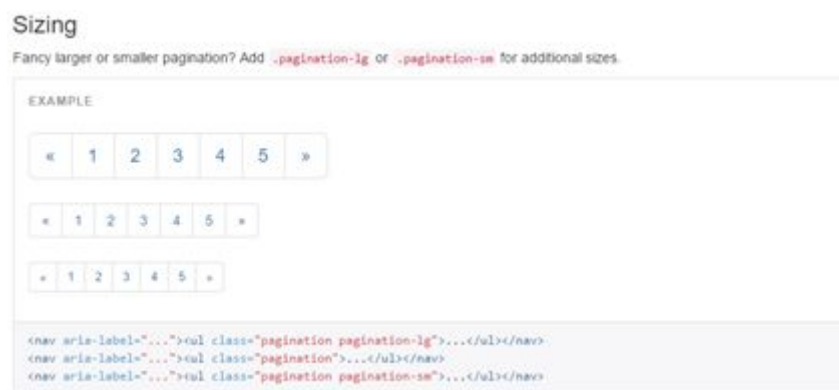


Figura 2. Bootstrap 3. Fonte: BOOTSTRAP.

1.8. Back-end

1.8.1. Java 8

O Java é uma linguagem de programação orientada para objetos, sendo esta pertencente a empresa Oracle. As aplicações desenvolvidas nessa linguagem não são compiladas em código nativo da plataforma, são compiladas para *bytecode* para que sejam executadas pela JVM – *Java Virtual Machine* ou Máquina Virtual do Java, após compilada uma única vez, a aplicação poderá ser executada em vários dispositivos que executam a JVM (CAELUM, 2017).

1.8.2. JRebel

A JRebel é uma ferramenta considerada como Plugin JVM, que permite ao desenvolvedor realizar alteração em classes Java, da mesma forma que visualizar o

resultado, instantaneamente, sem que seja necessário compilar todo o projeto novamente, pois ao utilizar o *plugin* somente a classe que sofreu alteração é compilada (ZEROTURNAROUND, 2017).

1.8.3. Spring Framework 4.3.0.RELEASE

O Spring é um projeto criado com a finalidade de facilitar o desenvolvimento de sistemas e a integração com outros *frameworks*, uma vez que utiliza um conjunto de ferramentas, que aumenta a produtividade ao fornecer uma estrutura completa de componentes, uma vez que estes gerenciam a aplicação como um todo. A principal característica desta ferramenta é a injeção de dependências no projeto, porém tal ferramenta possui outras características, conforme apresentadas a seguir (PIVOTAL, 2017).

- Desacoplamento de objetos: elimina o fato de instanciar uma classe na outra evitando, assim, o acoplamento entre classes.
- Facilita a manutenção: consequência do item a, eliminando o “acorrentamento” entre classes ao facilitar a manutenção do projeto.
- Injeção de dependências de forma declarativa: utilização de anotações para injetar dependências nas classes, tendo em vista utilizar arquivo XML externo para definição da dependência, evitando, assim, a necessidade de alteração nas classes e novas compilações devido às alterações.
- Independe do tamanho do sistema para ser utilizado.
- Alternativa ao EJB - *Enterprise JavaBeans*.

A Figura 3 demonstra a separação das camadas de atuação do Spring no projeto.

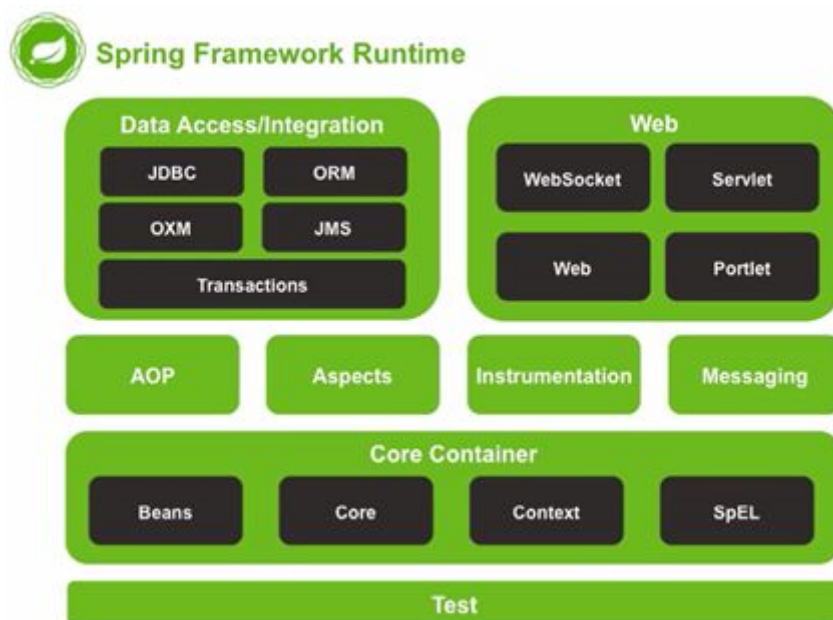


Figura 3. Separação das camadas de atuação do Spring no projeto. Fonte: ALGAWORKS.

1.8.3.1. Spring MVC 4.3.0.RELEASE

Outra ferramenta é o módulo do Spring Framework, que atua entre a *View* e o *Controller* da aplicação, para explicá-lo será necessário entender que a aplicação foi desenvolvida, utilizando o conceito *Model*, *View*, *Controller* ou Modelo, Visualização e Controlador ou ainda MVC, de acordo com o esquema que segue abaixo.

A Figura 4 demonstram as camadas MVC do projeto.

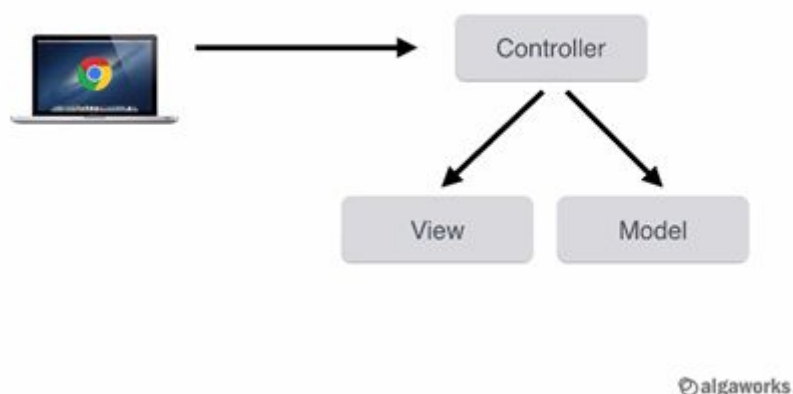


Figura 4. Camadas MVC do projeto. Fonte: ALGAWORKS.

A camada *Model* da aplicação define as regras de negócio no sistema, a camada *Controller* se encarrega por transitar e aplicar algumas regras do sistema e, por fim, a camada *View* é responsável por montar e apresentar toda a resposta para o usuário.

O Spring MVC é um módulo do *Spring Framework*, que atua entre a *View* e o *Controller* da aplicação, ele é o *Front Controller Spring MVC*, ou seja, atua recebendo a requisição criada pelo usuário e direcionando para a camada *Controller*, que possui o mapeamento adequado para atender àquela requisição (CAELUM, 2017).

O esquema a seguir na figura demonstra como ocorre este direcionamento de requisição do usuário, que se direciona para o *Controller*, de forma que esta requisição seja direcionada e se alcance a devida resposta para a solicitação.

A Figura 5 demonstram as camadas MVC do projeto junto ao Front Controller do Spring.

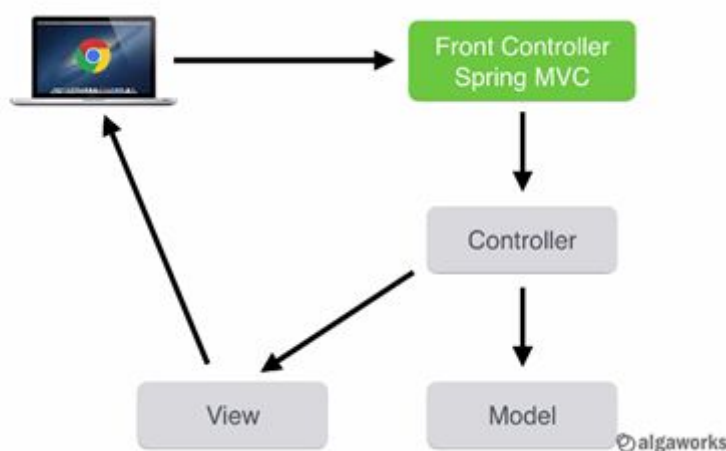


Figura 5. Camadas MVC do projeto com o Front Controller do Spring. Fonte: ALGAWORKS.

1.8.3.2. Spring Data JPA 1.10.2.RELEASE

Módulo do Spring é um *framework*, que atua na camada de persistência das informações, facilitando as operações realizadas no banco de dados da aplicação.

O Spring Data Java Persistence API ou API de persistência do Java ou simplesmente JPA, fornece métodos prontos e abstraídos para consultas, no banco de dados, eliminando a necessidade de se programar a consulta SQL no Java como, por

exemplo, o método *findPessoaByCodigo*, sem que seja necessária sua implementação, uma vez que esta trará um veículo pelo código.

A Figura 6 demonstra a atuação do Spring Data JPA na camada de persistência de dados da aplicação.

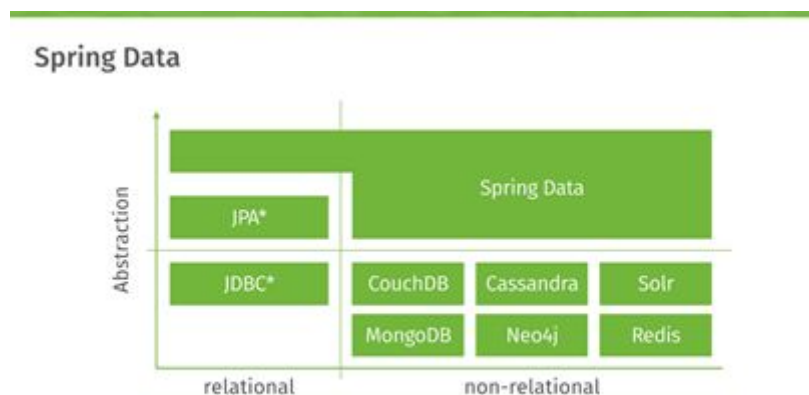


Figura 6. Camada do Spring Data JPA no Spring Framework. Fonte: PIVOTAL.

1.8.3.3. Spring Security 4.1.1.RELEASE

Módulo do Spring, sendo este um *framework* que atua na camada de segurança da aplicação, e este possui como finalidade facilitar a configuração da autenticação e permissão dos usuários ao sistema. Para melhor descrever sua funcionalidade se aborda do que se tratam os dois itens citados:

- Autenticação: Com poucas configurações na aplicação já se pode ter uma autenticação de um determinado usuário e sua senha.
- Permissão: Com as permissões se pode restringir o acesso de determinado usuário, mesmo que este esteja autenticado à determinada tela do sistema.

O Spring Security, além de manusear as autenticações e permissões ao sistema, também é responsável por responder à determinada *request* do usuário com o código 403, que é código de “*access denied*”. Pode criar um *controller* para mapear a negação de acesso e encaminhar uma *response* no formato de uma página personalizada ou até mesmo realizar alguma outra ação no sistema como, por exemplo, registrar em *log* (PIVOTAL, 2017).

1.8.3.4. Spring Boot 1.4.1.RELEASE

O Spring Boot é um módulo do Spring que atua facilitando a configuração e publicação da aplicação. Este módulo denominado de Spring Boot é responsável por envolver os demais módulos, sendo estes delineados em dependência do projeto, para que estes sejam assim gerenciados.

O módulo se encarregará de realizar todo o gerenciamento para que os módulos a este vinculados atuem no projeto, sendo relevante expor que nada impede, também, de se realizarem configurações em cada módulo, de modo a adequar-se às necessidades da aplicação (PIVOTAL, 2017).

A Figura 7 demonstra os módulos do Spring Boot que em conjunto atuam como um sistema integrado.

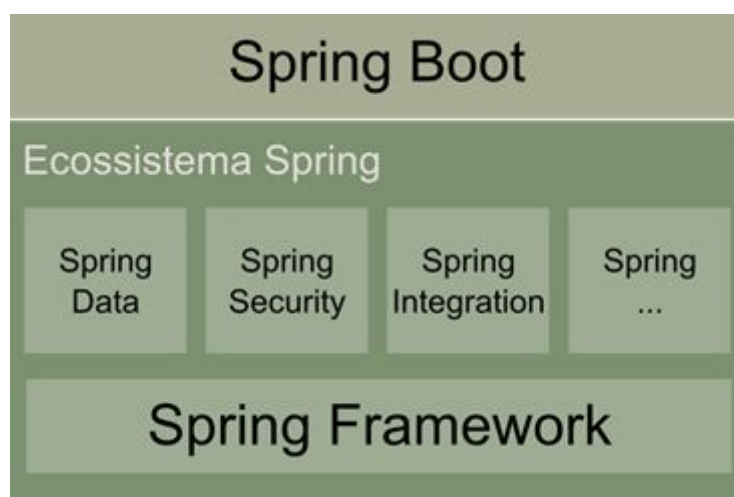


Figura 7. Camada do Spring Boot no Spring Framework. Fonte: PIVOTAL.

1.8.4. MySQL Connector 5.1.39

Utiliza-se o MySQL Connector 5.1.39 como uma forma de biblioteca, que oferece um conjunto de recursos e ferramentas para que o Java consiga realizar conexão com o banco de dados MySQL (ORACLE, 2017).

1.8.5. Hibernate Validator 5.2.4.Final

O Hibernate é uma biblioteca que possibilita o uso de anotações para validar dados nas classes do domínio da aplicação, ou seja, do *model*, no qual se definem as restrições nos

modelos, para que as informações não sejam persistidas ou distorcidas no banco de dados (HIBERNATE, 2017).

A Figura 8 demonstra um exemplo de uso do Hibernate Validator no projeto.

```
@Entity
@Table(name = "pessoa")
public class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "cod_pessoa")
    private Long codigo;

    @NotBlank(message = "Nome não pode estar vazio")
    @Size(max = 50, message = "O nome da pessoa não pode ultrapassar 50 caracteres!")
    @Column(name = "nome")
    private String nome;

    @NotBlank(message = "Data de nascimento não pode estar vazia")
    @Size(max = 10, message = "A data de nascimento da pessoa não pode ultrapassar 10 caracteres")
    @Column(name = "data_nascimento")
    @JsonIgnore
    private String dataNascimento;
```

Figura 8. Utilização do Hibernate Validator na camada model da aplicação. Fonte: Elaborada pelo autor.

1.8.6. Thymeleaf 3.0.1.RELEASE

O Thymeleaf é uma *Template Engine*, que fica situada entre a *view* do MVC e o navegador do usuário, a *template engine* é responsável por receber os dados retornados para a *view* e transformá-los em uma página HTML, de modo que o navegador consiga interpretar e apresentar ao usuário.

No projeto, a *template engine* escolhida foi o *Thymeleaf*, sendo este formado por um conjunto de *dialects* (dialetos), um dialeto representa uma funcionalidade do *Thymeleaf* na página HTML e cada dialeto é formado por um conjunto de processadores e estes são os reais responsáveis por implementar tudo o que um dialeto pode fazer na página (THYMELEAF, 2017).

A Figura 9 demonstra a declaração dos dialetos do Thymeleaf no projeto.


```
<html lang="pt" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:data="http://thymeleaf.org/extras/data"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout/tcc/LayoutPadrao}"
      xmlns:tcc="http://vendaauto.com">
```

Figura 9. Dialetos Thymeleaf declarados para uso no HTML. Fonte: Elaborada pelo autor.

1.8.6.1. Thymeleaf Standard Dialect ('th')

O Thymeleaf Standard Dialect ('th') é também chamado de *Standard Dialect* ou Dialetto Padrão do Thymeleaf e possui como objetivo possibilitar a realização de operações complexas, diretamente na página *web*.

Ao se utilizar o *Thymeleaf* no projeto, por padrão, já é possível trazer um objeto do Java para a página HTML, de modo que os atributos desse objeto possam ser utilizados na página.

A Figura 10 demonstra o objeto 'veículo' declarado em uma das páginas HTML.

```
<div class="container-fluid">
  <form method="POST" th:object="${veiculo}">
```

Figura 10. Objeto 'veículo' disponibilizado no HTML. Fonte: Elaborada pelo autor.

Pode-se também realizar operações simples como um '*if*' e '*else*' através dos processadores '*th:if*' e '*th:unless*' respectivamente.

A Figura 11 demonstra o uso do IF-ELSE possibilitado pelo Thymeleaf em uma das páginas HTML.

```
<h1 th:if="${veiculo.novo}">Cadastro de Veículo</h1>
<h1 th:unless="${veiculo.novo}" th:text="/Edição de ${veiculo.placa}"/>Edição de Veículo</h1>
```

Figura 11. IF-ELSE através do Thymeleaf na página HTML. Fonte: Elaborada pelo autor.

Através do uso do *Thymeleaf* se pode fazer com que determinado componente de um formulário HTML seja alimentado com valores do objeto, anteriormente chamado na página, por exemplo. No trecho de código abaixo, o *combo-box* '*id="marca"*' recebe valores do atributo '*th:field="**{marca}"*' presente no objeto Java

`th:object="${veiculo}"` através de uma lista de marcas em `th:each="marca:${marcas}"`, cada registro presente no combo será alimentado com o valor `th:value="${marca.codigo}"` e texto `th:text="${marca.descricao}"`, ao selecionar uma das opções ali presentes, o objeto veículo será alimentado e ao submeter o formulário o objeto ‘veículo’ no Java já estará com a opção escolhida pelo usuário.

A Figura 12 demonstra a variável ‘marca’ do objeto ‘veículo’ recebendo valor de um componente denominado *combobox* em uma das páginas HTML.

```
<div class="form-group col-sm-6 sa-required"
    tcc:classforerror="marca">
    <label for="marca" class="control-label">Marca</label> <select
        id="marca" class="form-control" th:field="*{marca}">
        <option value="">Selecione a marca</option>
        <option th:each="marca:${marcas}" th:value="${marca.codigo}"
            th:text="${marca.descricao}"></option>
    </select>
</div>
```

Figura 12. Variável ‘marca’ do objeto ‘veículo’ recebendo valor de um combo HTML. Fonte: Elaborada pelo autor.

1.8.6.2. Thymeleaf Layout Dialect (‘layout’)

Dialeto do Thymeleaf que possibilita a organização das páginas HTML do projeto, com ele é possível eliminar duplicidade de código através do reaproveitamento, tal reaproveitamento pode ser feito por meio de duas formas:

- *Include-style layouts* (`th:insert`): Esta forma de reaproveitamento de código já é nativa no *Standard Dialect do Thymeleaf*, seu uso ocorre com a criação de um arquivo HTML separado para um determinado trecho de código, que é utilizado com muita frequência, feito isso basta realizar o *insert*, conforme exemplo na Figura 13 abaixo, em todas as páginas, que possuíam o código duplicado:

```
<body>
    <div th:insert="footer :: copy">...</div>
</body>
```

Figura 13. Exemplo de include. Fonte: Elaborada pelo autor.

- *Hierarchical-style layouts* (`th:replace`): Esta é a forma utilizada no projeto, seu uso ocorre com a definição de uma página de *layout padrão*. Essa página

possuirá todo o conteúdo comum entre todas as páginas do projeto como, por exemplo, barra de menu, de importações de *scripts* e de arquivos css necessários para o funcionamento de todas as páginas, somente quando uma página específica necessitar de um conteúdo diferenciado é que se realiza a inserção desse conteúdo (THYMELEAF, 2017).

A Figura 14 demonstra um exemplo de uso do *replace*.

```
<div th:replace="${#authentication.principal.isAdmin()} ? ~(fragments/footer :: footer-admin) : ~(fragments/footer :: footer-admin)"
&copy; 2016 The Static Templates
</div>
```

Figura 14. Exemplo de replace. Fonte: Elaborada pelo autor.

1.8.6.3. Thymeleaf Data Dialect ('data')

O Thymeleaf Data Dialect ('data') é o dialeto utilizado para facilitar o trânsito de informações, que serão utilizadas pelo Ajax em uma requisição futura, sendo a Figura 15 a seguir ilustração de seu uso.

```
<div class="form-group col-sm-6 sa-required"
    tcc:classforerror="modelo">
    <label for="modelo" class="control-label">Modelo</label> <select
        id="modelo" class="form-control" th:field="*{modelo}"
        data:url="@{/modelos}">
        <option value="">Selecione o modelo</option>
    </select>
    <div class="va-field-action__icon js-img-loading"
        style="display: none">
        
    </div>
</div>
```

Figura 15. Exemplo de uso do dialect 'data'. Fonte: Elaborada pelo autor.

1.8.7. Apache Log4j 2.6

Framework que possibilita a geração de *logs* para depuração da aplicação e a sua utilização evita que o código-fonte do projeto fique poluído com vários '*System.out.println()*' por exemplo.

O Log4j é composto por três aspectos, sendo estes:

- *Logger*: encarregado por receber as requisições de *log*, em que se pode criar um *logger* para cada classe da aplicação se desejado, porém o *framework* já oferece um a cada padrão.

- *Appender*: recebe os *logs* do *Logger*, sendo encarregado por direcionar os *logs* recebidos para arquivos ou banco de dados, por exemplo.
- *Layout*: trata-se do formato no qual o *log* será armazenado, em que se pode organizar um *log* em formato de texto simples como HTML, também se definem: data, hora e linha de código-fonte, em que o *log* foi gerado.

Com o *framework* é possível, também, configurar os níveis de *logs* em *DEBUG* < *INFO* < *WARN* < *ERROR* < *FATAL* (APACHE, 2017).

1.8.8. Hibernate 5.1.0.Final

O Hibernate 5.1.0. Final é um *framework*, cuja função é realizar o mapeamento objeto-relacional na linguagem Java, tendo em vista que sua utilização no projeto facilitou o mapeamento das classes Java com as tabelas no banco de dados, da mesma forma que facilitou todas as consultas com o *Hibernate Criteria*, nas quais se gerenciou e evitou, que todo o código necessário para suprir essa necessidade, manualmente, fosse criado para mapear cada entidade da aplicação, aumentando assim a produtividade (EBERSOLE, 2017).

A Figura 16 demonstra a camada de atuação do Hibernate.

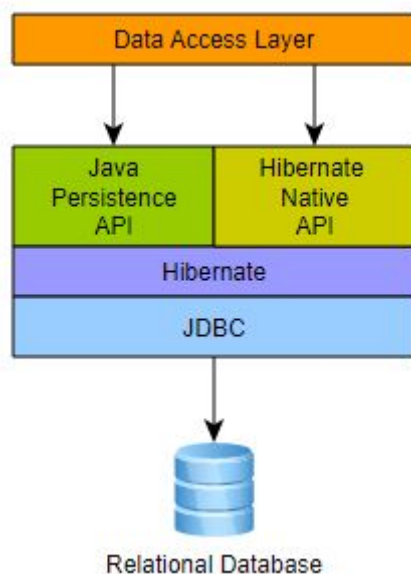


Figura 16. Hibernate 5.1.0.Final. Fonte: EBERSOLE.

1.8.9. Jackson JSON 2.7.5

O JSON é uma biblioteca utilizada no projeto, cuja funcionalidade implica facilitar a conversão de objetos para JSON - *JavaScript Object Notation*, facilitando a transferência de informações de objetos entre a *view* e o *controller* (LEARY, 2017).

1.8.10. Thumbnailator 0.4.8

O Thumbnailator é uma biblioteca utilizada para gerar os *thumbnails* das imagens de veículo, ou seja, imagens menores que as originalmente cadastradas.

O uso desta ferramenta foi necessário com foco em minimizar o volume de tráfego de imagens na tela de pesquisa. De forma geral, quando o usuário realiza a pesquisa, o sistema deve trazer a imagem vinculada ao veículo, se a imagem for a original e a página apresentar dez registros, por exemplo, o tráfego de dados gerado na rede será muito grande.

Com intuito de resolver esse aspecto acima registrado, a pesquisa de veículos traz a imagem com tamanho reduzido, somente apresentando a imagem original quando se visualiza o veículo por completo na tela de edição (GITHUB, 2017). Logo a seguir, na Figura 17, se pode visualizar como ocorre este processo.



Figura 17. Thumbnailator 0.4.8. Fonte: GITHUB.

1.9. Banco de dados

1.9.1 Flyway

O Flyway é um *Plugin* que automatiza a criação de tabelas no banco de dados e gerencia as migrações do projeto no banco de dados utilizado.

Com o decorrer do desenvolvimento do sistema foram necessárias alterações nas tabelas dos bancos, de modo a evitar que todo esse gerenciamento fosse feito, manualmente, se optou por utilizar o *plugin* Flyway para automatizar esse processo.

2. Trabalhos Correlatos Existentes

Existem diversas aplicações *web* similares a deste trabalho, como: RevendaMais, AutoWeb e Altimus. As três soluções correlatas possuem um número maior de funcionalidades, mas são pagas e impõem alguns limites, de maneira que a solução deste trabalho não possui como, por exemplo, número de usuários e veículos. Também não foram encontradas informações, nos três trabalhos correlatos, sobre restringir acesso de usuários por níveis de permissões, sendo característica essa que a solução proposta neste trabalho oferece.

A solução RevendaMais é uma aplicação *web* voltada para gestão de lojas de veículos.

De acordo com REVENDAMAIS (2017), sua versão mais completa oferece as seguintes características:

- 10 usuários;
- 4 CNPJ;
- 4GB de armazenamento.
- 150 veículos;
- Aplicativo *mobile*;
- Módulo de estoque;
- Emissão de notas fiscais;
- Integrador com sites para anunciar os veículos;
- Módulo financeiro;
- Módulo comercial;
- Módulo para refinanciamentos;
- Relatórios.

A Figura 18 demonstra a solução RevendaMais.



Figura 18. RevendaMais. Fonte: REVENDAMAIS.

AutoWeb é uma aplicação *web* voltada para gestão de lojas de veículos.

De acordo com AUTOWEB (2017), sua versão mais completa oferece as seguintes características:

- Sem limite de usuários;
- Emissão de notas fiscais;
- Integrador com sites para anunciar os veículos;
- Módulo financeiro;
- Módulo comercial;
- Módulo despachante;
- Módulo *Customer Relationship Management* (CRM), voltado para gestão de relacionamento com o cliente;
- Envio de *e-mails marketing*;
- Envio de SMS.

A Figura 19 demonstra a solução AutoWeb.



Figura 19. AutoWeb. Fonte: AUTOWEB.

A solução Altimus é uma aplicação *web* voltada para gestão de lojas de veículos.

De acordo com ALTIMUS (2017), sua versão mais completa oferece as seguintes características:

- Sem limite de usuários;
- 2 CNPJ;
- 100 veículos;
- Emissão de notas fiscais;
- Integrador com sites para anunciar os veículos;
- Módulo financeiro;
- Módulo comercial.

A Figura 20 demonstra a solução Altimus.



Figura 20. Altimus. Fonte: ALTIMUS.

A seguir, na Tabela 1, é apresentada uma comparação entre a solução *web* gratuita, que foi proposta pelo trabalho e as soluções pagas disponíveis no mercado.

Tabela 1. Comparação entre as funcionalidades oferecidas pelas soluções correlatas.

| | RevendaMais | AutoWeb | Altimus | VendaAuto |
|--------------------------|-------------|-----------|-----------|-----------|
| Qtde. Veículos | 150 | - | 100 | Ilimitado |
| Qtde. Usuários | 10 | Ilimitado | Ilimitado | Ilimitado |
| Permissões | - | - | - | Sim |
| Módulo Clientes | Sim | Sim | Sim | Sim |
| Armazenamento | 4GB | - | - | Ilimitado |
| CNPJ | 4 | - | 2 | - |
| Responsividade | Sim | Sim | Sim | Sim |
| Aplicativo | Sim | - | - | - |
| Módulo Estoque | Sim | Sim | Sim | Sim |
| Módulo Financeiro | Sim | Sim | Sim | - |

| | | | | |
|--------------------------------|---------|-----|---------|----------|
| Módulo Comercial | Sim | Sim | Sim | - |
| Módulo Refinanciamentos | Sim | - | - | - |
| Módulo Despachante | - | Sim | - | - |
| NF-e | Sim | Sim | Sim | - |
| Relatórios | Sim | Sim | Sim | - |
| Integrador web | Sim | Sim | Sim | - |
| CRM | - | Sim | - | - |
| Envio de e-mail | Sim | Sim | - | - |
| Envio de SMS | - | Sim | - | - |
| Custo | 399/mês | - | 399/mês | Gratuito |

Fonte: Elaborada pelo autor.

3. Requisitos

Esta seção apresenta os requisitos funcionais e não funcionais utilizados para o desenvolvimento do projeto.

3.1. Requisitos Funcionais

Esta seção apresenta os requisitos funcionais utilizados para o desenvolvimento do projeto.

RF001 – O sistema deve manter o cadastro de veículos.

RF002 – O sistema deve manter o cadastro de pessoas.

RF003 – O sistema deve manter o cadastro de vendas.

RF004 – O sistema deve manter o cadastro de usuários.

RF005 – O sistema deve manter o cadastro de grupos.

RF006 – O sistema deve manter o cadastro de países.

RF007 – O sistema deve manter o cadastro de estados.

RF008 – O sistema deve manter o cadastro de cidades.

RF009 – O sistema deve manter o cadastro de marcas.

RF010 – O sistema deve manter o cadastro de modelos.

RF011 – O sistema deve manter o cadastro de combustíveis.

RF012 – O sistema deve manter o cadastro de cores.

RF013 – O sistema deve manter o cadastro de número de portas.

RF014 – O sistema deve manter o cadastro de opcionais.

RF015 – O sistema deve manter o cadastro de adicionais.

RF016 – O sistema deverá possibilitar que se defina um usuário como Ativo para permitir o acesso ou Inativo para barrar o acesso ao sistema.

RF017 – O sistema deverá permitir acesso somente aos usuários autenticados.

RF018 – O sistema deverá possibilitar a personalização de níveis de acesso de usuários com a utilização de grupos e permissões de acesso às telas.

RF019 – O sistema deverá exibir, em números, as movimentações da revenda na tela inicial.

RF020 - O sistema deverá ser responsivo de modo a funcionar adequadamente em vários dispositivos.

3.2. Requisitos Não Funcionais

Esta seção apresenta os requisitos não funcionais utilizados para o desenvolvimento do projeto.

RNF001 – A aplicação deverá ser executada tanto em computadores como em dispositivos móveis, sendo necessário apenas um navegador e acesso à *internet*.

RNF002 – O desenvolvimento da aplicação deve ser realizado utilizando apenas ferramentas gratuitas.

RNF003 – A aplicação deve possibilitar acesso fora da loja física.

RNF004 – O tempo de resposta não deverá superar 10 (dez) segundos.

RNF005 – O SGBD deve ser o MySQL.

RNF006 – A linguagem deverá ser Java.

RNF007 – A sessão do usuário deverá expirar após 30 minutos de inatividade no sistema.

4. Diagramas UML

Esta seção apresenta o diagrama de casos de uso e a descrição de cada caso.

O diagrama de casos de uso documenta o que o sistema faz, sob o ponto de vista do usuário, uma vez que deve descrever todas as funcionalidades do sistema e a interação dessas funcionalidades com os usuários (FAKHROUTDINOV, 2017).

A seguir, na Tabela 2, são apresentadas as permissões de acesso às telas, o usuário após efetuar a autenticação terá acesso às seguintes funcionalidades.

Tabela 2. Funcionalidades definidas no diagrama de caso de uso.

| | |
|--------------------|--------------------|
| Cadastrar Pessoas | Cadastrar Veículos |
| Pesquisar Pessoas | Pesquisar Veículos |
| Excluir Pessoas | Excluir Veículos |
| Cadastrar Vendas | Pesquisar Vendas |
| Cadastrar Usuários | Cadastrar Grupos |
| Pesquisar Usuários | Pesquisar Grupos |
| Excluir Usuários | Excluir Grupos |
| Listar Permissões | Cadastrar Estados |
| Cadastrar Países | Pesquisar Estados |

| | |
|------------------------|---------------------|
| Pesquisar Países | Excluir Estados |
| Excluir Países | Cadastro Cidades |
| Cadastrar Marcas | Pesquisar Cidades |
| Pesquisar Marcas | Excluir Cidades |
| Excluir Marcas | Cadastrar Modelos |
| Cadastrar Combustíveis | Pesquisar Modelos |
| Pesquisar Combustíveis | Excluir Modelos |
| Excluir Combustíveis | Cadastrar Cores |
| Cadastrar Portas | Pesquisar Cores |
| Pesquisar Portas | Excluir Cores |
| Excluir Portas | Cadastrar Opcionais |
| Cadastrar Adicionais | Pesquisar Opcionais |
| Pesquisar Adicionais | Excluir Opcionais |
| Excluir Adicionais | |

Fonte: Elaborada pelo autor.

A descrição dos casos de usos e o diagrama se encontram no Apêndice A e Apêndice B deste trabalho.

5. Modelagem de dados

Esta seção apresenta o modelo do banco de dados relacional e seu dicionário de dados.

5.1. Modelo Entidade Relacionamento

O Modelo Entidade Relacionamento (MER) ou somente Entidade Relacionamento (ER) é utilizado para representar o banco de dados. Neste modelo são definidas entidades, relacionamentos, atributos, chaves primárias e chaves estrangeiras responsáveis por

manter a integridade das informações presentes no banco de dados (SMARTDRAW, 2017).

A Figura 21 demonstra o MER da aplicação, apresentando as tabelas e os relacionamentos do banco de dados. As informações das tabelas são: colunas, chaves e índices.

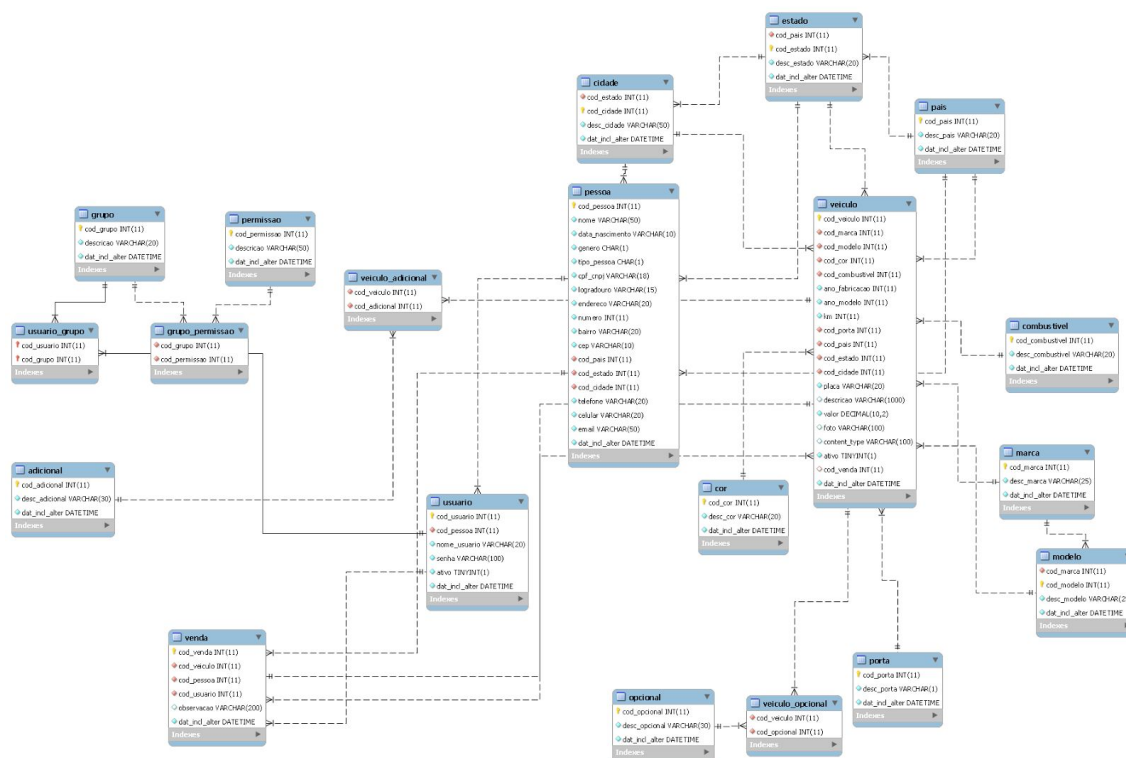


Figura 21. MER. Fonte: Elaborada pelo autor.

5.2. Dicionário de dados

Esta seção apresenta o dicionário de dados do banco de dados da aplicação, utilizado para descrever cada atributo, valor suportado, tamanho limite, se o campo oferece autoincremento, restrições de chave e nulidade.

| Tabela: marca | | | | | | | |
|----------------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_marca | INT | | X | | | X | Armazena o código da marca. |
| desc_marca | VARCHAR(25) | | | | | | Armazena a descrição da marca. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da marca. |

| Tabela: modelo | | | | | | | |
|-----------------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_marca | INT | | | X | marca | | Armazena o código da marca. |
| cod_modelo | INT | | X | | | X | Armazena o código do modelo. |
| desc_marca | VARCHAR(25) | | | | | | Armazena a descrição do modelo. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do modelo. |

| Tabela: cor | | | | | | | |
|--------------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_cor | INT | | X | | | X | Armazena o código da cor. |
| desc_cor | VARCHAR(20) | | | | | | Armazena a descrição da cor. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da cor. |

| Tabela: combustivel | | | | | | | |
|----------------------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_combustivel | INT | | X | | | X | Armazena o código do combustível. |
| desc_combustivel | VARCHAR(20) | | | | | | Armazena a descrição do combustível. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do combustível. |

| Tabela: porta | | | | | | | |
|----------------------|------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_porta | INT | | X | | | X | Armazena o código da porta. |
| desc_porta | VARCHAR(1) | | | | | | Armazena a descrição da porta. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da porta. |

| Tabela: opcional | | | | | | | |
|-------------------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_opcional | INT | | X | | | X | Armazena o código do opcional. |
| desc_opcional | VARCHAR(30) | | | | | | Armazena a descrição do opcional. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do opcional. |

| Tabela: adicional | | | | | | | |
|--------------------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_adicional | INT | | X | | | X | Armazena o código do adicional. |
| desc_adicional | VARCHAR(30) | | | | | | Armazena a descrição do adicional. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do adicional. |

| Tabela: pais | | | | | | | |
|---------------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_pais | INT | | X | | | X | Armazena o código do país. |
| desc_pais | VARCHAR(20) | | | | | | Armazena a descrição do país. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do país. |

| Tabela: estado | | | | | | | |
|----------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_pais | INT | | | X | pais | | Armazena o código do país. |
| cod_estado | INT | | X | | | X | Armazena o código do estado. |
| desc_estado | VARCHAR(20) | | | | | | Armazena a descrição do estado. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do estado. |

| Tabela: cidade | | | | | | | |
|----------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_estado | INT | | | X | estado | | Armazena o código do estado. |
| cod_cidade | INT | | X | | | X | Armazena o código da cidade. |
| desc_cidade | VARCHAR(50) | | | | | | Armazena a descrição da cidade. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da cidade. |

| Tabela: pessoa | | | | | | | |
|-----------------|-------------|-------|----|----|------------|----------------|---|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_pessoa | INT | | X | | | X | Armazena o código da pessoa. |
| nome | VARCHAR(50) | | | | | | Armazena o nome da pessoa. |
| data_nascimento | VARCHAR(10) | | | | | | Armazena a data de nascimento da pessoa. |
| genero | CHAR | | | | | | Armazena o gênero da pessoa. |
| tipo_pessoa | CHAR | | | | | | Armazena o tipo de pessoa. |
| cpf_cnpj | VARCHAR(18) | | | | | | Armazena o cpf ou cnpj da pessoa. |
| logradouro | VARCHAR(15) | | | | | | Armazena o logradouro onde a pessoa mora. |
| endereco | VARCHAR(20) | | | | | | Armazena o endereço onde a pessoa mora. |
| numero | INT | | | | | | Armazena número onde a pessoa mora. |
| bairro | VARCHAR(20) | | | | | | Armazena bairro onde a pessoa mora. |
| cep | VARCHAR(10) | | | | | | Armazena cep onde a pessoa mora. |
| cod_pais | INT | | | X | pais | | Armazena o código do país. |
| cod_estado | INT | | | X | estado | | Armazena o código do estado. |
| cod_cidade | INT | | | X | cidade | | Armazena o código da cidade. |
| telefone | VARCHAR(20) | | | | | | Armazena o telefone da pessoa. |
| celular | VARCHAR(20) | | | | | | Armazena o celular da pessoa. |
| email | VARCHAR(50) | | | | | | Armazena o e-mail da pessoa. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da pessoa. |

| Tabela: grupo | | | | | | | |
|----------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_grupo | INT | | X | | | X | Armazena o código do grupo. |
| desc_grupo | VARCHAR(20) | | | | | | Armazena a descrição do grupo. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do grupo. |

| Tabela: <u>permissao</u> | | | | | | | |
|---------------------------------|-------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_permissao | INT | | X | | | X | Armazena o código da permissão. |
| desc_permissao | VARCHAR(50) | | | | | | Armazena a descrição da permissão. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da permissão. |

| Tabela: <u>grupo_permissao</u> | | | | | | | |
|---------------------------------------|------|-------|----|----|------------|----------------|---------------------------------|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_grupo | INT | | | X | grupo | | Armazena o código do grupo. |
| cod_permissao | INT | | | X | permissao | | Armazena o código da permissão. |

| Tabela: <u>usuario</u> | | | | | | | |
|-------------------------------|--------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_usuario | INT | | X | | | X | Armazena o código do usuário. |
| cod_pessoa | INT | | | X | | | Armazena o código da pessoa. |
| nome_usuario | VARCHAR(20) | | | | | | Armazena o nome do usuário. |
| senha | VARCHAR(100) | | | | | | Armazena a senha do usuário. |
| ativo | BOOLEAN | | | | | | Armazena se o usuário está ativo. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do usuário. |

| Tabela: <u>usuario_grupo</u> | | | | | | | |
|-------------------------------------|------|-------|----|----|------------|----------------|-------------------------------|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_usuario | INT | | | X | usuario | | Armazena o código do usuário. |
| cod_grupo | INT | | | X | grupo | | Armazena o código do grupo. |

| Tabela: <u>veiculo</u> | | | | | | | |
|-------------------------------|---------------|-------|----|----|-------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_veiculo | INT | | X | | | X | Armazena o código do veículo. |
| cod_marca | INT | | | X | marca | | Armazena o código da marca. |
| cod_modelo | INT | | | X | modelo | | Armazena o código do modelo. |
| cod_cor | INT | | | X | cor | | Armazena o código da cor. |
| cod_combustivel | INT | | | X | combustivel | | Armazena o código do combustível. |
| ano_fabricacao | INT | | | | | | Armazena o ano de fabricação do veículo. |
| ano_modelo | INT | | | | | | Armazena o ano de modelo do veículo. |
| km | INT | | | | | | Armazena a quilometragem do veículo. |
| cod_porta | INT | | | X | porta | | Armazena o código da porta. |
| cod_pais | INT | | | X | pais | | Armazena o código do país. |
| cod_estado | INT | | | X | estado | | Armazena o código do estado. |
| cod_cidade | INT | | | X | cidade | | Armazena o código da cidade. |
| placa | VARCHAR(20) | | | | | | Armazena a placa do veículo. |
| descricao | VARCHAR(200) | X | | | | | Armazena a descrição do veículo. |
| valor | DECIMAL(10,2) | | | | | | Armazena o valor do veículo. |
| foto | VARCHAR(100) | | | | | | Armazena o nome da foto do veículo. |
| content_type | VARCHAR(20) | X | | | | | Armazena o tipo da foto do veículo. |
| ativo | BOOLEAN | | | | | | Armazena se o veículo está ativo. |
| cod_venda | INT | X | | X | venda | | Armazena o código da venda. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração do veículo. |

| Tabela: veiculo_opcional | | | | | | | |
|---------------------------------|------|-------|----|----|------------|----------------|--------------------------------|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_veiculo | INT | | | X | veiculo | | Armazena o código do veículo. |
| cod_opcional | INT | | | X | opcional | | Armazena o código do opcional. |

| Tabela: veiculo_adicional | | | | | | | |
|----------------------------------|------|-------|----|----|------------|----------------|---------------------------------|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_veiculo | INT | | | X | veiculo | | Armazena o código do veículo. |
| cod_adicional | INT | | | X | adicional | | Armazena o código do adicional. |

| Tabela: venda | | | | | | | |
|----------------------|--------------|-------|----|----|------------|----------------|--|
| Coluna | Tipo | Null? | PK | FK | References | Auto increment | Descrição |
| cod_venda | INT | | X | | | X | Armazena o código da venda. |
| cod_veiculo | INT | | | X | veiculo | | Armazena o código do veículo. |
| cod_pessoa | INT | | | X | pessoa | | Armazena o código da pessoa. |
| cod_usuario | INT | | | X | usuario | | Armazena o código do usuário. |
| observação | VARCHAR(200) | X | | | | | Armazena a observação da venda. |
| dat_incl_alter | DATETIME | | | | | | Armazena a data de inclusão ou alteração da venda. |

Figura 22. Dicionário de dados. Fonte: Elaborada pelo autor.

6. Projeto da Interface

Uma interface com o usuário é o meio termo entre a aplicação e o usuário final, ou seja, as telas do sistema. Procurou-se adequar todas as telas, de modo que os usuários não passem dificuldades em acessar qualquer opção, sendo também considerado que as telas deveriam ser, além de funcionais, de fácil utilização.

Nesse sentido, foi criado um padrão de *layout* limpo e bem definido, permitindo que o usuário possa acessar as funções da aplicação de forma rápida e fácil.

Na maioria das páginas foi utilizado o componente chamado *page-header* (topo de página), tendo em vista que seu uso fornece uma informação sobre a tela e um botão indicando uma próxima etapa, segundo um fluxo de uso.

A Figura 23 apresenta o *page-header* utilizado nas páginas.

Cadastro de Pessoa


 Pesquisa de Pessoa

Figura 23. Page-header presente na maioria das páginas. Fonte: Elaborada pelo autor.

A Figura 24 apresenta a tela de acesso ao sistema, na qual o usuário deverá

inserir seus dados de acesso para prosseguir para as funcionalidades do sistema, de acordo com as permissões do usuário.

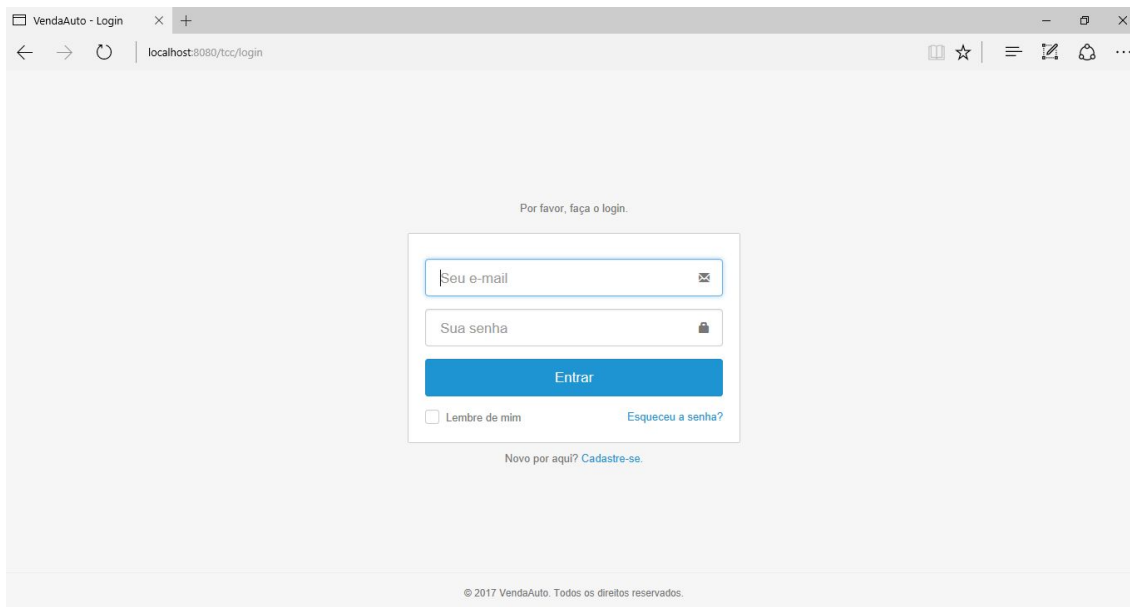


Figura 24. Tela de acesso ao sistema. Fonte: Elaborada pelo autor.

A Figura 25 apresenta a tela inicial do sistema, na qual o usuário poderá navegar nos menus de: Pessoas, Veículos, Vendas e acessar as configurações do sistema. A tela inicial do sistema se denomina *dashboard* ou painel de indicadores, atendendo ao requisito de exibir informações principais do sistema, número de veículos cadastrados, disponíveis em estoque, pessoas cadastradas e vendas cadastradas.

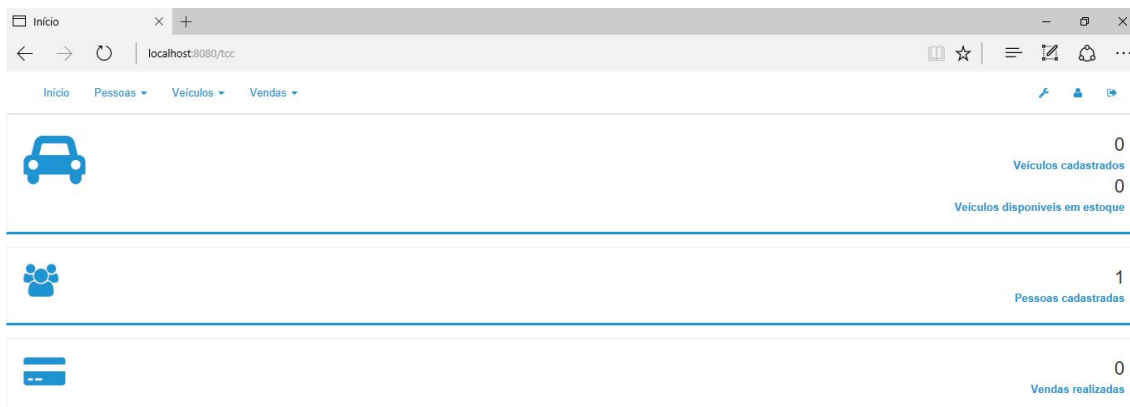


Figura 25. Tela inicial do sistema. Fonte: Elaborada pelo autor.

A Figura 26 apresenta a tela de cadastro de pessoa, na qual é possível realizar o cadastro de uma nova pessoa no sistema, sendo obrigatório para a criação de uma venda, na tela de cadastro de venda será necessário realizar o vínculo entre uma pessoa e um veículo.

É necessário, também, realizar o cadastro de pessoa para novos usuários da aplicação, futuramente, na tela de cadastro de usuário essa pessoa deve ser vinculada ao novo usuário do sistema.

A imagem mostra a interface de usuário para o cadastro de uma pessoa. No topo, há uma barra de navegação com links para 'Início', 'Pessoas', 'Veículos' e 'Vendas'. O título da página é 'Cadastro de Pessoa'. À direita, há um botão 'Pesquisa de Pessoa'. O formulário contém os seguintes campos:

- Tipo pessoa***: Radio buttons para 'Física' e 'Jurídica'.
- CPF/CNPJ***: Campo de texto para o número de identificação.
- Nome***: Campo de texto para o nome.
- Data Nascimento***: Campo de texto para a data de nascimento.
- Gênero***: Radio buttons para 'Masculino' e 'Feminino'.
- Logradouro***: Campo de texto para o endereço.
- Endereço***: Campo de texto para o endereço.
- Número***: Campo de texto para o número do endereço.
- Bairro***: Campo de texto para o bairro.
- CEP***: Campo de texto para o CEP.
- Pais***: Dropdown menu com a opção 'Selecione o país'.
- Estado***: Dropdown menu com a opção 'Selecione o estado'.
- Cidade***: Dropdown menu com a opção 'Selecione a cidade'.
- Telefone***: Campo de texto para o telefone.
- Celular***: Campo de texto para o celular.
- E-mail***: Campo de texto para o e-mail.

Um botão 'Salvar' em azul está localizado na parte inferior esquerda do formulário.

Figura 26. Tela de cadastro de pessoa. Fonte: Elaborada pelo autor.

A Figura 27 apresenta a tela de pesquisa de pessoa, na qual os filtros, que podem ser aplicados em uma pesquisa estão na mesma ordem da tela de cadastro, facilitando para o usuário durante sua utilização.

Pesquisa de Pessoa Cadastro de Pessoa

Tipo pessoa: ☐ Física ☐ Jurídica

CPF/CNPJ:

Nome: Data Nascimento: Gênero: ☐ Masculino ☐ Feminino

Logradouro: Endereço: Número: Bairro: CEP:

País: Estado: Cidade:

Telefone: Celular: E-mail:

| Nome | Data Nascimento | CpfCnpj | Logradouro | Endereço | Número | Bairro | Cidade | Email |
|---------------------------------|-----------------|--------------------------------|------------|-----------|--------|--------|------------|------------------------|
| Pedro Henrique Rodrigues Soares | 11/11/1990 | 824.354.465-29 | Rua | Joinville | 580 | Centro | Acrelandia | pedro.soares@email.com |

Figura 27. Tela de pesquisa de pessoa. Fonte: Elaborada pelo autor.

Para o desenvolvimento das telas de pesquisa da aplicação se utilizou do *Hibernate Criteria*, responsável por realizar consultas, utilizando critérios específicos no banco de dados, uma vez que este uso possibilitou trazer somente resultados com base nos filtros aplicados para a tela de pesquisa.

A Figura 28 apresenta a tela de edição de pessoa, na qual se pode notar que a tela de edição é idêntica a de cadastro de pessoa, a não ser pelo conteúdo no topo da página de cada uma, em que se apresenta o registro de “Cadastro de Pessoa” em uma e na outra “Edição de ‘Nome da Pessoa’”, respectivamente, sendo esta técnica de usar de uma mesma tela tanto para o cadastro quanto para a edição foi possibilitada pelo Thymeleaf.

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/tcc/pessoas/1'. The page title is 'Edição de Pedro Henrique Rodrigues Soares'. The form contains the following fields and controls:

- Tipo pessoa***: Radio buttons for 'Física' (selected) and 'Jurídica'.
- CPF***: Text input with value '824.354.465-29'.
- Nome***: Text input with value 'Pedro Henrique Rodrigues Soares'.
- Data Nascimento***: Text input with value '11/11/1990'.
- Gênero***: Radio buttons for 'Masculino' (selected) and 'Feminino'.
- Logradouro***: Text input with value 'Rua'.
- Endereço***: Text input with value 'Joinville'.
- Número***: Text input with value '580'.
- Bairro***: Text input with value 'Centro'.
- CEP***: Text input with value '89.010-902'.
- Pais***: Dropdown menu with 'Brasil' selected.
- Estado***: Dropdown menu with 'Selecione o estado'.
- Cidade***: Dropdown menu with 'Selecione a cidade'.
- Telefone***: Text input with value '(47) 3336-5454'.
- Celular***: Text input with value '(47) 99965-8475'.
- E-mail***: Text input with value 'pedro.soares@email.com'.
- Salvar**: Blue button at the bottom left.

Figura 28. Tela de edição de pessoa. Fonte: Elaborada pelo autor.

As Figuras 29 e 30 apresentam a tela de cadastro de veículo, e nesta tela se podem perceber os diferenciais apresentados com relação à tela de cadastro de pessoa, sendo estes: a presença dos seletores de opcionais, como componente responsável por receber a foto do veículo a ser cadastrado.

A área de opcionais foi desenvolvida com apenas um *checkbox* (componente do tipo checagem de conteúdo), o Thymeleaf possibilitou gerar o restante dos componentes sem duplicação de código.

A seção de *upload* da foto foi possível utilizando o UIKit Upload, no qual se criou um espaço na página, possibilitando que o usuário arraste e solte a imagem ou selecione a partir dos arquivos do dispositivo usado.

Cadastro de Veículo

Pesquisar Veículos

Marca*
Selecione a marca

Modelo*
Selecione o modelo

País*
Selecione o país

Estado*
Selecione o estado

Cidade*
Selecione a cidade

Ano Fabricação*
Ano Modelo*

Placa*

KM*
Cor*
Selecione a cor

Combustível*
Selecione o combustível

Porta*
Selecione o número de portas

Valor*
R\$ 0

Opcionais

- ☐ Air bag
- ☐ Ar quente
- ☐ Ar-condicionado digital
- ☐ Câmbio automático
- ☐ Controle de som volante
- ☐ Direção hidráulica
- ☐ Engate de reboque
- ☐ Farol de neblina
- ☐ Kit multimídia
- ☐ Lona marítima
- ☐ Para-choque na cor
- ☐ Protetor de caçamba
- ☐ Sensor estacionamento
- ☐ Tração 4x4
- ☐ Vidros elétricos
- ☐ Alarme
- ☐ Ar-condicionado
- ☐ Bancos em couro
- ☐ Computador de bordo
- ☐ Desembaçador traseiro
- ☐ Encosto cab. traseiro
- ☐ Espelhos elétricos
- ☐ Freios ABS
- ☐ Limpador traseiro
- ☐ MP3 player
- ☐ Piloto automático
- ☐ Rodas de liga leve
- ☐ Teto solar
- ☐ Travas elétricas
- ☐ Volante escamoteável

Adicionais

- ☐ Blindado
- ☐ Garantia de fábrica
- ☐ Manual do proprietário
- ☐ Único dono
- ☐ Chave reserva
- ☐ IPVA pago
- ☐ Nota fiscal

Figura 29. Tela de cadastro de veículo. Fonte: Elaborada pelo autor.

Adicionais

- ☐ Blindado
- ☐ Garantia de fábrica
- ☐ Manual do proprietário
- ☐ Único dono
- ☐ Chave reserva
- ☐ IPVA pago
- ☐ Nota fiscal

Descrição

Foto

Arraste a foto aqui ou [selecione](#)

Salvar

Figura 30. Seção para envio de foto no cadastro de veículo. Fonte: Elaborada pelo autor.

A Figura 31 apresenta a tela de pesquisa de veículo, essa tela não possui grande diferença em relação ao que já foi exposto na tela de pesquisa de pessoa, a não ser pelo uso de imagens, nos resultados vindos do banco de dados.

A imagem que vem no resultado da pesquisa não é a mesma cadastrada na tela de cadastro de veículo, tendo em vista que se trata de um *thumbnail*, ou seja, uma cópia da imagem enviada pelo usuário, porém com tamanho reduzido, na qual a criação de uma cópia menor se fez necessária, de modo a: diminuir o tráfego de dados de rede, diminuir recursos de processamento utilizados no servidor ao buscar as imagens

vinculadas aos resultados da busca e apresentar na tela do usuário e, por fim, melhorar a usabilidade do usuário, pelo fato de não sobrecarregar o dispositivo, sendo utilizado para acessar o sistema.

The screenshot shows a web browser window with the title 'Pesquisar Veículos'. The address bar shows 'localhost:8080/tcc/veiculos/pesquisarveiculos'. The page has a navigation bar with links: 'Início', 'Pessoas', 'Veículos', and 'Vendas'. The main heading is 'Pesquisa de Veículo'. On the right, there is a button 'Cadastro de Veículo'. The search form includes the following fields:

- Placa:** A text input field.
- Marca:** A dropdown menu with 'Todas as marcas' selected.
- KM:** Two text input fields separated by 'até'.
- Modelo:** A dropdown menu with 'Todos os modelo' selected.
- Cor:** A dropdown menu with 'Todas as cores' selected.
- Combustível:** A dropdown menu with 'Todos os combustíveis' selected.
- Porta:** A dropdown menu with 'Todas as portas' selected.
- Preço unitário:** Two text input fields separated by 'até'.

Below the form is a blue button labeled 'Pesquisar'. Underneath, there is a table header with the following columns: Placa, Marca, Modelo, Combustível, Ano fabricação, Ano modelo, KM, Porta, Cor, and Valor. The table body currently displays the message 'Nenhum veículo encontrado.'

Figura 31. Tela de pesquisa de veículo. Fonte: Elaborada pelo autor.

A Figura 32 apresenta a tela de edição de veículo, sendo essa tela, assim como as telas de cadastro e edição de pessoa vistas como idênticas à tela de cadastro de veículo, a não ser pelo topo da página, em que se indica que se trata da edição do veículo de placa AAA1234.

Edição de AAA1234

Marca* Volkswagen
 Modelo* Novo Gol 1.0
 País* Brasil
 Estado* Santa Catarina
 Cidade* Blumenau
 Ano Fabricação* 2017
 Ano Modelo* 2017
 Placa* AAA1234
 KM* 0
 Cor* Amarela
 Combustível* Tricombustível
 Porta* 4
 Valor* R\$ 35.000

Opcionais

- ☒ Air bag
- ☒ Ar quente
- ☒ Ar-condicionado digital
- ☒ Câmbio automático
- ☒ Controle de som volante
- ☒ Direção hidráulica
- ☒ Engate de reboque
- ☒ Farol de neblina
- ☒ Kit multimídia
- ☐ Lona marítima
- ☒ Para-choque na cor
- ☐ Protetor de caçamba
- ☒ Sensor estacionamento
- ☐ Tração 4x4
- ☒ Vidros elétricos
- ☒ Alarma
- ☒ Ar-condicionado
- ☒ Bancos em couro
- ☒ Computador de bordo
- ☒ Desembaçador traseiro
- ☒ Encosto cab. traseiro
- ☒ Espelhos elétricos
- ☒ Freios ABS
- ☒ Limpador traseiro
- ☒ MP3 player
- ☐ Piloto automático
- ☒ Rodas de liga leve
- ☐ Teto solar
- ☒ Travas elétricas
- ☒ Volante escamoteável

Adicionais

- ☐ Blindado
- ☒ Garantia de fábrica
- ☒ Chave reserva
- ☒ IPVA nano

Figura 32. Tela de edição de veículo. Fonte: Elaborada pelo autor.

A Figura 33 apresenta a tela de cadastro de venda, responsável por realizar o vínculo de um veículo à uma pessoa anteriormente cadastrados no sistema de modo a registrar a saída de um veículo do estoque da revenda.

Cadastro de Venda

Pessoa Selecionar a pessoa
 Veículo* Selecionar o veículo

Observação

Salvar

Figura 33. Tela de cadastro de venda. Fonte: Elaborada pelo autor.

A Figura 34 apresenta a tela de pesquisa de venda.

Pesquisa de Venda Cadastro de Venda

Código

Data de criação até

Pessoa

Veículo

Pesquisar

| Código | Cliente | Data de criação | Vendedor |
|--------|---------------------------------|-----------------|----------|
| 1 | Pedro Henrique Rodrigues Soares | 28/06/17 00:24 | admin |

Figura 34. Tela de pesquisa de venda. Fonte: Elaborada pelo autor.

A Figura 35 apresenta a tela de cadastro de configurações do sistema, e nesta são definidos todos os cadastros utilizados no restante das telas.

Configurações do Siste

Usuários 1 Novo Pesquisar

Grupos 2 Novo Pesquisar

Permissões 32 Listar

Países 1 Novo Pesquisar

Estados 27 Novo Pesquisar

Cidades 5563 Novo Pesquisar

Marcas 51 Novo Pesquisar

Modelos 1 Novo Pesquisar

Combustíveis 8 Novo Pesquisar

Cores 18 Novo Pesquisar

Portas 2 Novo Pesquisar

Opcionais 30 Novo Pesquisar

Adicionais 7 Novo Pesquisar

Figura 35. Tela de cadastros de configurações. Fonte: Elaborada pelo autor.

A Figura 36 apresenta a tela de cadastro de usuário, em que cada usuário no

sistema é vinculado a uma pessoa, o que implica que uma pessoa poderá ter somente 1 (um) usuário cadastrado. As telas de cadastro de usuário, de pesquisa de usuário e de edição de usuário possuem um componente do tipo *checkbox* personalizado, utilizando Bootstrap, uma vez que define se o usuário é ativo ou inativo, permitindo ou não acesso ao sistema.

Usuários

localhost:8080/tcc/usuarios/novo

Início Pessoas Veículos Vendas

Pesquisar Usuários

Cadastro de Usuário

Pessoa
Selecione a pessoa

Nome de usuário
[Campo de texto]

Senha do usuário
[Campo de texto]

Confirmação de senha
[Campo de texto]

Status
☐ Ativo ☒ Inativo

Grupos
☐ Administrador ☐ Vendedor

Salvar

Figura 36. Tela de cadastro de usuário. Fonte: Elaborada pelo autor.

A Figura 37 apresenta a tela de pesquisa de usuário, na qual existe a possibilidade de inativar ou ativar vários usuários, simultaneamente, bastando selecionar os *checkboxes* dos usuários correspondentes.

Pesquisa de Usuário

Cadastro de Usuário

Pessoa: Seleccione a pessoa

Nome de usuário:

Status: ☐ Inativo

Grupos: ☐ Administrador ☐ Vendedor

Pesquisar

Ativar Desativar

| | Nome | Pessoa | Status | Grupos |
|--------------------------|-------|---------------------------------|--------|---------------|
| <input type="checkbox"/> | admin | Pedro Henrique Rodrigues Soares | Ativo | Administrador |

Figura 37. Tela de pesquisa de usuário. Fonte: Elaborada pelo autor.

A Figura 38 apresenta a tela de edição de usuário, e esta também apresenta o componente para ativar ou inativar o usuário, que está sendo editado. No exemplo da figura em print não se exhibe tal componente, pois o usuário a ser editado é o mesmo usuário, que está sendo editado, também não é possível efetuar a exclusão do usuário se este estiver logado.

Edição de admin

Pesquisar Usuários

Pessoa: Pedro Henrique Rodrigue

Nome de usuário: admin

Senha do usuário:

Confirmação de senha:

Grupos: ☒ Administrador ☐ Vendedor

Salvar

Figura 38. Tela de edição de usuário. Fonte: Elaborada pelo autor.

A Figura 39 apresenta a tela de cadastro de grupo, sendo este o que possibilita a separação de níveis de permissões para acesso às telas do sistema. No caso do grupo

Administradores, todos os *checkboxes* podem ser marcados definindo acesso a todas as funcionalidades, de maneira que se pode criar um grupo denominado de Vendedores para acesso somente à tela de pesquisa de veículos, por exemplo, em que todos os outros recursos estarão escondidos dos usuários vinculados a esse grupo.

Figura 39. Tela de cadastro de grupo. Fonte: Elaborada pelo autor.

A Figura 40 apresenta a tela de pesquisa de grupo, em que se pode filtrar a pesquisa, utilizando o nome do grupo ou parte dele, aplicando uma operação semelhante ao operador ‘*like*’ ao consultar o banco de dados.

Figura 40. Tela de pesquisa de grupo. Fonte: Elaborada pelo autor.

A Figura 41 apresenta a tela de edição de grupo, nesta tela pode-se tanto alterar

o nome do grupo quanto retirar ou acrescentar suas permissões vinculadas no sistema.

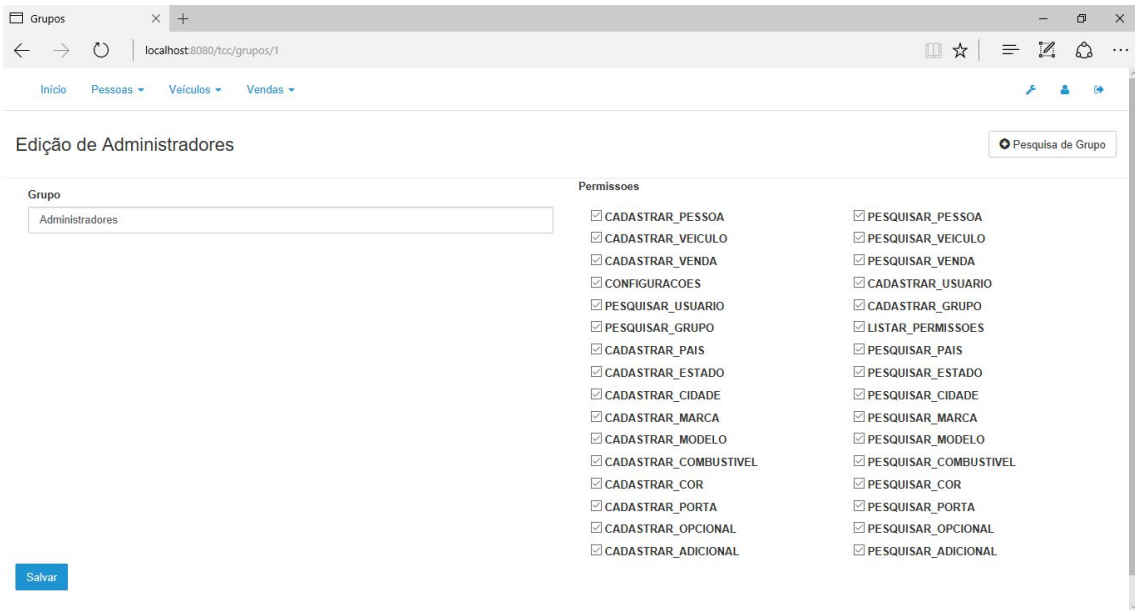


Figura 41. Tela de edição de grupo. Fonte: Elaborada pelo autor.

A Figura 42 apresenta a tela de listagem de permissões do sistema, desenvolvida para listar ao usuário quais permissões e telas são possíveis gerenciar juntamente dos grupos.

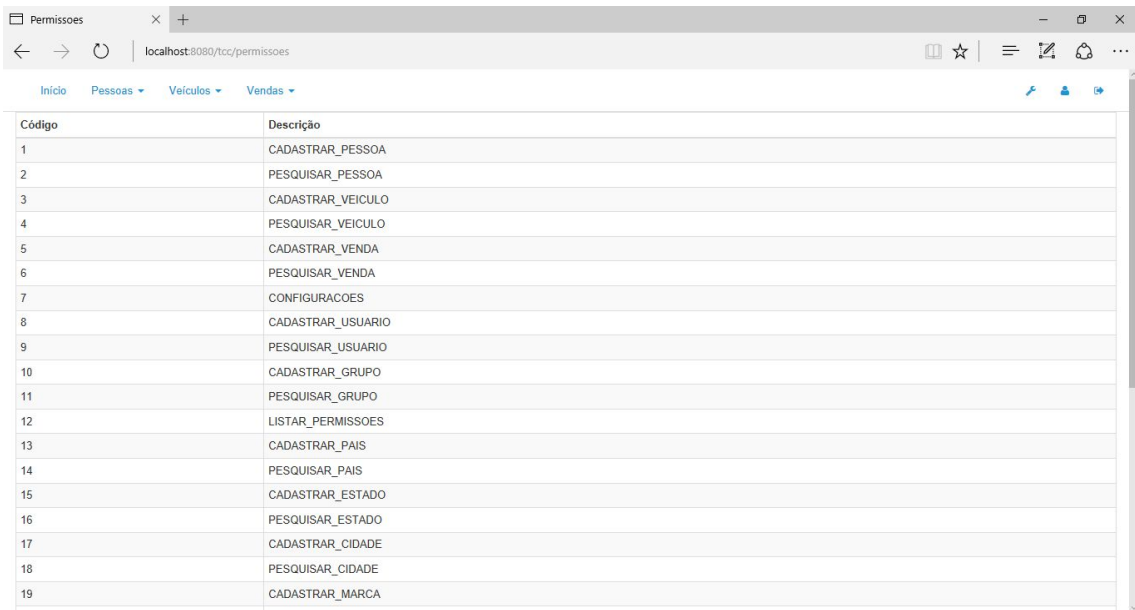
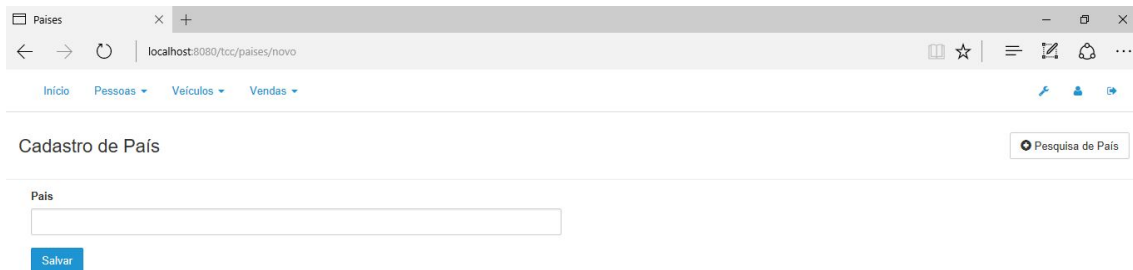


Figura 42. Tela de edição de listagem de permissões. Fonte: Elaborada pelo autor.

A Figura 43 apresenta a tela de cadastro de país, sendo estes dados utilizados nos cadastros de veículo e pessoa.



A captura de tela mostra uma interface web no navegador. No topo, há uma barra de navegação com links: 'Início', 'Pessoas', 'Veículos' e 'Vendas'. Abaixo, o título 'Cadastro de País' está à esquerda, e um botão 'Pesquisa de País' está à direita. O formulário principal contém um campo de texto rotulado 'País' e um botão azul 'Salvar' logo abaixo dele.

Figura 43. Tela de cadastro de país. Fonte: Elaborada pelo autor.

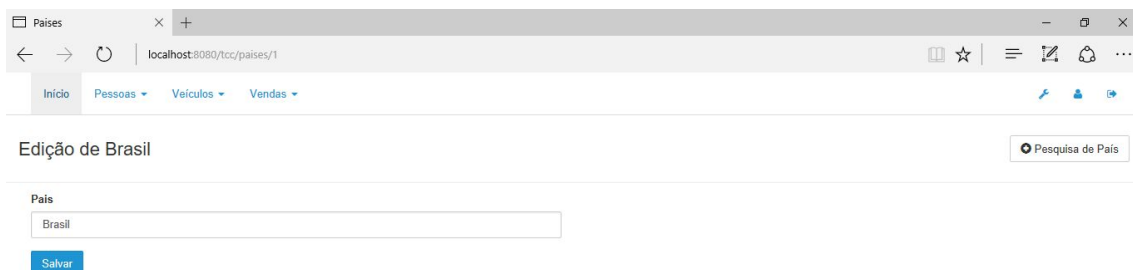
A Figura 44 apresenta a tela de pesquisa de país, em que se pode filtrar a pesquisa utilizando o nome do país ou parte dele.



A captura de tela mostra a interface de pesquisa. A barra de navegação é idêntica à da Figura 43. O título 'Pesquisa de País' está à esquerda, e um botão 'Cadastro de País' está à direita. O formulário possui um campo de texto rotulado 'País' e um botão azul 'Pesquisar' abaixo dele. Abaixo do formulário, há uma tabela com o cabeçalho 'País' e uma única linha de dados contendo 'Brasil', com ícones de edição e exclusão ao lado.

Figura 44. Tela de pesquisa de país. Fonte: Elaborada pelo autor.

A Figura 45 apresenta a tela de edição de país.



A captura de tela mostra a interface de edição. A barra de navegação é idêntica. O título 'Edição de Brasil' está à esquerda, e um botão 'Pesquisa de País' está à direita. O formulário contém um campo de texto rotulado 'País' com o valor 'Brasil' preenchido e um botão azul 'Salvar' abaixo dele.

Figura 45. Tela de edição de país. Fonte: Elaborada pelo autor.

O sistema possui mais vinte e uma telas dentro da tela de configurações, sendo todas estas destinadas ao gerenciamento dos cadastros de Estados, cidades, cores, opcionais, adicionais entre outras, todas semelhantes se comparadas às telas de gerenciamento de cadastro de país.

As Figuras 46, 47 e 48 apresentam o funcionamento das telas de login do sistema, tela inicial e tela de cadastro de usuário respectivamente, tais telas foram acessadas a partir de um dispositivo móvel, exibindo a característica de responsividade, ou seja, estão em adequação com os componentes das páginas à telas menores.



Figura 46. Tela de acesso ao sistema em dispositivo móvel. Fonte: Elaborada pelo autor.



Figura 47. Tela inicial do sistema em dispositivo móvel. Fonte: Elaborada pelo autor.

Figura 48. Tela de cadastro de veículo em dispositivo móvel. Fonte: Elaborada pelo autor.

Realizou-se o acesso ao sistema, por meio dos três principais navegadores, sendo estes: Internet Explorer, Google Chrome e Mozilla Firefox, e nenhum deles apresentou erros ao acessar a aplicação.

7. Resultados Obtidos

A solução desenvolvida por este trabalho, apesar de possuir menos funcionalidades, em comparação com a relação exposta dos sistemas correlatos, oferece vantagens, como: aplicação gratuita, cadastro e armazenamento de veículos, usuários e clientes ilimitados, dependendo apenas da capacidade do computador que a executa.

Todos os requisitos levantados foram atingidos na aplicação desenvolvida. O objetivo maior que envolve transformar todas as informações levantadas, em um sistema, utilizável pelo usuário, também foi atingido e, com isso, o VendaAuto se encontra pronto para uso em sua primeira versão.

8. Considerações Finais e Trabalhos Futuros

O trabalho superou as expectativas criadas inicialmente, tendo em vista que foi possível resolver todas as necessidades levantadas como requisitos e até superá-las com funcionalidades e características melhores. Não se imaginou possibilitar uma área para *upload* de foto, campos para associação de opcionais e adicionais, na tela de cadastro de veículos, uma vez que tais características foram desenvolvidas como pontos extras diante do planejado para o trabalho.

Entre as tarefas que se almejam realizar futuramente, pode-se destacar o desenvolvimento de uma interface pública para a disponibilização dos veículos cadastrados pela revenda, o atendimento a outras necessidades do dia a dia da loja, como cópias e armazenamento de documentos e, por fim, integrar o VendaAuto com serviços de terceiros, de modo a possibilitar a consulta do preço médio, em órgãos de pesquisa de preços, em que a situação cadastral do veículo possa ser verificada utilizando a base de dados do órgão regulador responsável.

Referências Bibliográficas

ALTIMUS. **O melhor sistema Web para lojas de veículos.** Disponível em: <<http://www.altimus.com.br/>>. Acesso em: 26 jun. 2017.

APACHE. **Apache Log4j 2.** Disponível em: <<https://logging.apache.org/log4j/2.x/>>. Acesso em: 26 jun. 2017.

APACHE. **Apache Tomcat.** Disponível em: <<http://tomcat.apache.org/>>. Acesso em: 26 jun. 2017.

ASTAH. **Astah Community.** Disponível em: <<http://astah.net/editions/community>>. Acesso em: 26 jun. 2017.

AUTOWEB. **Sistema web para lojas de veículos.** Disponível em: <<http://www.autoweb.com.br/>>. Acesso em: 26 jun. 2017.

BOOTSTRAP, Site de Componentes do. **Bootstrap Components.** Disponível em: <<http://getbootstrap.com/components/#pagination>>. Acesso em: 21 jun. 2017.

CAELUM. **APOSTILA JAVA PARA DESENVOLVIMENTO WEB.** Disponível em: <<https://www.caelum.com.br/apostila-java-web/spring-mvc/>>. Acesso em: 21 jun. 2017.

ECLIPSE. **Desktop IDEs.** Disponível em: <<http://www.eclipse.org/ide/>>. Acesso em: 21 jun. 2017.

EBERSON, Steve. **Hibernate ORM.** Disponível em: <http://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html>. Acesso em: 21 jun. 2017.

FAKHROUTDINOV, Kirill. **The Unified Modeling Language.** Disponível em: <<http://www.uml-diagrams.org/>>. Acesso em: 26 jun. 2017.

FONTAWESOME. **Get Started.** Disponível em: <<http://fontawesome.io/get-started/>>. Acesso em: 21 jun. 2017.

GITHUB. **GitHub - FasterXML/jackson-databind: General data-binding package for Jackson (2.x): works on streaming API (core) implementation(s).** Disponível em: <<https://github.com/FasterXML/jackson-databind/>>. Acesso em: 26 jun. 2017.

GITHUB. **Thumbnailator is a thumbnail generation library for Java.** Disponível

em: <<https://github.com/coobird/thumbnailator>>. Acesso em: 21 jun. 2017.

HIBERNATE. **Hibernate Validator**. Disponível em: <<http://hibernate.org/validator/>>. Acesso em: 26 jun. 2017.

HO, Don. **About**. Disponível em: <<https://notepad-plus-plus.org/>>. Acesso em: 26 jun. 2017.

JQUERY. **jQuery API**. Disponível em: <<https://api.jquery.com/>>. Acesso em: 26 jun. 2017.

LEARY, Sean. **Introducing JSON**. Disponível em: <<http://www.json.org/>>. Acesso em: 26 jun. 2017.

ORACLE. **MySQL Documentation**. Disponível em: <<https://dev.mysql.com/doc/>>. Acesso em: 26 jun. 2017.

ORACLE. **MySQL Connector/J 5.1 Developer Guide**. Disponível em: <<https://dev.mysql.com/doc/connector-j/5.1/en/>>. Acesso em: 26 jun. 2017.

ORACLE. **MySQL Workbench**. Disponível em: <<https://www.mysql.com/products/workbench/>>. Acesso em: 26 jun. 2017.

PIVOTAL. **Spring Boot**. Disponível em: <<https://projects.spring.io/spring-boot/>>. Acesso em: 26 jun. 2017.

PIVOTAL. **Spring Security**. Disponível em: <<https://projects.spring.io/spring-security/>>. Acesso em: 26 jun. 2017.

PIVOTAL. **Spring Framework**. Disponível em: <<https://projects.spring.io/spring-framework/>>. Acesso em: 26 jun. 2017.

PIVOTAL. **Spring Data JPA**. Disponível em: <<https://projects.spring.io/spring-data-jpa/>>. Acesso em: 26 jun. 2017.

PRESSMAN, Roger S.. **Engenharia de Software - Uma Abordagem Profissional**. 7. ed. Porto Alegre: Amgh Editora, 2011.

REVENDAMAIS. **Sistema para lojas de veículos**. Disponível em: <<http://www.revendamais.com.br/>>. Acesso em: 26 jun. 2017.

JAVASCRIPT. **Ready to try JavaScript?** Disponível em: <<https://www.javascript.com/>>. Acesso em: 26 jun. 2017.

SMARTDRAW. **Entity Relationship Diagram**. Disponível em:

<<https://www.smartdraw.com/entity-relationship-diagram/>>. Acesso em: 26 jun. 2017.

SPRING Framework Expert. Realização de Thiago Faria de Andrade. Uberlândia: Algaworks, 2015. Son., color.

THYMELEAF. **Getting started with the Standard dialects in 5 minutes.** Disponível em: <<http://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>>. Acesso em: 21 jun. 2017.

THYMELEAF. **Thymeleaf 3.** Disponível em: <<http://www.thymeleaf.org/documentation.html>>. Acesso em: 21 jun. 2017.

THYMELEAF. **Thymeleaf Page Layouts.** Disponível em: <<http://www.thymeleaf.org/doc/articles/layouts.html>>. Acesso em: 21 jun. 2017.

TURNAROUND, Zero. **Reload code changes instantly.** Disponível em: <<https://zeroturnaround.com/software/jrebel/>>. Acesso em: 26 jun. 2017.

UIKIT. **Upload Component.** Disponível em: <<https://getuikit.com/v2/docs/upload.html>>. Acesso em: 21 jun. 2017.

W3SCHOOLS. **HTML5 Introduction.** Disponível em: <https://www.w3schools.com/html/html5_intro.asp>. Acesso em: 26 jun. 2017.

W3SCHOOLS. **CSS Tutorial.** Disponível em: <<https://www.w3schools.com/css/>>. Acesso em: 26 jun. 2017.

APÊNDICE A– Descrição dos casos de uso

| Nome: | Autenticar |
|----------------------|--|
| Descrição: | A tela de autenticação aparecerá para o |
| RF016 | usuário e ele deverá informar nos campos |
| Ator: Usuário | indicados o seu nome de usuário e senha. |
| Nome: | Manter Veículos |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF001 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e Excluir veículos do sistema. |
| Nome: | Manter Pessoas |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF002 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e Excluir pessoas do sistema. |
| Nome: | Manter Vendas |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF003 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar e Pesquisar vendas no sistema. |

| | |
|----------------------|--|
| Nome: | Manter Usuários |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF004 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir usuários do sistema. |
| | |
| Nome: | Manter Grupos |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF005 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir grupos do sistema. |
| | |
| Nome: | Manter Configurações |
| | |
| Descrição: | Ao selecionar esta opção, o usuário de |
| Ator: Usuário | acordo com suas permissões de acesso |
| | visualizará as configurações e cadastros |
| | do sistema. |
| | |
| Nome: | Manter Permissões |
| | |
| Descrição: | Ao selecionar esta opção, o usuário de |
| Ator: Usuário | acordo com suas permissões de acesso |
| | visualizará as permissões de acesso às |
| | telas disponíveis do sistema. |

| | |
|----------------------|--|
| Nome: | Manter Países |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF006 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir países do sistema. |
| | |
| Nome: | Manter Estados |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF007 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir estados do sistema. |
| | |
| Nome: | Manter Cidades |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF008 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir cidades do sistema. |
| | |
| Nome: | Manter Marcas |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF009 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir marcas do sistema. |

| | |
|----------------------|---|
| Nome: | Manter Modelos |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF010 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir modelos do sistema. |
| | |
| Nome: | Manter Combustíveis |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF011 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir combustíveis do sistema. |
| | |
| Nome: | Manter Cores |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF012 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir cores do sistema. |
| | |
| Nome: | Manter Portas |
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF013 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir quantidade de portas dos veículos |

do sistema.

| Nome: | Manter Opcionais |
|----------------------|--|
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF014 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir opcionais do sistema. |

| Nome: | Manter Adicionais |
|----------------------|--|
| | |
| Descrição: | Ao selecionar esta opção, o usuário |
| RF15 | poderá de acordo com suas permissões de |
| Ator: Usuário | acesso Cadastrar, Pesquisar, Atualizar e |
| | Excluir adicionais do sistema. |

APÊNDICE B – Diagrama de casos de uso

