CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

MICAEL MUNSFELD

MCC: PROTÓTIPO DE SOFTWARE WEB PARA GERENCIAR LOJAS DE VEÍCULOS

RIO DO SUL

2024

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

MICAEL MUNSFELD

MCC: PROTÓTIPO DE SOFTWARE WEB PARA GERENCIAR LOJAS DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: M.e Jullian Hermann Creutzberg

RIO DO SUL

2024

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

MICAEL MUNSFELD

MCC: PROTÓTIPO DE SOFTWARE WEB PARA GERENCIAR LOJAS DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: M.e Jullian Hermann Creutzberg

Banca Examinadora:

_____

Prof. M.e Fernando Andrade Bastos

_____

Prof. Cleber Nardelli

Rio do Sul, 26 de novembro de 2024.

RESUMO

Mesmo com os amplos avanços tecnológicos e a presença constante da tecnologia em nosso cotidiano, o setor de veículos, especialmente em lojas de veículos, ainda enfrenta desafios significativos na gestão eficiente de seus estoques e operações. A ausência de sistemas informatizados dedicados pode resultar em desorganização e falta de precisão nos registros, dificultando a análise de dados e a tomada de decisões estratégicas. Este trabalho tem como objetivo principal o desenvolvimento de um protótipo de software web, denominado MCC (*Munsfeld Car Control*), voltado para a gestão em lojas de veículos. O sistema visa facilitar o controle de entrada e saída de veículos, o cadastro de clientes e lojistas, bem como a consulta à tabela FIPE (Fundação Instituto de Pesquisas Econômicas), fornecendo informações de preços atualizadas diretamente no sistema. Quanto à metodologia, o estudo foi conduzido como uma pesquisa aplicada e descritiva. Foram realizadas pesquisas bibliográficas sobre tecnologias e ferramentas voltadas ao desenvolvimento do protótipo, sendo que estas estão detalhadas no capítulo de revisão da literatura. Além disso, foi realizada uma análise de sistemas semelhantes que é apresentada na seção do estado da arte. Na fase de desenvolvimento, foram levantados os requisitos funcionais e não funcionais, assim como as regras de negócio pertinentes. Na seção de implementação, são descritos todos os detalhes técnicos do desenvolvimento do protótipo, abordando desde a criação de diagramas até a escolha das tecnologias. A utilização do protótipo MCC pode trazer benefícios para lojas que desejam melhorar ou que ainda não possuem um sistema automatizado, permitindo maior controle sobre o estoque de veículos e melhorando a eficiência operacional. Outro benefício é a integração com a tabela FIPE, que permite consultas automáticas de preços, eliminando a necessidade de acessar o site para realizar uma consulta manualmente.

Palavras-Chave: lojas de veículos, desenvolvimento web, sistemas de informação.

ABSTRACT

Even with major technological advances and the constant presence of technology in our daily lives, the automotive sector, especially car dealerships, still faces significant challenges in efficiently managing their inventories and operations. The absence of dedicated computerized systems can result in disorganization and lack of accuracy in records, making data analysis and strategic decision-making difficult. This work has as its main objective the development of a web software prototype, called MCC (*Munsfeld Car Control*), aimed at management in vehicle stores. The system aims to facilitate the control of vehicle entry and exit, the registration of customers and store owners, as well as consultation of the FIPE (Fundação Instituto de Pesquisas Econômicas), providing updated price information directly in the system. Regarding the methodology, the study was conducted as an applied and descriptive research. Bibliographic research was carried out on technologies and tools for developing the prototype, which are detailed in the literature review chapter. In addition, an analysis of similar systems was performed, presented in the state of the art section. In the development phase, functional and non-functional requirements were raised, as well as the relevant business rules. In the implementation section, all technical details of prototype development are described, ranging from creating diagrams to choosing technologies. The use of the MCC prototype can bring benefits to stores that want to improve or that do not yet have an automated system, allowing greater control over vehicle inventory and improving operational efficiency. Another benefit is the integration with the FIPE table, which allows automatic price queries, eliminating the need to access the website to perform the query manually.

Keywords: vehicle stores, web development, information systems.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

No cenário atual, a tecnologia desempenha um papel fundamental em diversos setores, incluindo a gestão de negócios. A importância de um software eficiente para a administração empresarial não pode ser subestimada. Em particular, o uso de sistemas informatizados em lojas e concessionárias de veículos tem se mostrado essencial para otimizar processos, melhorar a eficiência operacional e garantir a transparência dos dados.

O controle da entrada e saída de veículos, bem como a manutenção de um histórico dessas movimentações, representam um desafio significativo para muitos lojistas que ainda não adotaram um sistema de gestão. A ausência de um *software* dedicado pode resultar em desorganização, falta de precisão nos registros e dificuldades na análise de dados, impactando negativamente a tomada de decisões e a performance geral do negócio.

Diante desse contexto, surge a oportunidade de desenvolver soluções tecnológicas que atendam às especificidades desse setor. Este trabalho tem como objetivo principal desenvolver um protótipo de software web para gerenciar a movimentação de veículos em lojas, denominado MCC (*Munsfeld Car Control*). O MCC visa proporcionar um controle eficaz do estoque de veículos, incluindo a análise da disponibilidade, a distribuição por marca e modelo, o cadastro de clientes e lojistas.

A relevância deste estudo é evidente na capacidade do MCC de transformar dados brutos em informações, essenciais para uma gestão eficiente de vendas e a maximização dos resultados empresariais. Em um ambiente de negócios competitivo, onde a agilidade e a precisão são cruciais, a implementação de um sistema como o MCC oferece uma abordagem simplificada e direta para melhorar a administração de lojas de veículos, alinhando-se aos objetivos financeiros das empresas.

Sendo assim, a criação deste protótipo não visa apenas a modernização e organização das lojas de veículos, mas também se posiciona como uma ferramenta que visa aprimorar a gestão de vendas, promover a transparência e aumentar a eficiência operacional.

1.1 PROBLEMA DE PESQUISA

Como gerenciar de forma eficiente a entrada e saída de estoque em lojas de veículos?

1.2 OBJETIVOS

1.2.1 Geral

- Desenvolver um protótipo de software web, denominado MCC (*Munsfeld Car Control*), voltado para a gestão em lojas de veículos.

1.2.2 Específicos

- Detalhar as tecnologias empregadas no desenvolvimento do protótipo por meio de uma revisão da literatura.
- Analisar softwares semelhantes ao protótipo proposto.
- Identificar os requisitos funcionais, não funcionais e as regras de negócio do protótipo por meio da seção de análise.
- Empregar no desenvolvimento do protótipo as tecnologias escolhidas.

1.3 JUSTIFICATIVA

No contexto atual, marcado por avanços tecnológicos constantes e pela competitividade no mercado, a gestão organizada e eficiente de lojas de veículos tem se mostrado um diferencial importante para as empresas que buscam otimizar seus processos e melhorar a operação. A crescente quantidade de informações e a necessidade de gerenciar dados como estoque, entrada e saída de veículos, além do atendimento a clientes, podem se tornar desafios para lojas que ainda não contam com sistemas informatizados para apoiar essas atividades.

O desenvolvimento do protótipo MCC (Munsfeld Car Control) busca oferecer uma solução que auxilie lojistas no dia a dia, organizando informações de forma mais clara e acessível. O protótipo é projetado para facilitar o controle de estoque, a gestão de veículos disponíveis e o cadastro de clientes e lojistas, além de integrar consultas automatizadas à tabela

FIPE, permitindo maior praticidade na obtenção de valores de mercado. Dessa forma, o MCC pretende contribuir para simplificar tarefas administrativas e apoiar as decisões operacionais.

Outro aspecto relevante é que o protótipo pode atender especialmente pequenas e médias empresas, como a Sálvio Automóveis, que ainda realizam muitas atividades manualmente. Ao oferecer uma ferramenta que organiza e centraliza informações, o MCC pode proporcionar melhorias na gestão de veículos e no atendimento ao cliente, sem exigir grandes investimentos ou mudanças estruturais.

Além disso, o MCC se posiciona como um recurso que pode apoiar lojistas no acompanhamento de suas operações, tornando-as mais organizadas e alinhadas às demandas do mercado. A possibilidade de registrar, atualizar e consultar informações de forma simplificada e integrada, com funcionalidades como consulta à tabela FIPE, pode economizar tempo e reduzir erros, promovendo uma rotina mais prática e eficiente.

Portanto, este trabalho visa apresentar o MCC como um protótipo de apoio à gestão de lojas de veículos, buscando atender a demandas específicas e promover melhorias graduais nas operações. Embora o foco principal seja auxiliar lojistas em suas tarefas diárias, espera-se que o protótipo também contribua para estimular reflexões sobre como as soluções tecnológicas podem contribuir para a evolução e aprimoramento dos processos de gestão nesse setor. O desenvolvimento do MCC representa, assim, um passo inicial em direção a processos mais organizados e eficientes, com benefícios tanto para as empresas quanto para seus clientes.

1.4 CONTEXTUALIZAÇÃO DA EMPRESA

A Sálvio Automóveis é uma empresa de venda de veículos fundada no início dos anos 2000, localizada na cidade de Imbuia, Santa Catarina. Desde sua criação, a Sálvio Automóveis estabeleceu-se como uma referência no mercado local de veículos, oferecendo uma variedade de automóveis novos e usados.

A empresa recebeu seu nome em homenagem ao fundador, Sálvio, cuja visão e dedicação foram fundamentais para o sucesso inicial do negócio. Após o falecimento de Sálvio, seu filho assumiu a gestão da empresa, mantendo o nome em homenagem ao pai e continuando a tradição de excelência e compromisso com os clientes que caracterizam a Sálvio Automóveis.

Hoje, a empresa continua a ser um pilar na cidade de Imbuia e região, conhecida por sua integridade e serviço de qualidade. No entanto, apesar de seu sucesso e tradição, a Sálvio Automóveis ainda não possui um software para gerenciar seu negócio, o que representa uma oportunidade para modernização e aumento da eficiência operacional.

1. REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão da literatura das tecnologias e conceitos relativos ao desenvolvimento de software, engenharia de *software*, linguagens de programação, *frameworks* e bibliotecas utilizadas na criação do protótipo, que são apresentados a seguir.

2.1 ENGENHARIA DE SOFTWARE

Pressman e Maxim (2021) relatam que a engenharia de software abrange processos, métodos e ferramentas que facilitam a criação de sistemas complexos baseados em computador, cumprindo prazos e mantendo a qualidade. O processo de *software* inclui cinco atividades estruturais: comunicação, planejamento, modelagem, construção e entrega, cobrindo todos os projetos de *software*. Praticar engenharia de *software* é uma atividade de resolução de problemas que segue princípios básicos, quanto mais se aprende sobre essa área, começa a entender porque esses princípios são essenciais ao iniciar qualquer projeto de software.

A engenharia de sistemas visa compreender as necessidades de negócio do cliente, a partir das quais são estabelecidos requisitos de maneira sistêmica. Isso permite definir, em alto nível, o escopo do que será desenvolvido. Uma das etapas cruciais é a identificação das funções, restrições e desempenho do sistema, visando a execução de métodos, procedimentos ou processamento de dados (Hirama, 2011).

2.1.1 Levantamento de Requisitos

De acordo com Vetorazzo (2018, p.86) “Requisitos correspondem às necessidades e restrições do produto de software e ajudam nas soluções e nos problemas do mundo real. Nessa área, temos a elicitação, a análise, as especificações e a validação dos requisitos funcionais e não funcionais”.

Vetorazzo (2018) complementa dizendo que na maioria dos projetos de *software*, as principais falhas surgem da dificuldade em compreender as necessidades dos usuários. Por isso, realizar um levantamento de requisitos eficiente é de extrema importância. E menciona que a área de requisitos de *software* está dividida em 7 subáreas.

As subáreas mencionadas podem ser observadas no Quadro 1.

Quadro 1 - Subáreas de Levantamento de Requisitos

Fundamentos dos requisitos de softwares	Requisitos são características ou propriedades que devem refletir as necessidades do mundo real com o objetivo de solucionar um problema. Eles podem ser categorizados em requisitos de produto, processo, funcionais, não funcionais e emergentes.
Processo de requisitos	Essa disciplina abrange o planejamento de requisitos, delineando a integração dos processos de requisitos com os processos de desenvolvimento de software. Ela se divide em quatro subáreas: modelos de processos, atores envolvidos nos processos, gestão e suporte do processo, e melhoria contínua da qualidade do processo.
Elicitação de requisitos	Essa área concentra-se na captura e aquisição dos requisitos de software, identificando suas fontes. Ela é subdividida em duas áreas: fontes de requisitos e técnicas de elicitação.
Análise de requisitos	A análise desempenha um papel crucial ao conectar a alocação de software em nível de sistema ao projeto de software, auxiliando os engenheiros a melhorar e construir modelos. Para descrever requisitos com precisão, é essencial identificar e resolver conflitos entre requisitos e compreender a interação do software com o ambiente. Essa área é dividida em quatro subáreas: classificação de requisitos, modelagem conceitual, projeto arquitetural e negociação de requisitos.
Especificação de requisitos	A especificação de requisitos envolve a atribuição de valores numéricos aos objetivos do projeto, mas sua principal atividade é a documentação do sistema. Essa fase é subdividida em três áreas: documentação da definição do sistema, especificação dos requisitos do sistema e especificação dos requisitos de software.
Validação de requisitos	A documentação produzida durante a fase de especificação de requisitos pode ser empregada para verificar a conformidade com os padrões da organização. Esta etapa busca identificar e listar os problemas encontrados e inclui as seguintes etapas: revisão dos requisitos, prototipagem, validação de modelos e testes de aceitação.
Considerações práticas	Nesta área, são delineados tópicos que visam à compreensão prática e à validação dos atributos e do impacto das alterações nos requisitos, bem como à estimativa de custos para o desenvolvimento e a manutenção.

Fonte: Elaborado a partir de Vetorazzo (2018).

2.1.2 Regras de Negócio

Regras de negócio no olhar de Paula Filho (2019, p.139) “São regras que definem ou restringem aspectos dos processos de negócio. Elas são elementos do modelo de negócio, e devem ser levantadas durante a modelagem de negócio, se essa atividade for executada.”.

Paula Filho (2019) ainda destaca que um modelo de requisitos deve incluir todas as regras relevantes para o produto especificado. Para garantir que as regras de negócio sejam sempre claras e bem definidas, é essencial realizar *workshops* e entrevistas de requisitos. Frequentemente, os usuários só percebem essas regras após receberem *feedback* de um protótipo ou mesmo depois do lançamento do produto, devido à falta de detalhes explícitos durante a fase inicial.

2.1.3 Requisitos Funcionais e Não Funcionais

De acordo com Pressman (1995), a análise de requisitos conecta o *software* aos níveis de sistema e projeto, permitindo a especificação das funcionalidades e do desempenho do *software*, bem como as restrições do sistema. O analista desenvolve modelos dos processos de dados e dos domínios comportamentais do *software*. Essa análise fornece uma representação da informação que pode ser convertida em projeto procedimental, arquitetônico e de dados.

> O amplo espectro de tarefas e técnicas que levam a um entendimento dos requisitos é chamado de engenharia de requisitos. Do ponto de vista do processo de software, a engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho. (Pressman, 2021, p.103).

Pressman (2021, p.109) afirma que "Um requisito não funcional pode ser descrito como um atributo de qualidade, de desempenho, de segurança ou como uma restrição geral em um sistema. Frequentemente, os envolvidos têm dificuldade de articulá-los.".

Sommerville (2011) complementa dizendo que requisitos funcionais especificam as funções que o sistema deve realizar, enquanto os requisitos não funcionais não estão diretamente relacionados às ações do sistema, mas podem se referir às suas características de qualidade e desempenho.

Quadro 2 - Requisitos de Software

Requisito	**Concepção**
Funcional	Os requisitos funcionais delineiam as ações que o sistema deve executar, enquanto os requisitos não funcionais não estão diretamente ligados às operações do sistema, mas sim às suas características gerais. Embora não se refiram diretamente às funcionalidades específicas do sistema, os requisitos não funcionais podem estar relacionados ao seu desempenho, segurança e outros aspectos que contribuem para sua eficácia e usabilidade.
Não Funcional	Os requisitos não funcionais representam limitações sobre os serviços ou funcionalidades fornecidas pelo sistema. Eles abrangem restrições temporais, limitações no processo de desenvolvimento e conformidade com normas estabelecidas. Diferentemente das características ou serviços específicos do sistema, os requisitos não funcionais frequentemente se aplicam ao sistema como um todo, influenciando sua arquitetura e desempenho globais.

Fonte: Elaborado a partir de Sommerville (2011).

2.2 HTML

Alves (2021, p.7) afirma que “Assim como são necessárias linguagens de programação para o desenvolvimento de programas e aplicativos, também é necessária uma linguagem para criação de páginas de sites. Essa linguagem é denominada *HTML* (*HyperText Markup Language*, em português Linguagem de Marcação de Hipertexto)”.

Na visão de Miletto e Bertagnolli (2014) *HTML* é a linguagem fundamental para criar páginas da web, composta por um conjunto de *TAG’s* que permitem a exibição de conteúdo e o uso de recursos hipermídia, como links, imagens, tabelas e vídeos. No entanto, seus recursos de formatação visual são limitados e básicos.

2.3 CSS

Abraham (2019), afirma que *CSS* (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) é um código que modifica o *HTML* nas páginas e controla a aparência do conteúdo alterando tamanho de textos e imagens e outros atributos.

De acordo com Alves (2021), as folhas de estilo *CSS* contribuem para a melhoria das interfaces do usuário, oferecendo diversas opções que vão desde a criação de menus até o uso de efeitos especiais em imagens.

2.4 JAVASCRIPT

Segundo Abraham (2019), *JavaScript* adiciona interatividade e animação a páginas. O *JavaScript* também detecta eventos no navegador, como cliques do mouse, valida as inserções de usuários, como as textuais, e recupera dados de sites externos.

Flanagan (2014) afirma que o *JavaScript* é uma linguagem de alto nível, dinâmica, interpretada e não tipada, o que é essencial para a programação orientada a objetos e funcionais.

> JavaScript foi criada na Netscape na fase inicial da Web e, tecnicamente, “JavaScript” é marca registrada, licenciada pela Sun Microsystems (agora Oracle), usada para descrever a implementação da linguagem pelo Netscape (agora Mozilla). A Netscape enviou a linguagem para a ECMA – *European Computer Manufacturer’s Association* – para padronização e, devido a questões relacionadas à marca registrada, a versão padronizada manteve o nome estranho “ECMAScript”. (Flanagan, 2014, p. 19).

Flanagan (2014) diz que uma linguagem deve possuir pelo menos uma plataforma, biblioteca padrão ou até mesmo uma *API* (*Application Programming Interface* ou Interface de Programação de Aplicação) de funções (que trate coisas como entrada e saída básicas) para assim se tratar de uma linguagem útil.

Flanagan (2014, p. 19) afirma que "A linguagem *JavaScript* básica define uma *API* mínima para trabalhar com texto, *arrays*, datas e expressões regulares, mas não inclui funcionalidade alguma de entrada ou saída.".

Oliveira e Zanetti (2020) explicam que o *JavaScript* é uma linguagem de programação orientada a objetos interpretada e executada pelo navegador *web* (*server-side script*). Na qual possui uma sintaxe semelhante à linguagem Java e seu principal objetivo é fornecer maior interatividade às páginas. Uma característica importante do *JavaScript* é a ausência de tipos de dados fixos, todas as variáveis são do tipo variante. Isso significa que o tipo de dados é determinado pela informação armazenada no momento da definição da variável e pode ser alterado ao longo da execução da aplicação conforme seu conteúdo é modificado.

Na Figura 1 é demonstrado que a estrutura do *JavaScript* faz uso de funções, objetos, eventos, propriedades e métodos de maneira semelhante aos fundamentos das linguagens orientadas a objetos.

Figura 1 - Estrutura de um código fonte de Javascript

Fonte: Oliveira e Zanetti (2020, p.47).

2.5 PHP

O *PHP* (*Hypertext Preprocessor*) de acordo com Miletto e Bertagnolli (2014), foi concebido por Ramus Ledorf em 1995, é uma linguagem projetada para criar *scripts* comumente interpretados em servidores *web*. A interpretação desses scripts requer a configuração adequada do interpretador *PHP* no servidor. Além disso, os scripts *PHP* podem ser executados localmente através da linha de comando, desde que um interpretador esteja disponível. Um destaque do *PHP* é sua integração fluida com as marcações *HTML*, possibilitando a adição de dinamismo às páginas desenvolvidas com esta linguagem. Para identificar os trechos a serem interpretados como scripts *PHP* pelo servidor web, é necessário delimitá-los com *TAG'S* iniciais, diferenciando-os do *HTML* ou *JavaScript*.

Miletto e Bertagnolli (2014, p.172) definem o *PHP* da seguinte forma "*PHP* é uma linguagem de *script open source* de uso geral muito utilizada para o desenvolvimento de aplicações *web* integradas com códigos *HTML*." e exemplificam uma *TAG PHP*, conforme a Figura 2.

Figura 2 - Exemplo de uma TAG PHP

Fonte: Miletto e Bertagnolli (2014, p.172)

2.5.1 Variáveis

Para declarar uma variável em *PHP*, é preciso adicionar um $ antes de uma sequência de caracteres que deve começar com uma letra do alfabeto ou um sublinhado. Além disso, as variáveis no *PHP* diferenciam maiúsculas de minúsculas, o que significa que $Sistemas e $SISTEMAS são consideradas variáveis distintas. (Lobo, 2007).

Duckett (2024, p.18) diz que variáveis utilizam um nome para representar um valor que pode ser alterado cada vez que uma página *PHP* é requisitada e complementa "Nome descreve o tipo de dado que a variável contém e valor é o que a variável deve armazenar cada vez que a página for solicitada".

2.5.2 Padrão MVC (*Model, View, Controller*)

Segundo Turini (2015) A estrutura *MVC*, ou *Model View Controller*, serve para dividir de uma forma melhor as responsabilidades da aplicação.

Turini (2015, p.20) afirma que "A grande ideia desse padrão arquitetural é que você separe suas regras de negócio em 3 camadas, cada uma com sua responsabilidade muito bem definida.", como enfatiza o Quadro 3.

Quadro 3 - Padrão MVC

Model	A camada *Model* é responsável por armazenar as regras de negócio, as entidades e as classes que acessam o banco de dados.
View	A camada *View* é encarregada de exibir as páginas e outros tipos de resultados para o usuário ou para outros sistemas que se comunicam. Esta camada gera a resposta que o *framework* envia ao navegador, geralmente no formato *HTML*.
Controller	O *Controller* é responsável por receber as requisições *web* e determinar como tratá-las. Nesta camada, decidimos quais modelos serão executados para cada ação e para qual *view* a resposta será encaminhada. Em resumo, o *controller* faz a ligação entre todas as outras camadas.

Fonte: Elaborado a partir de Turini (2015).

2.6 BANCO DE DADOS

De acordo com Alves (2014, p. 17) "Um banco de dados é um conjunto de dados com um significado implícito", e ainda ressalta:

> Antes de entrar na definição de banco de dados propriamente dita, é preciso esclarecer a diferença entre informação e dado. Informação é qualquer fato ou conhecimento do mundo real e que pode ou não ser registrado/armazenado. Dado é a representação da informação, que pode estar registrado em papel, num quadro de aviso ou no disco rígido do computador. **(**Alves, 2014, p. 17).

Um banco de dados sob o olhar de Alves (2014), é como uma representação organizada e significativa de parte do mundo real, onde as informações são armazenadas de forma lógica e estruturada. Não se trata apenas de uma coleção aleatória de dados, mas sim de informações que têm um propósito específico e relevância para determinado contexto.

> Um banco de dados é um conjunto de valores e arquivos que se relacionam entre si e demonstram um cadastro de pessoas, vendas, produtos, agendas, etc. Por exemplo, uma planilha ou uma ficha cadastral podem ser exemplos de cadastros. O banco de dados guarda todos esses cadastros em formato virtual e os disponibiliza para as aplicações consultarem e emitirem relatórios, realizarem vendas, etc. (Silva et al., 2021, p.36).

Silva et al., (2021) destaca que em um Banco de Dados existem outras características importantes além da velocidade, sendo eles atomicidade, isolamento, consistência e durabilidade.

Compreendendo isso, torna-se essencial discernir o significado e a importância de cada um desses pilares para os bancos de dados.

As características mencionadas podem ser observadas no Quadro 4.

Quadro 4 - Características de um Banco de Dados

Característica	Descrição
Atomicidade	A atomicidade assegura que uma transação seja completamente concluída com sucesso e os seus efeitos sejam confirmados (*commit*), ou então, em caso de falha ou interrupção, que todas as suas operações sejam desfeitas (*rollback*), restaurando o banco de dados ao estado anterior à transação. É essencial que um sistema de gerenciamento de banco de dados não permita transações incompletas, pois isso comprometeria a integridade e a consistência dos dados armazenados. Atomicidade é a regra do "ou tudo ou nada". Ou toda a transação é salva com sucesso no banco ou toda a transação é descartada
Isolamento	Essa característica garante a independência de cada transação no sistema de gerenciamento de banco de dados (SGBD). Cada transação é tratada como única e independente, evitando conflitos quando várias transações alteram os mesmos valores. Imagine um *e-commerce* durante a temporada de Natal, onde apenas um item de Papai Noel está em estoque e dois clientes tentam comprá-lo simultaneamente. O princípio de isolamento garante que o SGBD processe a primeira transação com sucesso, enquanto a segunda recebe uma notificação de que o item está esgotado. Assim, o sistema resolve conflitos entre transações independentes sem intervenção da equipe de desenvolvimento.
Consistência	Consistência no contexto de bancos de dados refere-se à necessidade de que todas as regras sejam seguidas para garantir que o sistema funcione conforme suas características fundamentais. Isso implica que o tipo de valor inserido corresponda ao tipo do atributo (VARCHAR, INT, DATE, etc.). Por exemplo, em termos de consistência, um campo do tipo Inteiro não pode aceitar caracteres alfabéticos. Essa distinção é essencial para otimizar buscas e resultados em campos numéricos, já que os computadores processam principalmente números. Dados não numéricos requerem mais processamento, pois precisam ser convertidos em linguagem compreensível pela máquina. Isolamento é a garantia de que as transações não terão disputa para edição ou escrita em um determinado dado ou valor.
Durabilidade	Na perspectiva da durabilidade em transações de banco de dados, uma vez que uma transação é finalizada, seus dados tornam-se imutáveis. Somente outra transação pode alterá-los. Essa característica garante a integridade dos dados. Por exemplo, ao concluir uma compra com sucesso e, posteriormente, não conseguir encontrar o histórico dessa transação, seria bastante frustrante, neste caso a durabilidade assegura que somente outra transação poderá modificar os dados da sua compra, permitindo rastrear as ações realizadas. O sistema de gerenciamento de banco de dados não realiza modificações após a conclusão de todas as transações pendentes, portanto, qualquer alteração nos dados será originada pelo usuário ou pelo sistema. Durabilidade é a proteção dos dados contra qualquer coisa que não seja uma nova transação do banco de dados.

Fonte: Elaborado a partir de Silva et al., (2021).

2.6.1 Sistema Gerenciador de Banco de Dados (SGBD)

De acordo com Heuser (2011), os Sistemas de Gerenciamento de Banco de Dados (*SGBD*) foram desenvolvidos na década de 1970 com o propósito de simplificar a programação de aplicações de banco de dados.

> Durante muito tempo, criou-se a ideia de que projetar bancos de dados era uma disciplina com identidade própria, uma atividade específica e, até certo ponto, isolada no ciclo de vida de um sistema, que tinha existência própria e podia ser realizada a partir de primitivas e conceitos exclusivos da técnica de modelagem de dados. (Machado, 2020, p.15).

Um Sistema de Gerenciamento de Banco de Dados (SGBD) conforme Silva et al., (2021) deve suportar diversas atividades relacionadas ao gerenciamento de dados, como consulta e armazenamento, além de fornecer interfaces para o seu ambiente. Para alcançar esses objetivos, um *SGBD* é composto por vários módulos de interação que, juntos, formam sua arquitetura. A organização de um *SGBD* pode seguir dois tipos de arquitetura: banco de dados centralizados e banco de dados distribuídos.

2.6.2 PostgreSQL

IBM (2024) diz que o PostgreSQL, muitas vezes pronunciado como "Post-GRES", é um banco de dados de código aberto conhecido por sua confiabilidade, flexibilidade e suporte a padrões técnicos abertos. Desenvolvido como um acompanhamento do INGRES na década de 1980, foi concebido por Michael Stonebraker, professor de ciência da computação em Berkeley. Inicialmente chamado de POSTGRES, o projeto adicionou suporte para *SQL* (*Structured Query Language*) em 1994 e evoluiu para o PostgreSQL. Diferentemente de outros sistemas de gerenciamento de banco de dados relacionais, o PostgreSQL suporta tanto tipos de dados relacionais quanto não relacionais, tornando-o altamente compatível e maduro.

Conforme a documentação do PostgreSQL (2024a), ele oferece diversas funcionalidades importantes para o desenvolvimento e modelagem de um banco de dados, as quais são essenciais para o desenvolvimento eficiente e eficaz de aplicações.

As funcionalidades gerais do PostgreSQL podem ser observadas no Quadro 5.

Quadro 5 - Funcionalidades Gerais PostgreSQL

Funcionalidade	Descrição
Chaves Estrangeiras	Chaves Estrangeiras Campo ou conjunto de campos em uma tabela que associa a uma instância de outra tabela, criando um relacionamento entre as tabelas. Caracteriza-se de uma coluna que faz referência à chave primária de outra tabela.
Funções de agregação	Calculam um único valor de resultado de um conjunto de valores de entrada. As funções de agregação internas são: *avg; bit_and; bit_or; bool_and; bool_or; count; every; max; min e sum*;
Funções e Operadores	O PostgreSQL possui uma variedade ampla de funções e operadores prontos para os mais variados tipos de dados. Também é possível definir funções e operadores próprios para cada tipo de uso. Alguns dos tipos de funções e operadores do PostgreSQL são: lógicos; de comparação; matemáticos; de texto; de texto binário; de texto em *bit*; de data; geométricos; de e-mail e de vetores;
Integridade Transacional	Corresponde as quatro propriedades básicas de integridade transacional, conhecidas como *ACID*: Atomicidade, Consistência, Isolamento e Durabilidade.
Linguagens Procedurais	Trata-se da possibilidade que o PostgreSQL dá ao usuário de definir funções que sejam escritas em outras linguagens além do *SQL* e *C*. Essas outras linguagens são conhecidas como linguagens procedurais. O banco de dados solicita a instrução para o manipulador capacitado que analisará e executará o algoritmo retornando os resultados desejados.
Métodos de Índice	São índices que permitem que a tabela seja indexada, facilitando e acelerando o processo de consulta em um banco de dados, evitando a necessidade de varrer toda a tabela para se obter um único valor.
Queries	Processo de recuperação ou o comando para recuperar dados de um banco de dados. Em *SQL*, o comando *SELECT* é usado para especificar essas consultas.
Tipos de dados	Suporta os tipos de dados SQL padrão *int, smallint, real, double precision, char (N), varchar (N), date, time, timestamp e interval*, assim como outros tipos de utilidade geral, e um conjunto diversificado de tipo geométricos.
Triggers	É uma especificação de que o banco de dados deve executar automaticamente uma função específica sempre que um determinado tipo de operação for executado. Os *triggers* podem ser definidos para serem executados antes ou depois de qualquer operação *INSERT*, *UPDATE* ou *DELETE*, uma vez por linha modificada ou uma vez por instrução SQL. Se um evento trigger ocorrer, a função do trigger será chamada no momento apropriado para lidar com o evento.
Views	São pseudo-tabelas geradas a partir de uma instrução *SQL* que representam um subconjunto de dados presentes em tabelas reais.

Fonte: Elaborado a partir de PostgreSQL (2024b).

2.7 FRAMEWORKS E BIBLIOTECAS

Segundo Oliveira e Zanetti (2020, p.101) “A adoção de *frameworks* facilita o desenvolvimento de aplicações, pois implementa as funções mais comumente utilizadas em determinada parte do projeto.”.

2.7.1 Bootstrap

De acordo com Zabot e Matos (2020, p. 101) afirmam que o Bootstrap é um *framework front-end* de código aberto para desenvolvimento rápido com tecnologias *web*. Inclui modelos

de design com base em *HTML* e *CSS*, formulários, botões, tabelas, barras de navegação e muitos outros elementos.

Conforme Zabot e Matos (2020, p. 101), "É fácil de utilizar e, para começar, basta ter conhecimentos básicos sobre *HTML* e *CSS*. O Bootstrap disponibiliza recursos responsivos que se ajustam a telefones, *tablets* e *desktops* na abordagem *mobile-first*.

Zabot e Matos (2020, p. 101)**,** complementam afirmando que o Bootstrap utiliza uma versão reduzida do *jQuery* e que fornece elementos predefinidos em *CSS* e *plug-ins JavaScript*.

2.7.2 jQuery

O *jQuery* é considerado como uma biblioteca dominante do *JavaScript*. Foi escrita em 2005 por John Resig, sendo utilizado por mais de dois terços de todos os sites *web*. É uma biblioteca gratuita e de código aberto, emprega uma sintaxe que facilita a utilização do *CSS*, JavaScript e o *DOM (Document Object Model)*. (Robbins, 2018).

De acordo com Flanagan (2014), a utilização do *jQuery* facilita o encontro dos elementos de um documento, tornando-se fácil a manipulação e a adição de conteúdo. Sendo assim, é possível editar os atributos *HTML* e propriedades *CSS*, definindo métodos e tratamentos de evento, até mesmo animações. Com o *jQuery* é possível realizar requisições *HTTP (Hypertext Transfer Protocol Secure)* através do *Ajax (Asynchronous JavaScript and XML)*.

O jQuery se concentra em consultas (*query*, em inglês). Uma consulta comum seria a utilização de um seletor *CSS* para identificar um determinado elemento dentro do documento *HTML*, retornando um objeto representando os elementos. Com o retorno do objeto pode ser utilizado muitos métodos úteis para operar como um grupo nos elementos condizentes. (Flanagan, 2014).

2.7.3 SweetAlert

O SweetAlert é considerado uma biblioteca *JavaScript* que tem como objetivo facilitar a criação de alertas em aplicações web. Todo o código-fonte está aberto, sendo disponibilizado no GitHub, sendo assim qualquer programador possui acesso. Um dos grandes benefícios é que a biblioteca é compatível com todos os navegadores atuais. Desta forma, essa biblioteca permite a criação de alertas mais agradáveis, responsivos e customizados. (Pinho, 2018).

2.8 JSON

JSON (*JavaScript Object Notation*), segundo Deitel, Deitel e Wald (2016) é um conhecido formato de troca de dados baseados em texto utilizado para representar objetos como pares chave-valor de dados.

Conforme Freitas et al., (2021), *JSON* é um subconjunto da linguagem *JavaScript* que, embora utilize a mesma sintaxe, não se limita apenas ao *JavaScript*. Ele foi criado para facilitar o intercâmbio de dados, apesar de não ser pioneiro nessa área, já que o *XML* (Extensible Markup Language) já existia. Mesmo surgindo após o *XML*, o *JSON* é mais leve e simples. O Quadro 6 sugere algumas diferenças notáveis entre *JSON* e *XML*.

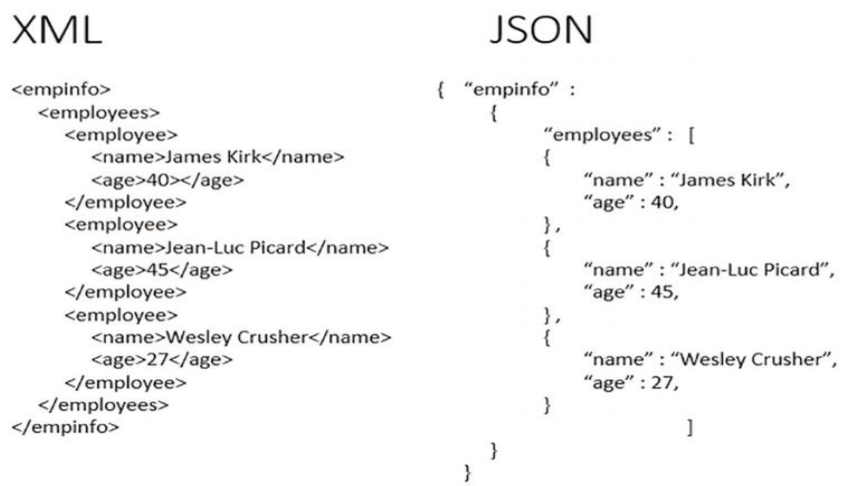
Quadro 6 - Diferenças básicas entre JSON e XML

XML	JSON
Extensible Markup Language.	*JavaScript Object Notation.*
Sintaxe similar à linguagem HTML.	Sintaxe similar à linguagem *JavaScript.*
Difícil de escrever o código.	Fácil de escrever o código.
Não consegue armazenar dados em variáveis.	Capacidade de armazenar até seis tipos de dados em variáveis.
Orientada a documento.	Orientada a Objeto.
O dado tem maior segurança.	O dado tem menor segurança.
Arquivos menos legíveis.	Arquivos mais legíveis.
Exibição simplificada para o navegador *Web* descrever.	Não pode ser exibido no navegador sem análise.

Fonte: Elaborado a partir de Freitas et al. (2021).

Na figura 3 é realizada outra comparação entre a linguagem *XML* e em formato *JSON*, porém em linhas de código fonte.

Figura 3 - Comparação de Código XML e em JSON



Fonte: Freitas et al. (2021, p.17)

2.9 API

Freitas et al. (2021, p.43) definem API (*Application Programming Interface*, ou em português, Interface de Programação de Aplicação) como, "[...] construções de aplicações que permitem que os desenvolvedores criem funcionalidades complexas mais facilmente. Tais construções abstraem o código mais complexo, proporcionando o uso de sintaxes de forma mais simples.".

Segundo Rodrigues et al. (2020), uma *API* pode ser definida como um conjunto de procedimentos e diretrizes de um sistema que permite acesso externo às suas funcionalidades de maneira abstrata, sem a necessidade de interação direta com o código ou a implementação subjacente, mantendo a independência em relação à programação específica.

Conforme o MDN (2024), uma API geralmente é composta por um conjunto padronizado de métodos, propriedades, eventos e *URLs (Uniform Resource Locator)* que facilitam a interação entre a aplicação, o servidor e serviços de terceiros no desenvolvimento de softwares.

2.9.1 FIPE

A Fundação Instituto de Pesquisas Econômicas (FIPE) é a instituição responsável pela manutenção da Tabela FIPE, conhecida pela disponibilidade dos preços médios de veículos em âmbito nacional. Estas informações podem ser integradas com aplicações terceira por meio da FIPE API.

A FIPE API (2024) afirma que "A API permite acesso aos dados atualizados sobre os preços médios de veículos no mercado nacional, possibilitando consultas de valores para carros, motos e caminhões com base nas tabelas da Fundação Instituto de Pesquisas Econômicas".

De acordo com a FIPE API (2024), "É possível consultar os preços médios de veículos, filtrando por tipo de veículo, marca, modelo e ano, facilitando a integração de informações de mercado para diversos sistemas e aplicações".

2.9.2 IBGE

O Instituto Brasileiro de Geografia e Estatística (IBGE) é a instituição responsável pela coleta e análise de dados geográficos e estatísticos do Brasil, oferecendo uma ampla gama de

informações sobre as divisões territoriais e administrativas do país. Essas informações podem ser integradas em aplicações de terceiros por meio da API de Localidades do IBGE.

A API de Localidades do IBGE (2024) afirma permite acesso a dados atualizados sobre estados, municípios, regiões e outras divisões administrativas do Brasil, possibilitando consultas de informações geográficas para diversas aplicações.

De acordo com a API de Localidades do IBGE (2024), é possível consultar dados geográficos e administrativos, filtrando por unidade federativa, município, mesorregião e microrregião, facilitando a integração de informações territoriais e populacionais para diferentes sistemas e plataformas.

2. METODOLOGIA DA PESQUISA

O presente trabalho de conclusão de curso é caracterizado por uma pesquisa aplicada e descritiva, cujo objetivo é investigar e documentar os processos envolvidos no desenvolvimento de uma aplicação web para a loja de veículos “Sálvio Automóveis” da cidade de Imbuia, Santa Catarina. Esta aplicação é projetada com ênfase na gestão eficiente da movimentação de veículos nas lojas. A pesquisa busca resolver questões cruciais como: Como monitorar de maneira eficaz a entrada e saída de veículos? Como melhorar a eficiência operacional para a empresa? Como apresentar aos clientes o estoque de veículos de forma simples e prática?

Para o desenvolvimento da pesquisa, realizou-se uma revisão da literatura existente, explorando tecnologias necessárias para a construção do protótipo. Esta investigação preliminar não apenas esclareceu os conceitos fundamentais, mas também orientou a escolha das tecnologias mais adequadas e consolidadas no mercado. O intuito é desenvolver o protótipo de maneira que reduza ao máximo a necessidade de manutenções frequentes e que seja eficiente ao cliente.

Para embasar o desenvolvimento desta pesquisa, foi realizada uma revisão da literatura existente, explorando as tecnologias essenciais para a criação do protótipo. Essa revisão inicial ajudou a esclarecer os conceitos fundamentais de gestão de veículos e direcionou a escolha de algumas das tecnologias mais adequadas e amplamente utilizadas no mercado, como *PHP*, PostgreSQL, *HTML5*, *CSS3* e *JavaScript*.

Além disso, foi realizada uma análise de ferramentas semelhantes ao protótipo proposto para a construção do estado da arte. Essa pesquisa comparativa forneceu informações sobre como outros sistemas lidam com problemas semelhantes e ajudou a identificar oportunidades que o protótipo MCC pode aproveitar.

3.1 ESTADO DA ARTE

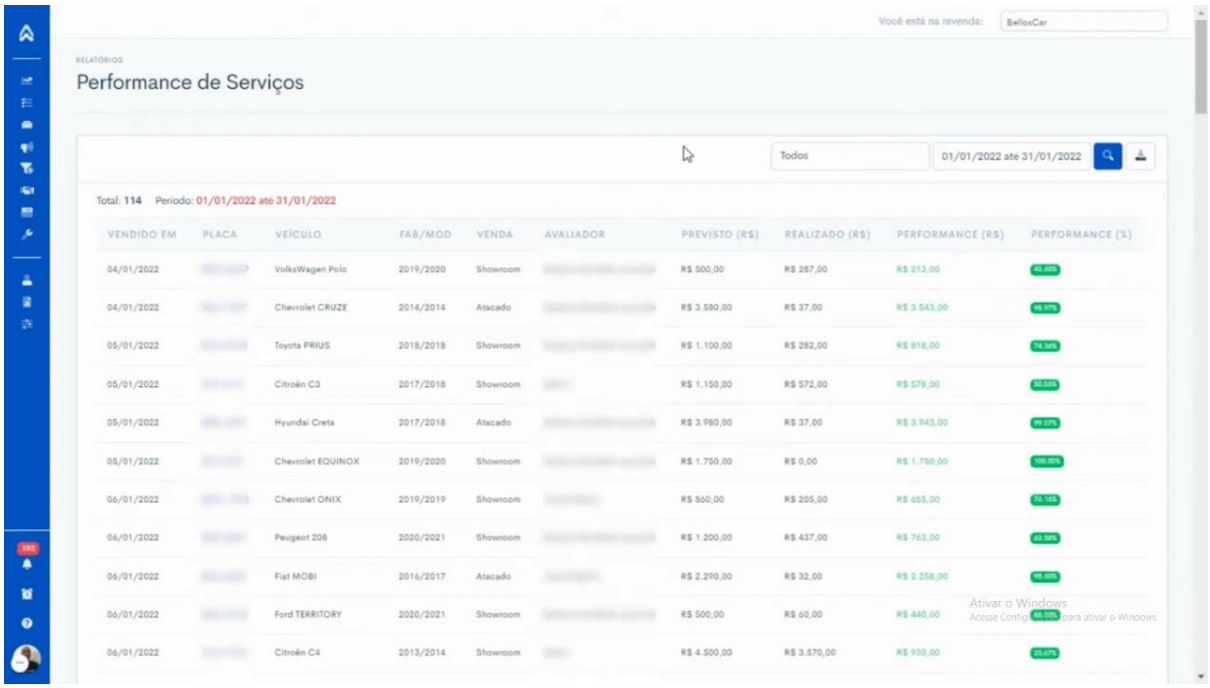
Nesta seção, são apresentadas quatro ferramentas já existentes no mercado que possuem objetivos semelhantes aos deste projeto. Os objetivos dessas ferramentas são detalhados com o intuito de ilustrar algumas das aplicações já existentes.

3.1.1 Autoconf

O *software* Autoconf, conforme apresentado na Figura 4, é uma ferramenta voltada para a gestão de lojas de veículos. Seu objetivo é otimizar a organização e a eficiência operacional. O sistema permite gerenciar processos de compra e venda de veículos, oferecendo funcionalidades como cadastro de veículos, dados pessoais, emissão de notas fiscais eletrônicas (NF-e) e publicação de anúncios em portais de venda de veículos. (Autoconf, 2024).

Entre suas funcionalidades está o sistema de Avaliação 360º, que fornece uma análise detalhada dos veículos, além de recursos para geração de contratos e integração com portais de anúncios. (Autoconf, 2024).

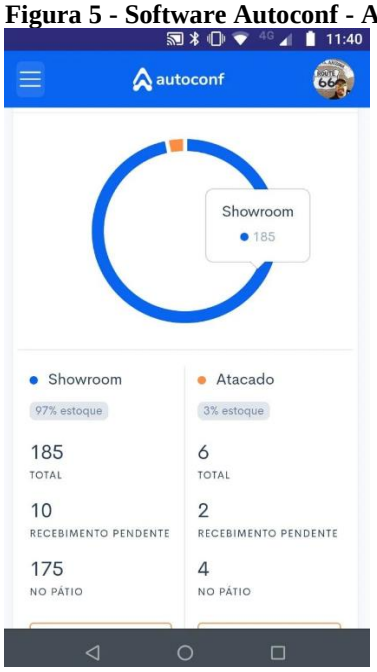
Figura 4 - Software Autoconf - WEB



Fonte: Autoconf (2024).

A aplicação conta com diferentes planos de contratação, incluindo *Site*, *Basic*, *Pro* e *Premium*, e está disponível tanto na versão web quanto em plataformas móveis, como *iOS* e *Android*. (Autoconf, 2024).

Figura 5 - Software Autoconf - Android

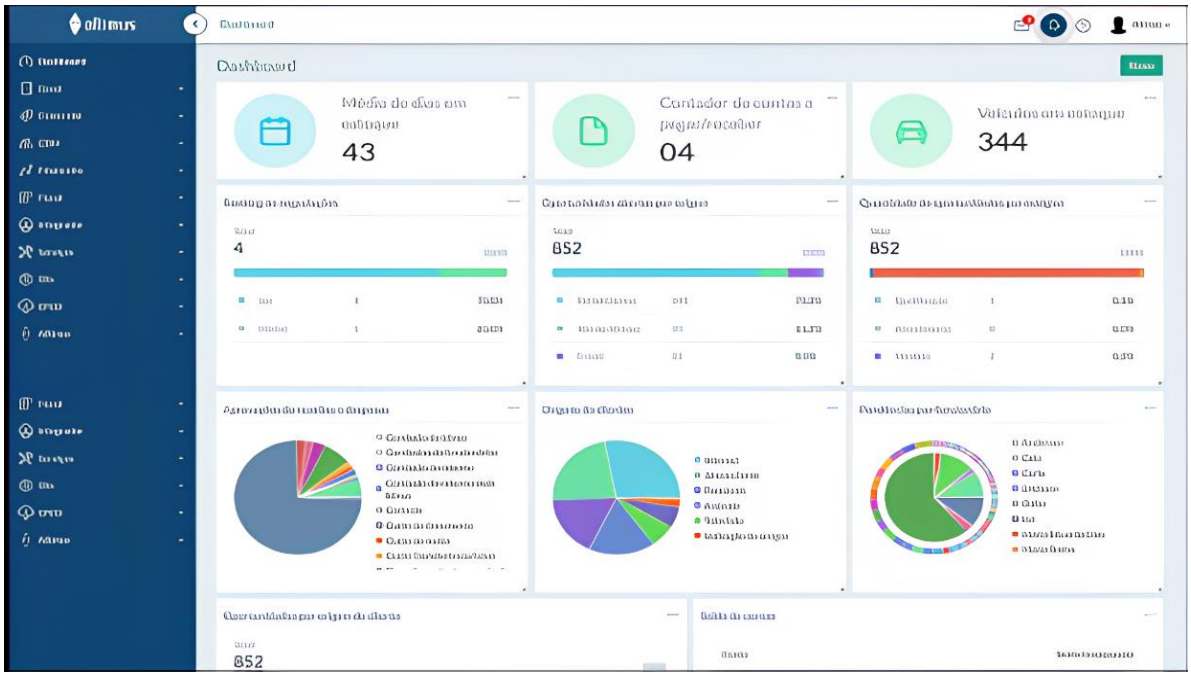


Fonte: Autoconf (2024).

3.1.2 Altimus

O software Altimus, conforme demonstrado na Figura 6, é uma plataforma web voltada para lojas de veículos, oferecendo soluções para a gestão comercial e de operações. Utilizado por diversas lojas do setor automotivo, o Altimus se destaca por integrar diversas funcionalidades, como gestão comercial, financeira e fiscal, e publicação de anúncios em mais de 50 portais, além de fornecer um CRM (*Customer Relationship Management*, ou Gestão de Relacionamento com o Cliente) para gerenciar oportunidades de vendas. (Altimus, 2024).

Figura 6 - Software Altimus WEB - Dashboard



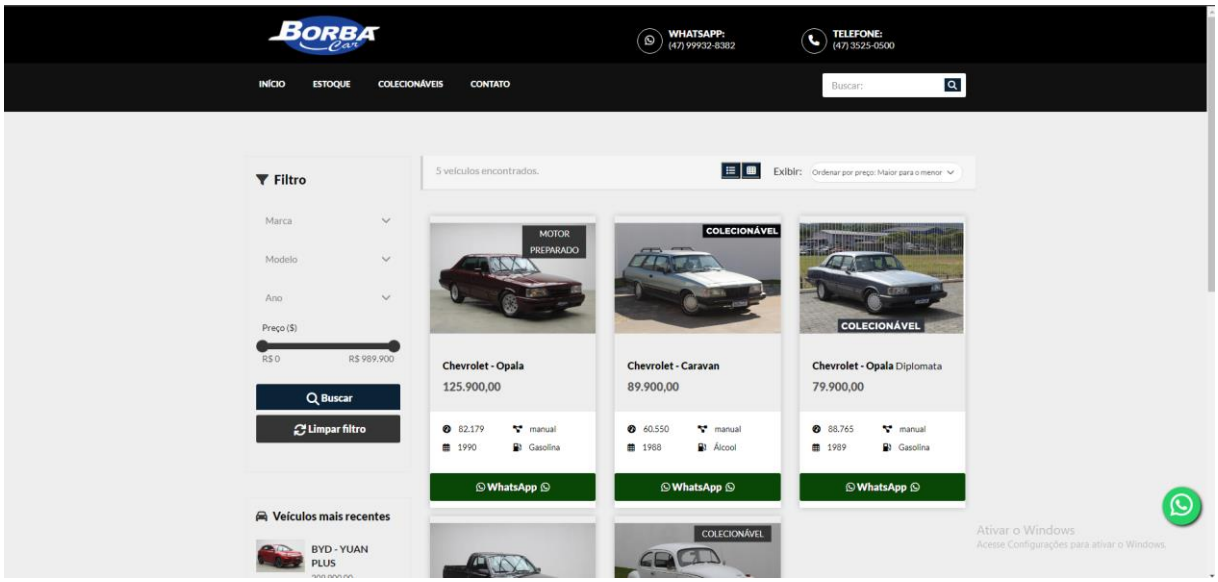
Fonte: Altimus (2024).

A plataforma também permite a criação de sites personalizados para revendas de veículos e oferece mobilidade, sendo acessível a partir de diferentes dispositivos, como computadores, *tablets* e *smartphones*, com interface responsiva adaptável. (Altimus, 2024).

3.1.3 Borba Car

O site Borba Car é uma de loja de veículos localizada na cidade de Rio do Sul - SC, que oferece veículos para diferentes perfis de clientes. Sua plataforma (conforme Figura 7) inclui uma seção dedicada à venda de veículos colecionáveis, que apresenta modelos antigos em bom estado de conservação, destacados pela sua exclusividade. Esses veículos têm preços que, por vezes, não seguem a tabela FIPE (*Fundação Instituto de Pesquisas Econômicas*) devido ao seu caráter único (Borba Car, 2024).

Figura 7 - Site Borba Car WEB



Fonte: Borba Car (2024).

O site (conforme a Figura 8) é responsivo, otimizado tanto para *desktops* quanto para dispositivos móveis, proporcionando uma experiência de navegação consistente para seus usuários (Borba Car, 2024).

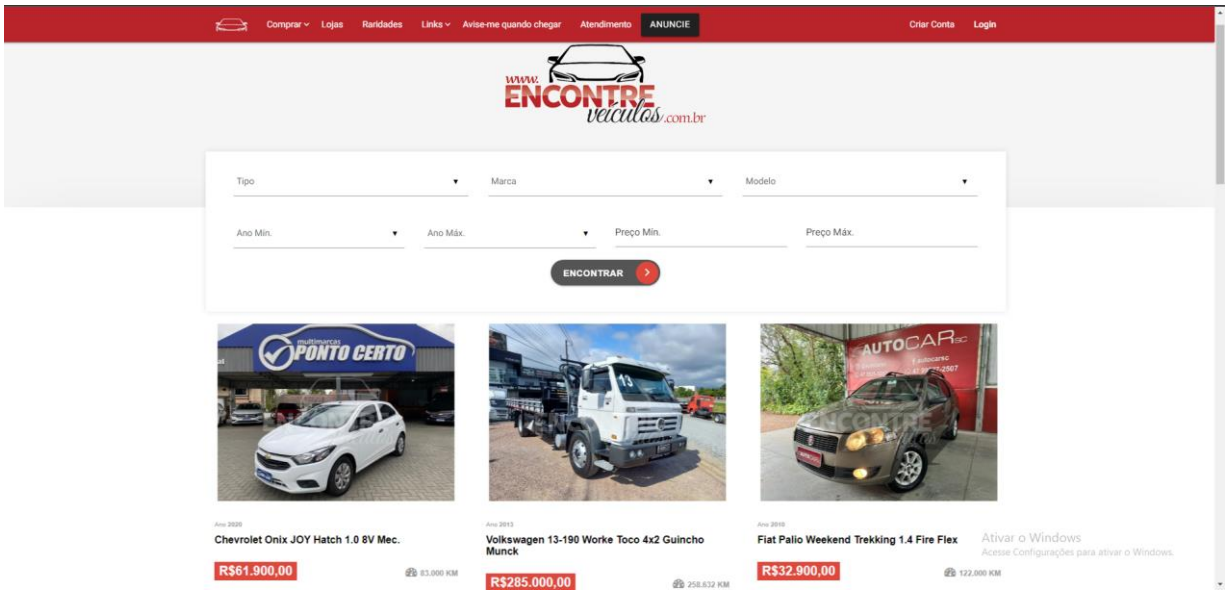
Figura 8 - Site Borba Car - IOS

Fonte: Borba Car (2024).

3.1.4 Encontre Veículos

O Encontre Veículos, demonstrado na Figura 9, é um portal *online* que atua como plataforma de anúncios para o mercado de veículos. Voltado para a exposição e busca de veículos, o site oferece ferramentas para lojistas e indivíduos anunciarem seus veículos, com funcionalidades como descrição detalhada dos modelos e upload de fotos (Encontre Veículos, 2024).

Figura 9 - Site Encontre Veículos WEB



Fonte: Encontre Veículos (2024).

Diferentemente dos softwares de gestão de lojas, o foco do Encontre Veículos é focado na visibilidade e alcance dos anúncios, servindo como um canal de conexão entre vendedores e potenciais compradores de maneira eficiente e acessível (Encontre Veículos, 2024).

Embora o Encontre Veículos não ofereça um sistema de gestão comercial ou *CRM*, a plataforma se destaca pela simplicidade e acessibilidade no processo de compra e venda, atendendo a uma grande variedade de usuários, tanto pessoas físicas quanto jurídicas (Encontre Veículos, 2024).

3. MCC: PROTÓTIPO DE SOFTWARE WEB PARA GERENCIAR LOJAS DE VEÍCULOS

Neste capítulo são apresentados todos os aspectos técnicos do processo de análise e implementação do protótipo.

4.1 ANÁLISE

Durante o capítulo de revisão da literatura, foram discutidos temas pertinentes à engenharia de *software*, que se mostraram indispensáveis para esta etapa, pois proporcionaram o entendimento necessário sobre as demandas do negócio do cliente e os objetivos da aplicação. Assim, a análise é essencial para esclarecer o escopo do que será desenvolvido. Além disso, uma análise bem estruturada facilita a manutenção da aplicação, já que os processos utilizados durante o desenvolvimento estão claramente documentados, tornando a resolução de possíveis problemas mais eficientes e prática.

4.1.1 Visão Geral do Protótipo

O protótipo de software web MCC (*Munsfeld Car Control*) foi desenvolvido com o objetivo de aprimorar o gerenciamento de vendas e o controle de veículos em lojas. Em um mercado onde a gestão eficiente dos recursos é essencial, o MCC é projetado para ser adaptado especificamente para cada cliente, suas instalações são individuais com replicações próprias de Banco de Dados, garantindo uma solução otimizada para as necessidades particulares de cada empresa. Isso permite uma administração mais organizada e acessível do estoque de veículos, proporcionando um controle detalhado e atualizado das operações, facilitando a execução de tarefas diárias e melhorando o desempenho geral da empresa.

O principal recurso do MCC é a capacidade de gerenciar o estoque de veículos de forma eficiente e transparente. O software permite o controle detalhado da disponibilidade de veículos, oferecendo funcionalidades como cadastro, atualização, exclusão e baixa de veículos. Além disso, o MCC possibilita a visualização personalizada da lista de veículos disponíveis na loja, com filtros por marca, modelo, ano e preço, adaptados às preferências de cada cliente.

Outro destaque do protótipo é a integração de uma ferramenta para consulta à tabela FIPE, que permite aos usuários verificar o valor de mercado atualizado dos veículos

disponíveis. Na parte administrativa, o MCC oferece recursos para a manutenção de usuários, gerenciamento de histórico de movimentações dos veículos e registro de ocorrências, garantindo um melhor controle.

O software também inclui uma interface intuitiva e compatível com múltiplas plataformas, facilitando o acesso tanto em dispositivos móveis quanto em *desktops*. Em resumo, o MCC se destaca por oferecer uma solução prática para a gestão de lojas de veículos, com um controle assertivo que facilita o desempenho das vendas e alinha as operações com os objetivos financeiros da empresa.

4.1.2 Comparação do Protótipo com o Estado da Arte

Nos Quadros 7 e 8, são apresentadas comparações entre os recursos disponibilizados pelos *softwares* pesquisados, listados no estado da arte, e o protótipo proposto neste trabalho. O Quadro 7 destaca os recursos relacionados ao *website*, enquanto o Quadro 8 apresenta uma comparação dos recursos voltados para o ambiente administrativo. O objetivo dessas comparações é evidenciar quais funcionalidades estão implementadas nos sistemas avaliados.

Quadro 7 - Recursos dos softwares apresentados (Website)

Recurso	Protótipo MCC	Borba Car	Encontre Veículos
Anúncio de veículos	X	X	X
Filtros de busca	X	X	X
Consulta automática por veículo à tabela FIPE	X		
Plataforma responsiva	X	X	X

Fonte: acervo do autor (2024).

Quadro 8 - Recursos dos softwares apresentados (Ambiente Administrativo)

Recurso	Protótipo MCC	Autoconf	Altimus
Gerenciamento de veículos	X	X	X
Consulta automática por veículo à tabela FIPE	X		
Registros de ocorrências com o veículo	X		
Emissão de notas fiscais		X	
CRM para gerenciamento		X	X
Acesso multiplataforma	X	X	X

Fonte: acervo do autor (2024).

4.1.3 Requisitos

Esta seção lista os requisitos essenciais para o funcionamento da aplicação. Esses requisitos devem ser implementados para atender às necessidades do cliente e garantir o

cumprimento das propostas estabelecidas. O Quadro 9 detalha os requisitos funcionais necessários para o desenvolvimento do protótipo por parte do *website*, e visando facilitar o detalhamento dos mesmos, as regras de negócio são relacionadas a seguir.

Quadro 9 - Requisitos Funcionais (Website)

Número	Nome	Descrição
RF1	**Página Inicial**	O protótipo deverá conter uma página inicial de apresentação do site.
	Regras de Negócio	
	RN1	Para a visualização dos veículos deve ser exibido uma lista com os veículos disponíveis na loja e cada veículo deve ter as principais informações apresentadas: Marca, modelo, ano, quilometragem, valor, câmbio e combustível. Deverá exibir também informações corporativas, como a história da empresa, missão, visão, valores e um banner de imagem.
RF2	**Sobre**	O protótipo deverá conter a página "Sobre" com informações sobre a empresa.
	Regras de Negócio	
	RN1	O protótipo deverá exibir informações corporativas, como a história da empresa, missão, visão e valores.
RF3	**Veículos**	O protótipo deverá exibir a listagem dos veículos disponíveis na loja, com suas respectivas informações.
	Regras de Negócio	
	RN1	Para a visualização dos veículos deve ser exibido uma lista com os veículos disponíveis na loja e cada veículo deve ter as principais informações apresentadas: Marca, modelo, ano, quilometragem, valor, câmbio e combustível.
	RN2	Na listagem dos veículos, deverá ser disponibilizado filtros de veículos por marca, modelo, ano e faixa de preço.
RF4	**Contato**	O protótipo deverá exibir a página de contato com informações da empresa e localização.
	Regras de Negócio	
	RN1	Na rotina de contato o formulário deve conter determinados campos, tais como: Nome (obrigatório), telefone (obrigatório), e-mail (obrigatório), estado (obrigatório), cidade (obrigatório), assunto (obrigatório) e mensagem.
	RN2	Para preenchimento de cidades e estados deve ser utilizada a API disponibilizada pelo IBGE.
	RN3	A rotina de contato deverá conter um pequeno mapa com a localização da empresa, telefone para contato e e-mail da empresa.
RF5	**Política de Privacidade**	O protótipo deverá conter uma rotina para exibir a política de privacidade da empresa.
	Regras de Negócio	
	RN1	A empresa deve implementar e exibir uma política de privacidade, garantindo a proteção dos dados dos usuários.
RF6	**Consulta à tabela FIPE**	O protótipo deverá permitir a consulta à tabela FIPE dos veículos disponíveis na loja.
RF7	**Menu Principal**	O protótipo deverá conter um menu que tenha um redirecionamento para todas as páginas.
RF8	**Detalhamento do Veículo**	O protótipo deverá conter uma rotina para exibir informações específicas do veículo.
	Regras de Negócio	
	RN1	O detalhamento do veículo deve conter informações específicas do veículo, tais como: imagem, marca, ano, modelo, cor, placa (caractere inicial e final), quilometragem, valor, câmbio, combustível, observações adicionais e acessórios.

	RN2	Para a visualização dos veículos deve ser exibido uma lista com os veículos disponíveis na loja e cada veículo deve ter as principais informações apresentadas: marca, modelo, ano, quilometragem, valor, câmbio e combustível.
RF9	**Integração com WhatsApp**	O protótipo deverá disponibilizar um link para o WhatsApp da empresa disponível em todas as páginas.

Fonte: acervo do autor (2024).

O Quadro 10 descreve os requisitos funcionais para o ambiente administrativo do protótipo, focando na gestão interna. As regras de negócio associadas a esses requisitos definem como essas operações devem ser realizadas para garantir a integridade e a eficiência do sistema.

Quadro 10 - Requisitos Funcionais (Ambiente Administrativo)

Número	**Nome**	**Descrição**
RF10	**Login**	O protótipo deverá permitir que o lojista realize o login para acesso.
	Regras de Negócio	
	RN1	O protótipo deverá controlar o acesso dos usuários através de login, garantindo segurança, sendo necessário informar usuário e senha.
	RN2	A senha utilizada pelo usuário do sistema deve conter no mínimo 8 caracteres e incluir pelo menos uma letra maiúscula, uma letra minúscula, um número e um caractere especial.
RF11	**Manutenção de Veículos**	O protótipo deverá permitir a manutenção de veículos, incluindo cadastro, atualização, exclusão e baixa dos veículos.
	Regras de Negócio	
	RN1	Para manutenção do veículo o mesmo deve estar cadastrado no protótipo e conter determinados campos: imagem (obrigatório), marca (obrigatório), ano (obrigatório), modelo (obrigatório), cor (obrigatório), tipo (obrigatório), placa (obrigatório), quilometragem (obrigatório), valor (obrigatório), câmbio (obrigatório), combustível, observações adicionais, acessórios, ativo (obrigatório), único dono (obrigatório), chassis (obrigatório), antigo dono.
	RN2	Para preenchimento da marca de veículo deve ser utilizado a API disponibilizada pela FIPE.
	RN3	Para preenchimento do modelo de veículo deve ser utilizado a API disponibilizada pela FIPE.
RF12	**Manutenção de Usuários**	O protótipo deverá permitir a manutenção de usuários, incluindo cadastro, atualização e exclusão de usuários.
	Regras de Negócio	
	RN1	Para manutenção do usuário o mesmo deve estar cadastrado no protótipo e conter determinados campos: nome (obrigatório), sobrenome, telefone (obrigatório), sexo (obrigatório), e-mail, cidade (obrigatório), tipo (obrigatório), senha (obrigatório para lojista).
RF13	**Histórico de Movimentações**	O protótipo deverá manter histórico de movimentações de entrada e saída dos veículos.
	Regras de Negócio	
	RN1	O protótipo deverá manter um histórico detalhado das movimentações de entrada e saída dos veículos na loja, deve conter os seguintes campos: data/hora (obrigatório), nome veículo (obrigatório), tipo (Usuário/Lojista) (obrigatório).
RF14	**Ocorrências do Veículo**	O protótipo deverá conter uma rotina para registrar e listar eventuais ocorrências com os veículos.
	Regras de Negócio	
	RN1	Eventuais ocorrências com o veículo deverão ser registradas e devem conter os seguintes campos: título (obrigatório), descrição

		(obrigatório), data/hora (obrigatório), cidade (obrigatório), endereço (obrigatório).
RF15	**Página Inicial**	O protótipo deverá contar com uma página inicial, onde terá informações sobre o sistema e acesso às demais telas.
	Regra de Negócio	
	RN1	Para a visualização dos veículos deve ser exibido uma lista com os veículos disponíveis na loja e cada veículo deve ter as principais informações apresentadas: marca, modelo, ano, quilometragem, valor, câmbio e combustível.
RF16	**Menu Principal**	O protótipo deverá conter um menu, trazendo acesso a todas as rotinas.
RF17	**Manutenção de Marca**	O protótipo deverá conter uma rotina que permita a manutenção de marcas, incluindo cadastro, atualização e exclusão.
	Regra de Negócio	
	RN1	Para preenchimento da marca de veículo deve ser utilizado a API disponibilizada pela FIPE.
RF18	**Manutenção de Modelo**	O protótipo deverá conter uma rotina que permita a manutenção do modelo do veículo, incluindo cadastro, atualização e exclusão.
	Regra de Negócio	
	RN1	Para preenchimento do modelo de veículo deve ser utilizado a API disponibilizada pela FIPE.
RF19	**Manutenção Sobre**	O protótipo deverá conter uma rotina que permita a manutenção das informações sobre a empresa, como história, missão, visão e valores.
RF20	**Manutenção de Contato**	O protótipo deverá conter uma rotina que permita a manutenção das informações de contato da empresa, assim como visualizar os dados do formulário de contato.
RF21	**Manutenção de Cor**	O protótipo deverá conter uma rotina que permita a manutenção da cor do veículo, incluindo cadastro, atualização e exclusão

Fonte: acervo do autor (2024).

O Quadro 11 apresenta os requisitos não funcionais do protótipo, esses requisitos asseguram que o sistema não apenas funcione conforme o esperado, mas também ofereça uma experiência de usuário eficiente, segura, e sustentável para futuras manutenções.

Quadro 11 - Requisitos Não Funcionais (WebSite e Ambiente Administrativo)

Número	Nome	Descrição
RNF1	**Performance**	O tempo de resposta para a consulta de veículos no protótipo deverá ser inferior a 15 segundos.
RNF2	**Usabilidade**	O protótipo deverá possuir uma interface intuitiva.
RNF3	**Segurança**	O protótipo deverá implementar criptografia para armazenamento de dados sensíveis.
RNF4	**Compatibilidade**	O protótipo deverá ser compatível com os principais navegadores web (Chrome, Firefox, Safari, Edge e Opera).
RNF5	**Suporte multiplataforma**	O protótipo deverá funcionar em dispositivos móveis e desktops.
RNF6	**Manutenibilidade**	O código-fonte do protótipo deverá ser documentado e seguir boas práticas de desenvolvimento para facilitar a manutenção futura.
RNF7	**Versão do PHP**	O protótipo deverá ser desenvolvido com a versão 7 do *PHP* ou superior.
RNF8	**Registro de login**	O protótipo deverá registrar o login do usuário.
RNF9	**Integração com Banco de Dados**	O protótipo deverá conectar e interagir com o banco de dados para armazenamento e recuperação de dados.
RNF10	**Integridade dos dados**	O protótipo deverá assegurar a integridade e consistência dos dados armazenados.

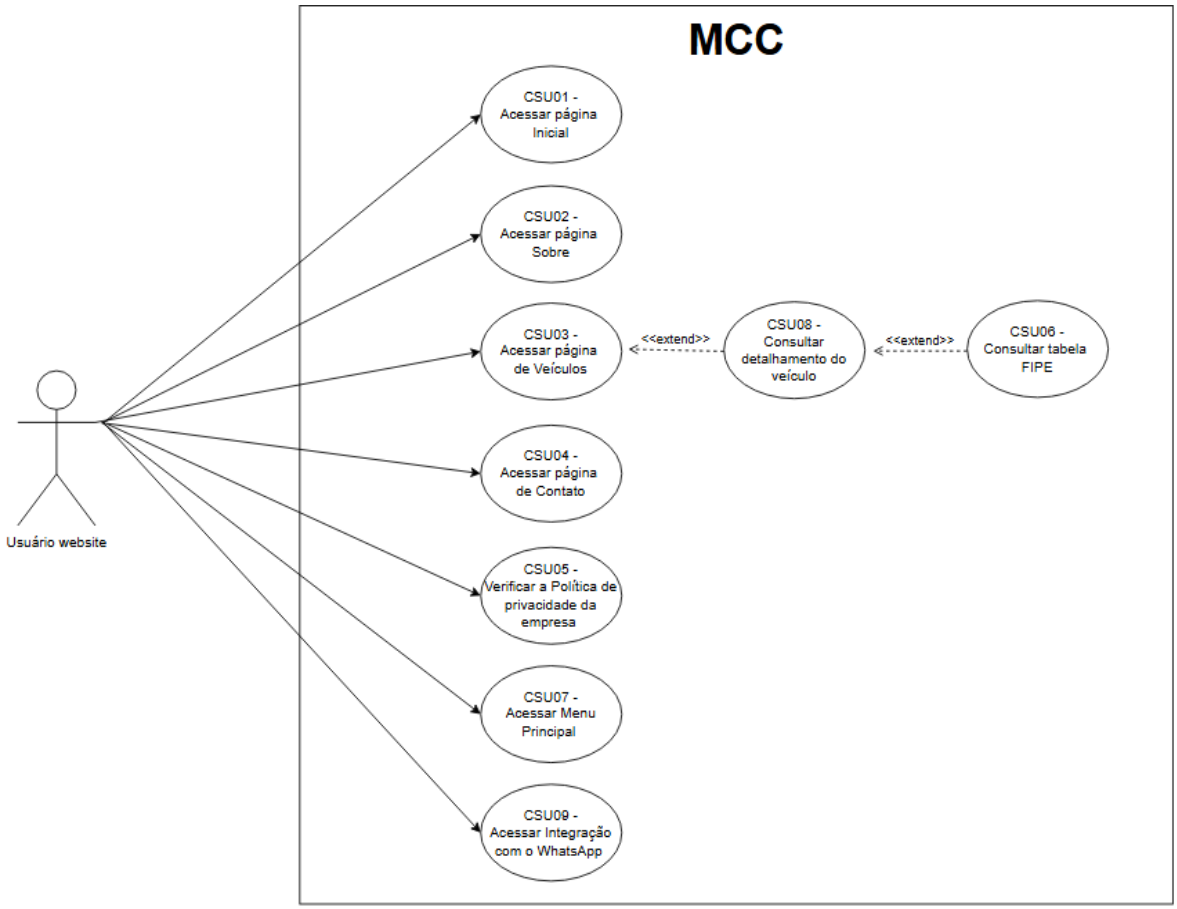
Fonte: acervo do autor (2024).

4.1.4 Diagramas

Para melhorar o entendimento dos requisitos listados e das funcionalidades do protótipo, foram criados diagramas. Esses diagramas ilustram os recursos acessíveis aos usuários, além de detalhar o processo necessário para realizar a atividade principal proposta pelo protótipo.

Na Figura 10, é mostrado um diagrama de caso de uso que apresenta as funcionalidades por parte do *website*, com foco nas funcionalidades disponíveis no *website*.

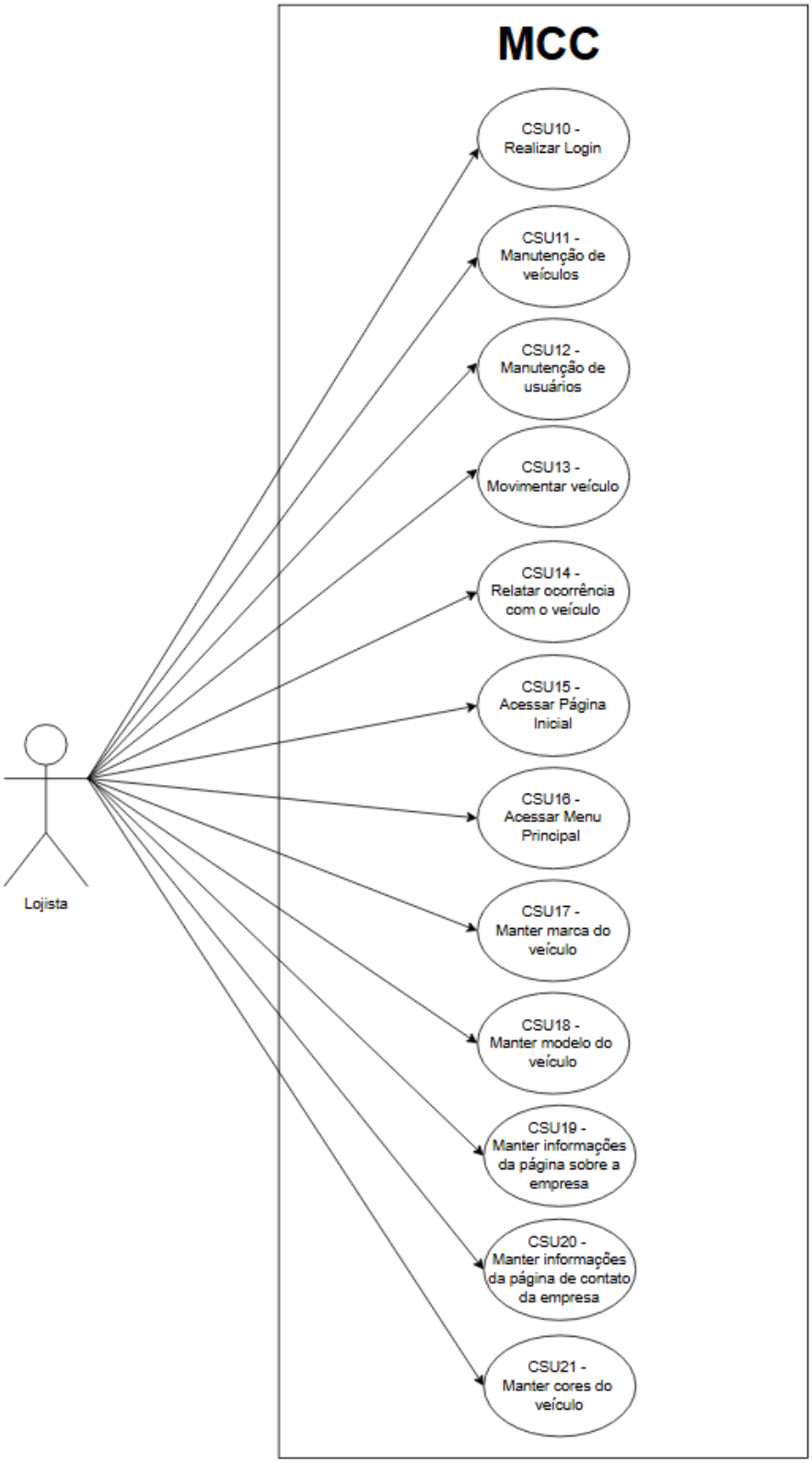
Figura 10 - Diagrama de Caso de Uso (Website)



Fonte: acervo do autor (2024).

Já na Figura 11, é exibido o diagrama de caso de uso que apresenta as funcionalidades do *software web* do ambiente administrativo, juntamente com o ator, que neste cenário é o usuário da empresa.

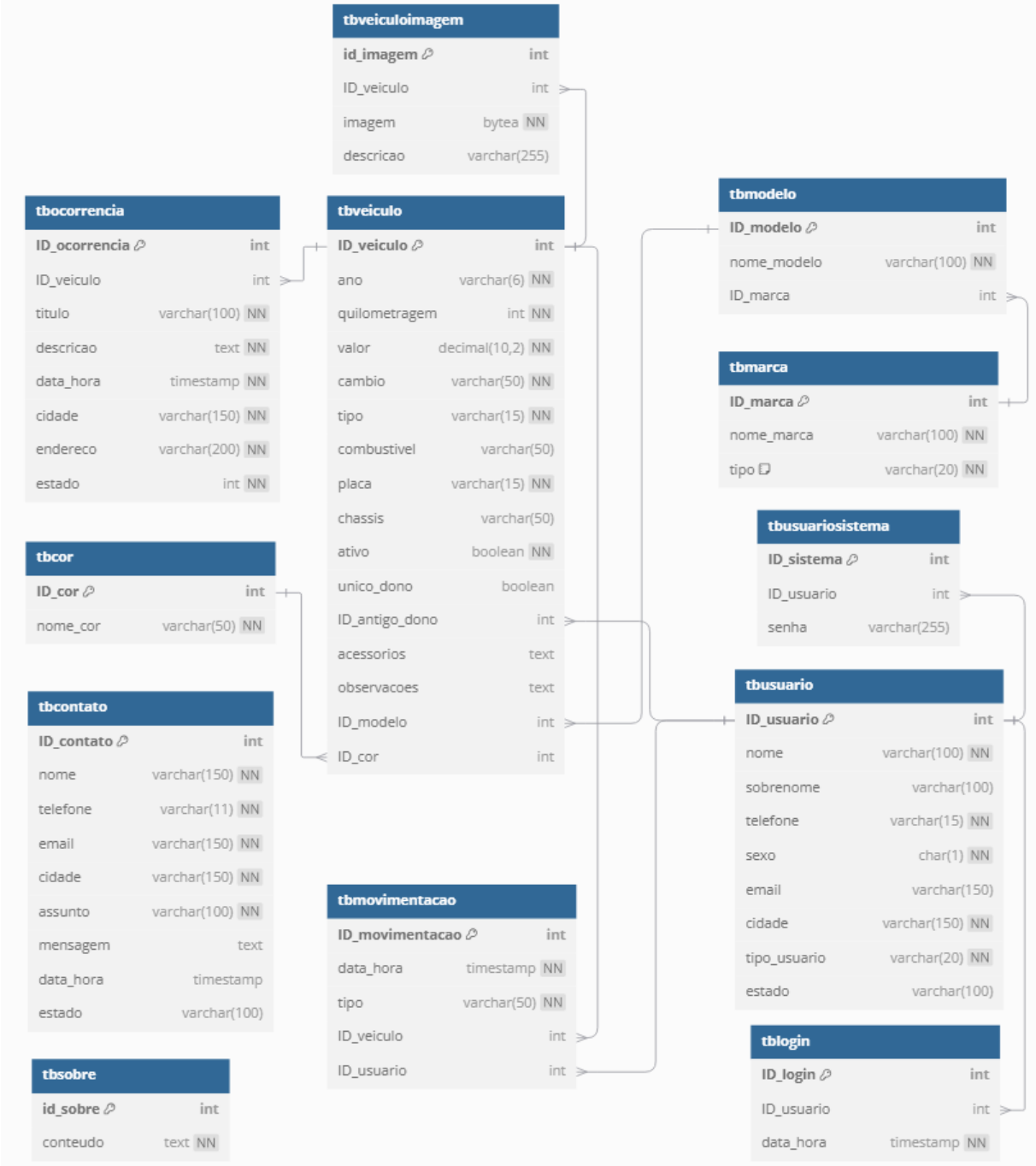
Figura 11 - Diagrama de Caso de Uso (Ambiente Administrativo)



Fonte: acervo do autor (2024).

A seguir é apresentado o diagrama de entidade relacionamento que representa a estrutura do Banco de Dados. A Figura 12 apresenta o diagrama desenvolvido para o protótipo.

Figura 12 - Entidade Relacionamento - Banco de Dados



Fonte: acervo do autor (2024).

Foram criadas 12 tabelas para armazenar os dados de usuários, veículos, ocorrências, movimentações, marca, modelo, cor, usuário do sistema, *login* e contato como os serviços oferecidos pelo protótipo.

Na próxima seção, serão apresentadas as informações referentes à implementação do protótipo, com uma demonstração detalhada dos recursos desenvolvidos e suas funcionalidades.

4.2 IMPLEMENTAÇÃO

Nesta seção serão listadas as ferramentas e tecnologias utilizadas para o desenvolvimento do protótipo, e também, serão apresentadas as rotinas implementadas, atendendo os requisitos funcionais do projeto.

4.2.1 Ferramentas e Técnicas Utilizadas

Durante a fase de análise (detalhada anteriormente na seção 4.1), foram utilizadas ferramentas importantes para estruturar e modelar o protótipo. Uma delas foi o *Lucid.app*, que possibilitou a criação de diagramas de casos de uso. Além disso, foi utilizado o DbDiagram.io para a construção do modelo entidade-relacionamento, o que facilitou o planejamento e a criação do banco de dados.

No que diz respeito ao desenvolvimento das funcionalidades do protótipo, as principais tecnologias web empregadas foram *HTML5*, *CSS3*, *Bootstrap*, *JavaScript* e *PHP*. O *HTML5* foi utilizado para construir a estrutura básica das páginas e dos formulários, sendo uma linguagem de marcação amplamente reconhecida e compatível com a maioria dos navegadores. Sua implementação é relativamente simples, e sua utilização foi essencial para a formatação do conteúdo visual.

Para aprimorar o aspecto visual das páginas, foi utilizado o *CSS3*, que permitiu personalizar os elementos *HTML* e tornar as telas mais amigáveis e atraentes para o usuário. O *Bootstrap*, um *framework* de *CSS*, também foi incorporado ao projeto, proporcionando estilizações pré-construídas, facilitando o desenvolvimento de layouts responsivos que se adaptam a diferentes dispositivos, como *smartphones*, *tablets* e *desktops*.

A manipulação de elementos interativos e a execução de ações no lado do cliente foram implementadas com *JavaScript*. Essa linguagem permitiu a criação de comportamentos dinâmicos, como a habilitação ou desabilitação de campos, dependendo da ação do usuário e das regras de negócio da aplicação. Além disso, a biblioteca *jQuery* foi utilizada para simplificar a manipulação de dados em *JavaScript*. Essa abordagem permitiu que os dados fossem enviados e recebidos do banco de dados sem a necessidade de recarregar a página, melhorando significativamente a experiência do usuário.

O *back-end* do sistema foi construído utilizando a linguagem *PHP* (versão 7.4.2), escolhida por sua robustez e compatibilidade com a arquitetura do projeto. Além disso, a aplicação foi estruturada seguindo o padrão *MVC* (*Model-View-Controller*), que organiza a

separação entre a lógica de apresentação, controle e manipulação de dados. Isso facilitou a manutenção do código e a adição de novas funcionalidades.

O PostgreSQL (versão 16.7) foi o sistema de gerenciamento de banco de dados relacional escolhido para armazenar as informações da aplicação, devido à sua confiabilidade, recursos avançados e à sua natureza *open-source*, o que o torna uma escolha comum entre desenvolvedores de grandes sistemas.

Para o desenvolvimento do código, foi utilizada a *IDE* (I*ntegrated Development Environment* ou Ambiente de Desenvolvimento Integrado) Visual Studio Code, que oferece uma série de extensões que facilitam a organização do projeto e a execução de tarefas como *commits* para o *GitHub*. Essa ferramenta foi fundamental para garantir a organização e o versionamento do código durante o ciclo de desenvolvimento. O GitHub é uma plataforma de hospedagem e colaboração de código para controle de versão. Ele permite que desenvolvedores armazenem, compartilhem e trabalhem juntos em projetos de software.

Como ferramenta de auxílio à criação e formatação foi utilizado o CKEditor que é um editor de texto *WYSIWYG* (*What You See Is What You Get*) que permite criar e formatar conteúdo diretamente em interfaces web, semelhante a um editor de texto tradicional. Ele suporta diversas opções de formatação, como negrito, itálico, listas e imagens, sem exigir conhecimento de HTML.

As validações de campos obrigatórios em formulários, em especial nas telas de inclusão de registros, foram implementadas utilizando o *SweetAlert*, que oferece uma interface amigável e interativa para alertar o usuário sobre informações pendentes antes de concluir o processo.

Para comunicação com as APIs foi utilizado o *JSON* como formato de dados para a comunicação, facilitando a troca de informações de maneira estruturada e eficiente. Esse formato permitiu que os dados retornados pelas APIs fossem manipulados no protótipo de forma padronizada, simplificando a integração e o acesso às informações necessárias.

Adicionalmente, para o cadastro de veículos, foi integrada a *API* da Tabela FIPE, que permitiu que o sistema obtivesse automaticamente as informações de marcas e modelos de veículos diretamente da base de dados oficial. Essa integração garantiu que os dados inseridos no sistema estivessem sempre atualizados com os valores de mercado, otimizando o processo de cadastramento de novos veículos e eliminando a necessidade de entrada manual de informações complexas.

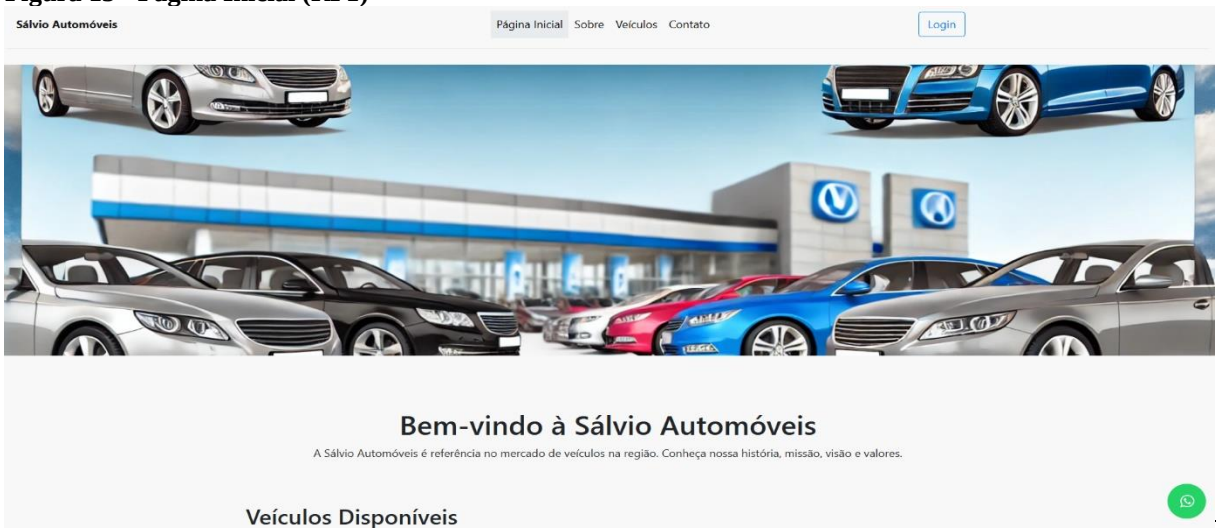
4.2.2 Utilização e Funcionamento

Conforme citado anteriormente, o protótipo foi desenvolvido para facilitar a visualização e gestão dos veículos disponíveis na loja. O sistema possui dois atores principais: o usuário da loja, que pode realizar a navegação e pesquisa através do *website* e tem acesso ao ambiente administrativo, e o usuário comum, que pode navegar e pesquisar apenas pelo *website*.

4.2.2.1. Website

A página inicial do website do protótipo foi projetada para apresentar as principais informações da loja, incluindo um *banner* de boas-vindas, conforme demonstrado na Figura 13. Essa página é composta por dois elementos principais: a seção de veículos disponíveis e as informações institucionais da empresa.

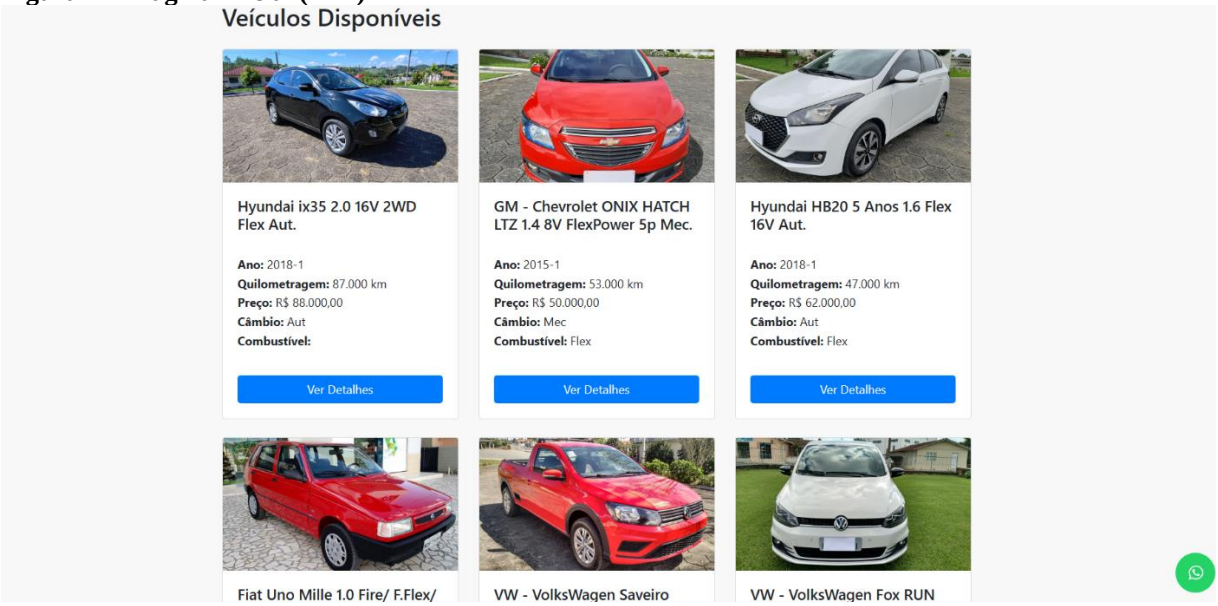
Figura 13 - Página Inicial (RF1)



Fonte: acervo do autor (2024).

A seção "Veículos Disponíveis" (Figura 14) exibe alguns dos veículos disponíveis na loja, apresentando informações como marca, modelo, ano, quilometragem, valor, câmbio e combustível (conforme RF1). A listagem permite ao usuário visualizar rapidamente as opções disponíveis e obter detalhes de cada veículo com facilidade.

Figura 14 - Página Inicial (RF1)



Fonte: acervo do autor (2024).

A seção "Sobre Nós" (Figura 15) inclui um resumo da história da empresa, reforçando a identidade corporativa da Sálvio Automóveis, sendo informativa ao usuário desde o primeiro acesso. Essa seção é importante para transmitir ao visitante a trajetória, missão, visão e valores da empresa, conforme estipulado pelas regras de negócio associadas ao protótipo. Além disso, a página inicial e todas as outras páginas do website contam com uma integração direta ao *WhatsApp*, disponibilizando um ícone de acesso rápido no canto inferior da tela (RF9), o que facilita a comunicação direta com a empresa.

Figura 15 - Página Inicial (RF1) e Integração com WhatsApp (RF9)

Fonte: acervo do autor (2024).

A página "Sobre" do protótipo foi projetada para apresentar informações institucionais da empresa, conforme Figura 16. Esta página inclui a história da Sálvio Automóveis, além de sua missão, visão e valores, (conforme RF2). Essas informações são fundamentais para reforçar a identidade corporativa da empresa e para informar o usuário sobre o compromisso da Sálvio Automóveis com seus clientes e com a qualidade dos serviços prestados.

Figura 16 - Sobre (RF2)

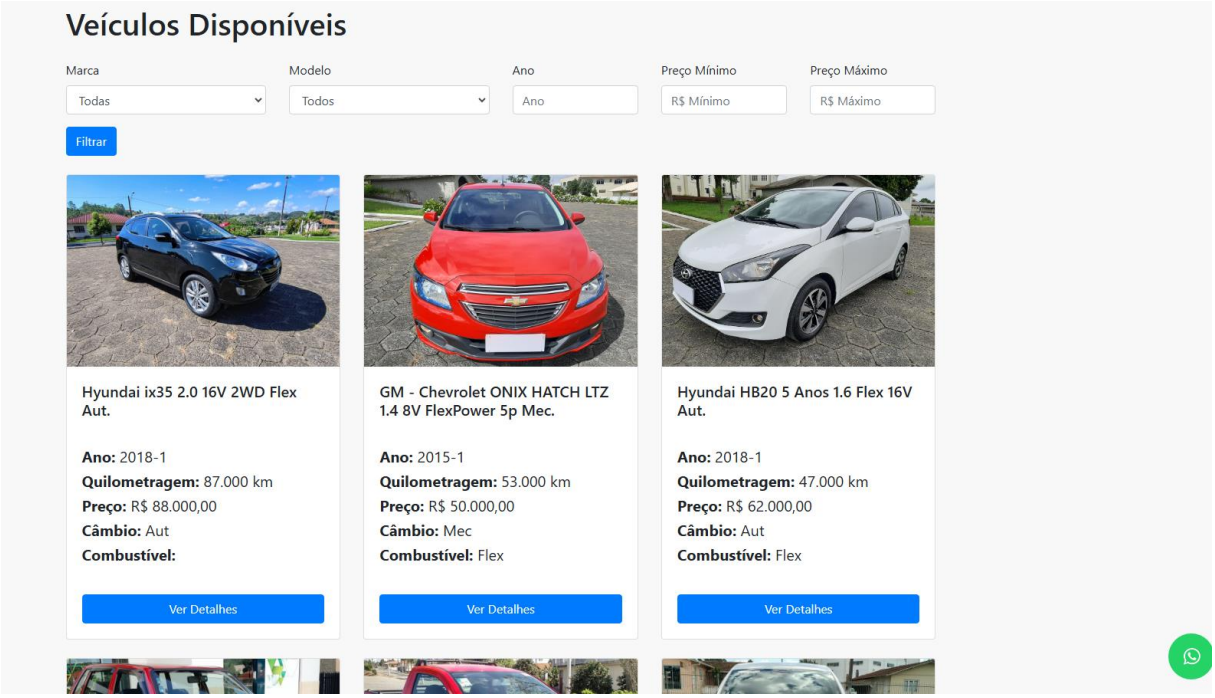
Fonte: acervo do autor (2024).

A página "Veículos" do protótipo foi desenvolvida para exibir uma listagem dos veículos disponíveis na loja, conforme demonstra a Figura 17. Esta página oferece aos usuários alguns detalhes dos veículos, como marca, modelo, ano, quilometragem, valor, câmbio e tipo de combustível, atendendo ao RF3.

Além disso, o protótipo implementa filtros que permitem ao usuário refinar sua busca com base em marca, modelo, ano e faixa de preço. Esses filtros tornam a navegação mais intuitiva e direcionada, facilitando a busca por veículos que atendam às necessidades específicas do cliente.

O botão de “Ver Detalhes” levará diretamente a fotos e maiores informações do veículo selecionado.

Figura 17 - Veículos (RF3)

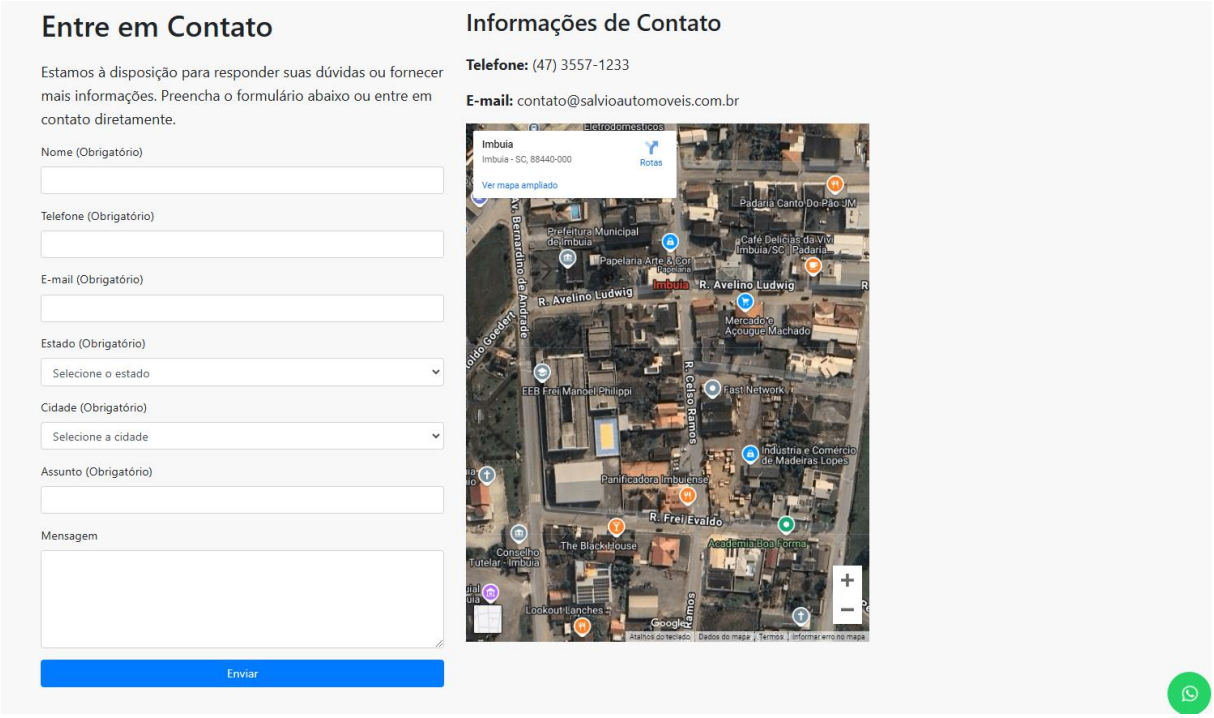


Fonte: acervo do autor (2024).

A página "Contato" do protótipo foi desenvolvida para permitir que os usuários entrem em contato diretamente com a empresa através de um formulário (conforme Figura 18). Esta página inclui um formulário com os campos obrigatórios, como nome, telefone, e-mail, estado, cidade, assunto e mensagem (de acordo com RF4). Esses campos são validados antes do envio para garantir que o usuário forneça informações essenciais a empresa.

Além do formulário, a página exibe um mapa com a localização da empresa, e inclui as informações de contato, como telefone e e-mail. Para o preenchimento dos campos de estado e cidade foi utilizada a *API* do *IBGE (Instituto Brasileiro de Geografia e Estatística)*, o que torna o processo de preenchimento mais simples para o usuário.

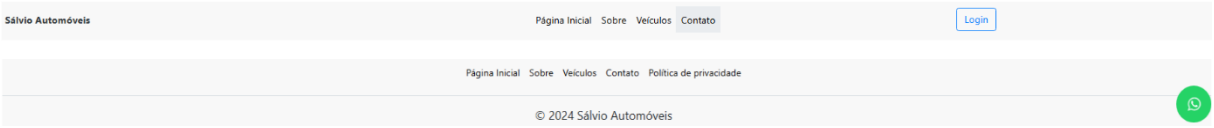
Figura 18 - Contato (RF4)



Fonte: acervo do autor (2024).

O protótipo também inclui um menu principal que permite o redirecionamento para todas as páginas essenciais do site, conforme a Figura 19. O menu contém *links* para as páginas "Página Inicial", "Sobre", "Veículos", "Contato" e "Política de Privacidade", (atendendo ao RF7). Esse menu está presente em todas as páginas do site, proporcionando uma navegação fácil e intuitiva para os usuários.

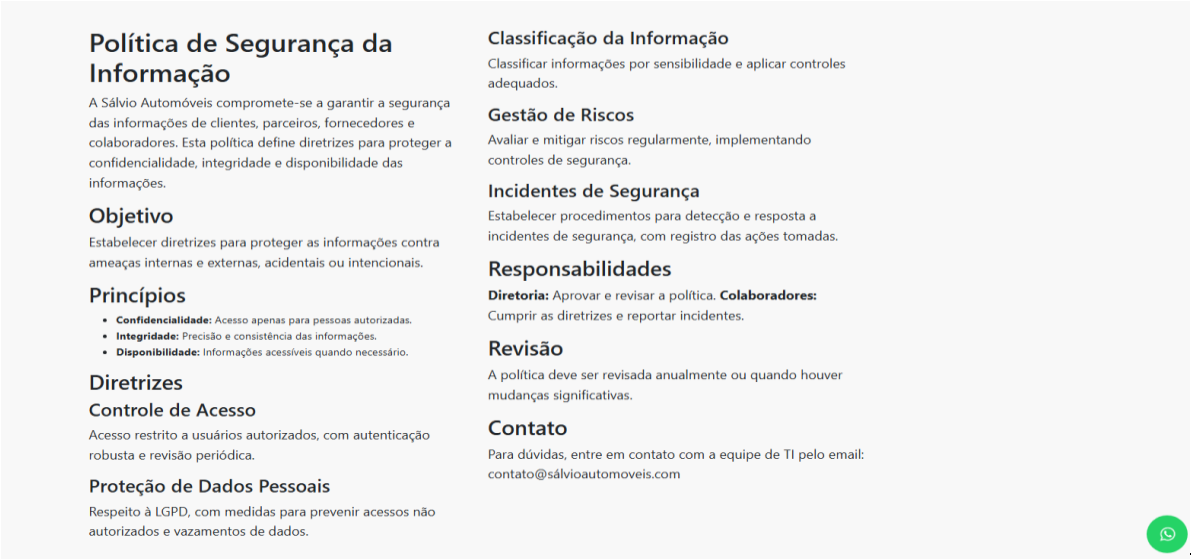
Figura 19 - Menu Principal (RF7)



Fonte: acervo do autor (2024).

A página de "Política de Privacidade", conforme a Figura 20, apresenta as diretrizes da empresa para a proteção das informações dos clientes (de acordo com o RF5). Essa página detalha o compromisso da empresa com a segurança e a privacidade dos dados, destacando princípios como a confidencialidade, integridade e disponibilidade das informações. Também são incluídas orientações sobre controle de acesso definidas pelo cliente em conjunto a empresa, proteção de dados pessoais em conformidade com a *LGPD* (*Lei Geral de Proteção de Dados*), além de informações sobre a classificação de dados e a gestão de riscos.

Figura 20 - Política de Privacidade (RF5)

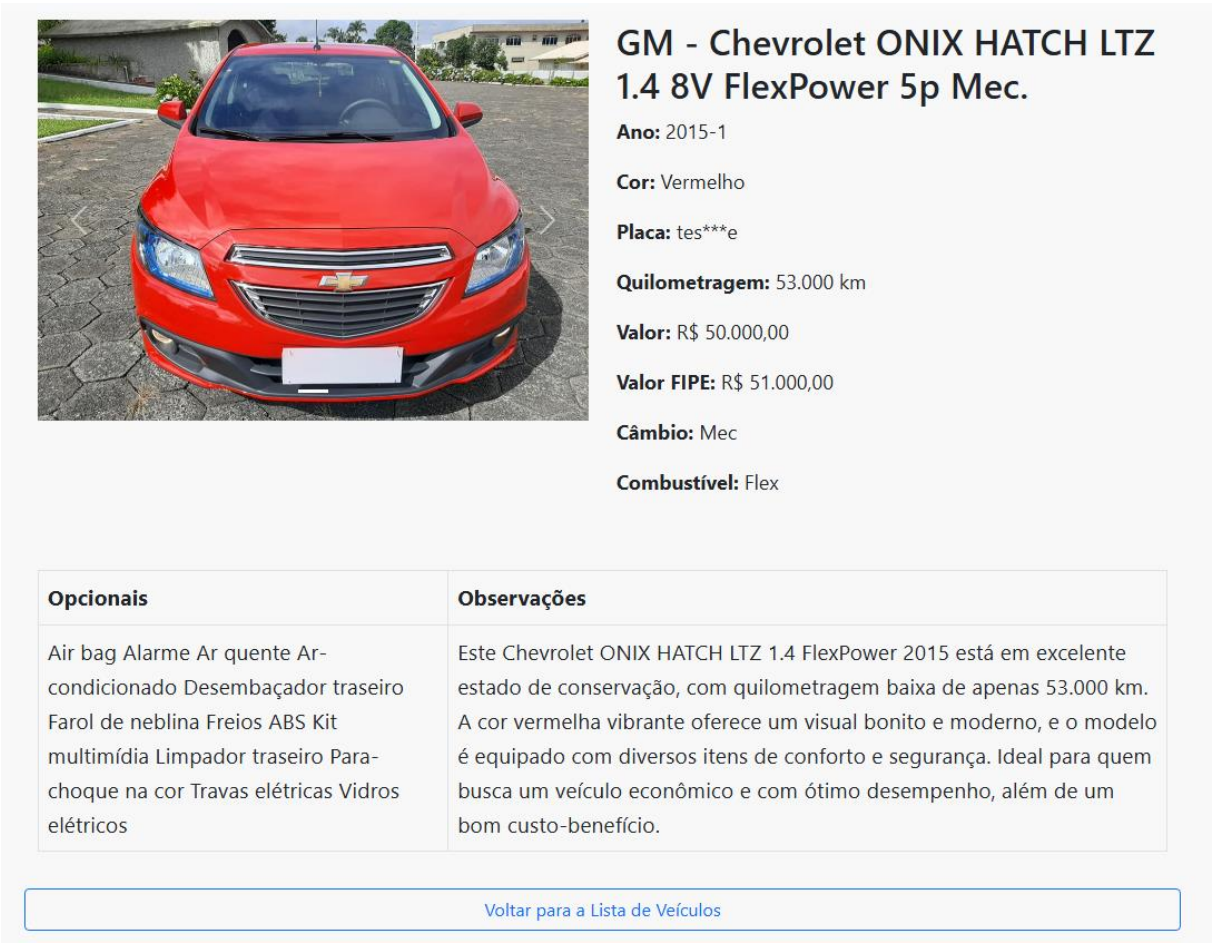


Fonte: acervo do autor (2024).

A página de "Detalhamento do Veículo" do protótipo foi desenvolvida para exibir informações específicas sobre cada veículo disponível na loja, conforme demonstrado na Figura 21. Esta página contém detalhes como imagem, marca, ano, modelo, cor, placa (apenas caracteres iniciais e finais), quilometragem, valor, câmbio, combustível, além de observações adicionais e acessórios (conforme RF8). Além disso, a página inclui um botão para voltar à lista de veículos, facilitando a navegação do usuário.

Adicionalmente, o protótipo implementa uma funcionalidade que permite consultar automaticamente o valor do veículo na tabela FIPE, (conforme previsto no RF6), utilizando a *API* da FIPE para garantir que os valores estejam sempre atualizados.

Figura 21 - Detalhamento do Veículo (RF8) e Consulta a tabela FIPE (RF6)



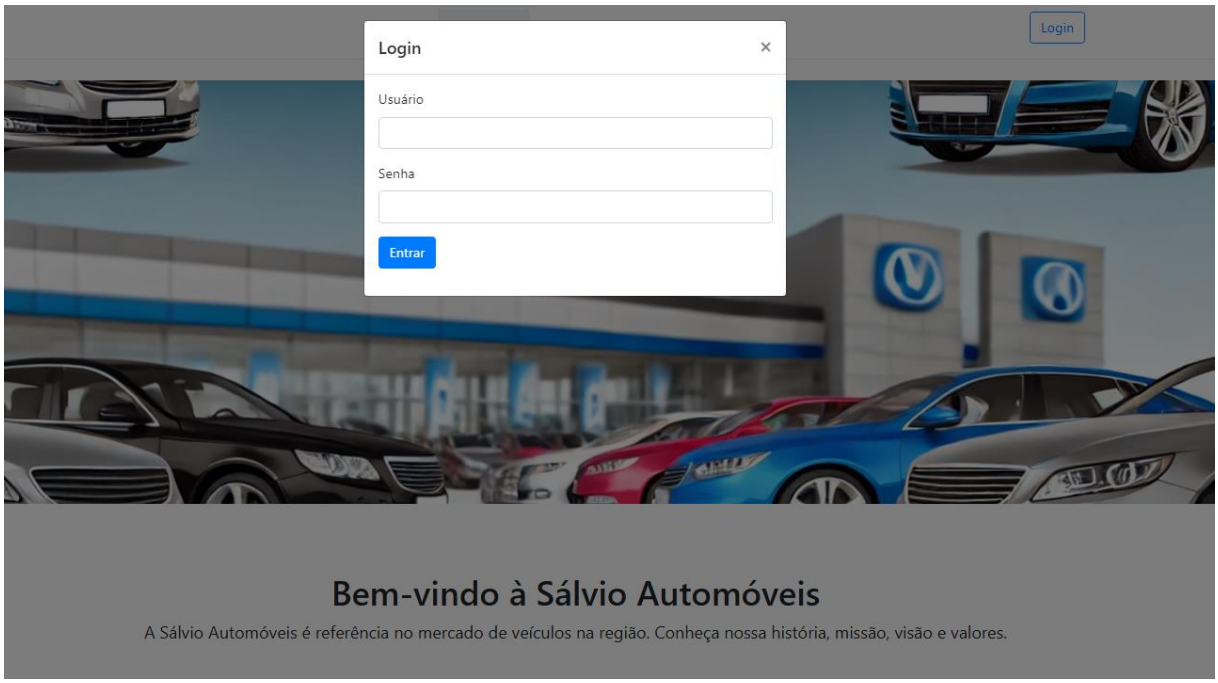
Fonte: acervo do autor (2024).

4.2.2.2. Ambiente Administrativo

Para acesso ao ambiente administrativo o protótipo possui uma rotina de login destinada ao lojista, conforme demonstrado na Figura 22. Esta funcionalidade garante a segurança por meio do controle de acesso com *login* e senha (conforme exigido pelo RF10).

Para realizar o *login*, o usuário deve fornecer seu nome de usuário e uma senha, que, conforme especificado pela regra de negócio RN2, deve conter no mínimo 8 caracteres, pelo menos uma letra maiúscula, uma letra minúscula, um número e um caractere especial. Essa medida assegura a proteção das informações sensíveis e limita o acesso ao ambiente administrativo somente a usuários autorizados.

Figura 22 - Login (RF10)



Fonte: acervo do autor (2024).

A funcionalidade de "Manutenção de Veículos" (RF11) no protótipo permite a inclusão, atualização e exclusão de veículos, conforme demonstrado na Figura 23. Esta tela permite que o administrador da loja tenha controle total sobre os veículos, apresentando uma lista com as principais informações, como marca, modelo, ano, placa e valor do mesmo.

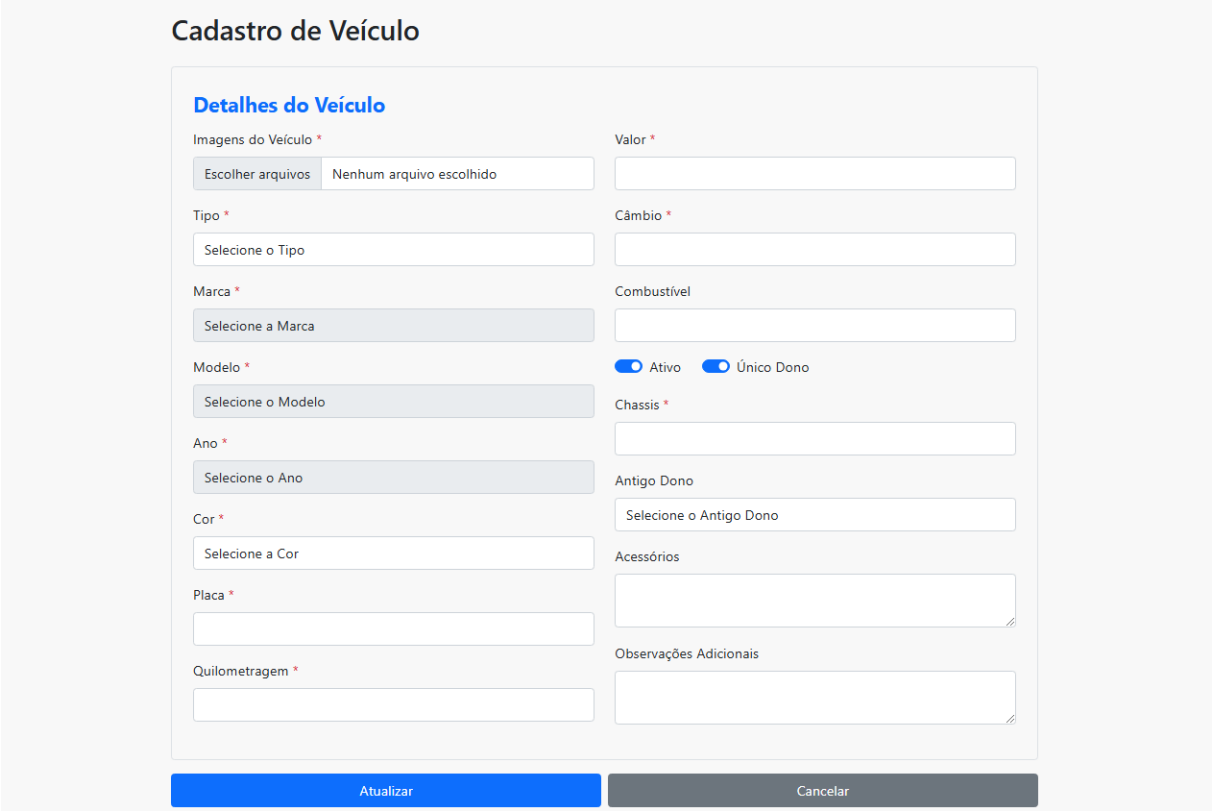
Figura 23 - Manutenção de Veículos (RF11)

Fonte: acervo do autor (2024).

Para atender às regras de negócio descritas no RF11, na Figura 24 cada veículo deve estar devidamente cadastrado no sistema com campos obrigatórios, como imagem, marca, ano, modelo, cor, tipo, placa, quilometragem, valor, câmbio, entre outros. Esses campos são necessários para garantir que todas as informações relevantes sejam inseridas e mantidas corretamente no banco de dados.

Além disso, o protótipo utiliza a *API* da FIPE para preencher os campos de marca e modelo, visando garantir que as informações estejam sempre atualizadas de acordo com o mercado. O sistema também verifica que todos os campos obrigatórios estejam preenchidos antes de permitir que o veículo seja incluído.

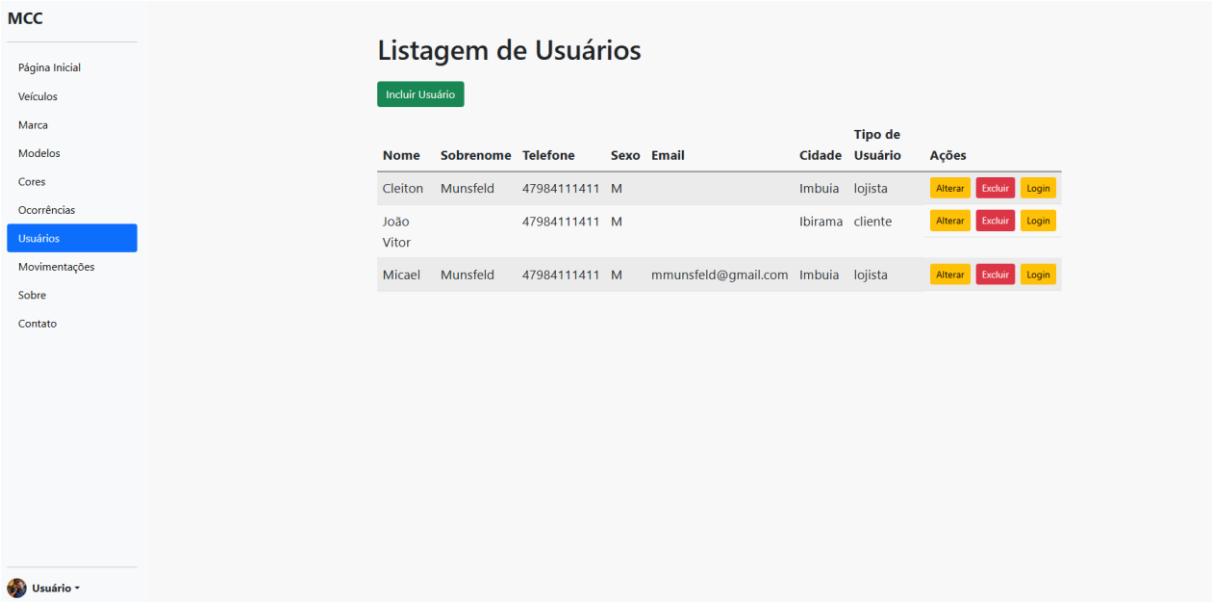
Figura 24 - Manutenção de Veículos (RF11)



Fonte: acervo do autor (2024).

A funcionalidade de "Manutenção de Usuários" (RF12) no protótipo permite o cadastro, atualização e exclusão de usuários, conforme ilustrado na Figura 25. Esta rotina facilita a administração dos usuários, exibindo uma lista com detalhes como nome, sobrenome, telefone, sexo, *e-mail*, cidade e tipo de usuário. As ações de alteração e exclusão estão disponíveis para cada registro. Além disso, a ação de login permite ao lojista informar uma senha de acesso, que utilizará para acessar o ambiente administrativo’.

Figura 25 - Manutenção de Usuários (RF12)



Fonte: acervo do autor (2024).

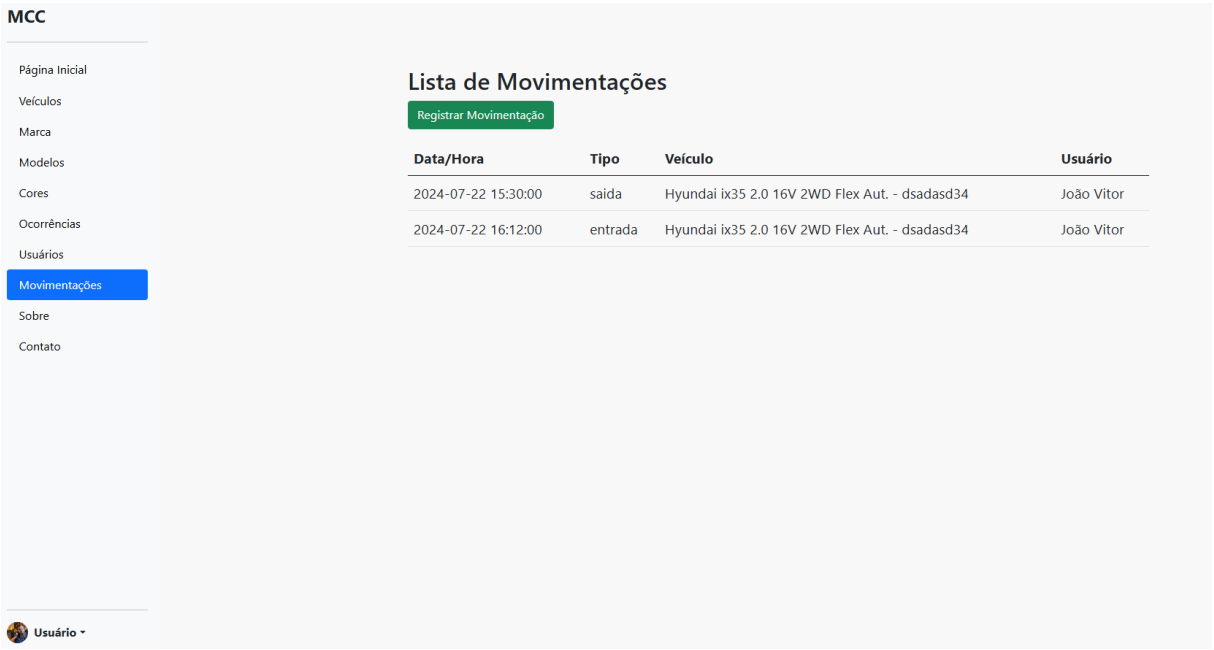
Para realizar o cadastro de novos usuários, o protótipo exige o preenchimento de campos obrigatórios, como nome, telefone, sexo, cidade, tipo de usuário e senha (para usuários do tipo lojista), conforme RF12 e ilustrado na Figura 26.

Figura 26 - Manutenção de Usuários (RF12)

Fonte: acervo do autor (2024).

A funcionalidade de "Histórico de Movimentações" no protótipo foi implementada para registrar e manter um histórico das movimentações de entrada e saída dos veículos na loja, conforme demonstrado na Figura 27. Esta funcionalidade cumpre os requisitos definidos no RF13 e permite que o sistema armazene informações como data e hora, nome do veículo, tipo de movimentação (entrada ou saída) e o usuário responsável pela movimentação.

Figura 27 - Histórico de Movimentações (RF13)



Fonte: acervo do autor (2024).

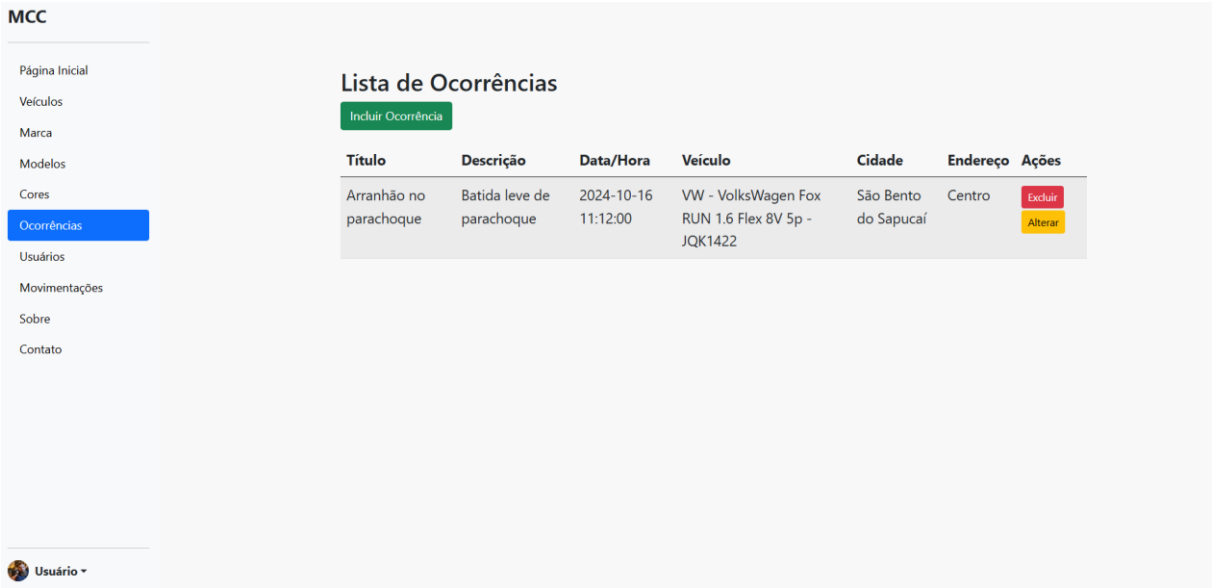
O formulário de registro de movimentações conforme Figura 28, exige que os campos obrigatórios, como data/hora, veículo e tipo de movimentação, sejam preenchidos corretamente. O que garante que as movimentações sejam registradas e que o histórico possa ser consultado posteriormente para fins administrativos e de controle.

Figura 28 - Histórico de Movimentações (RF13)

Fonte: acervo do autor (2024).

A funcionalidade de "Ocorrências do Veículo" foi implementada no protótipo para registrar e listar eventuais ocorrências relacionadas aos veículos, conforme demonstrado nas Figuras 29. Esta rotina permite que o lojista registre problemas ou eventos envolvendo os veículos, cumprindo as exigências do RF14.

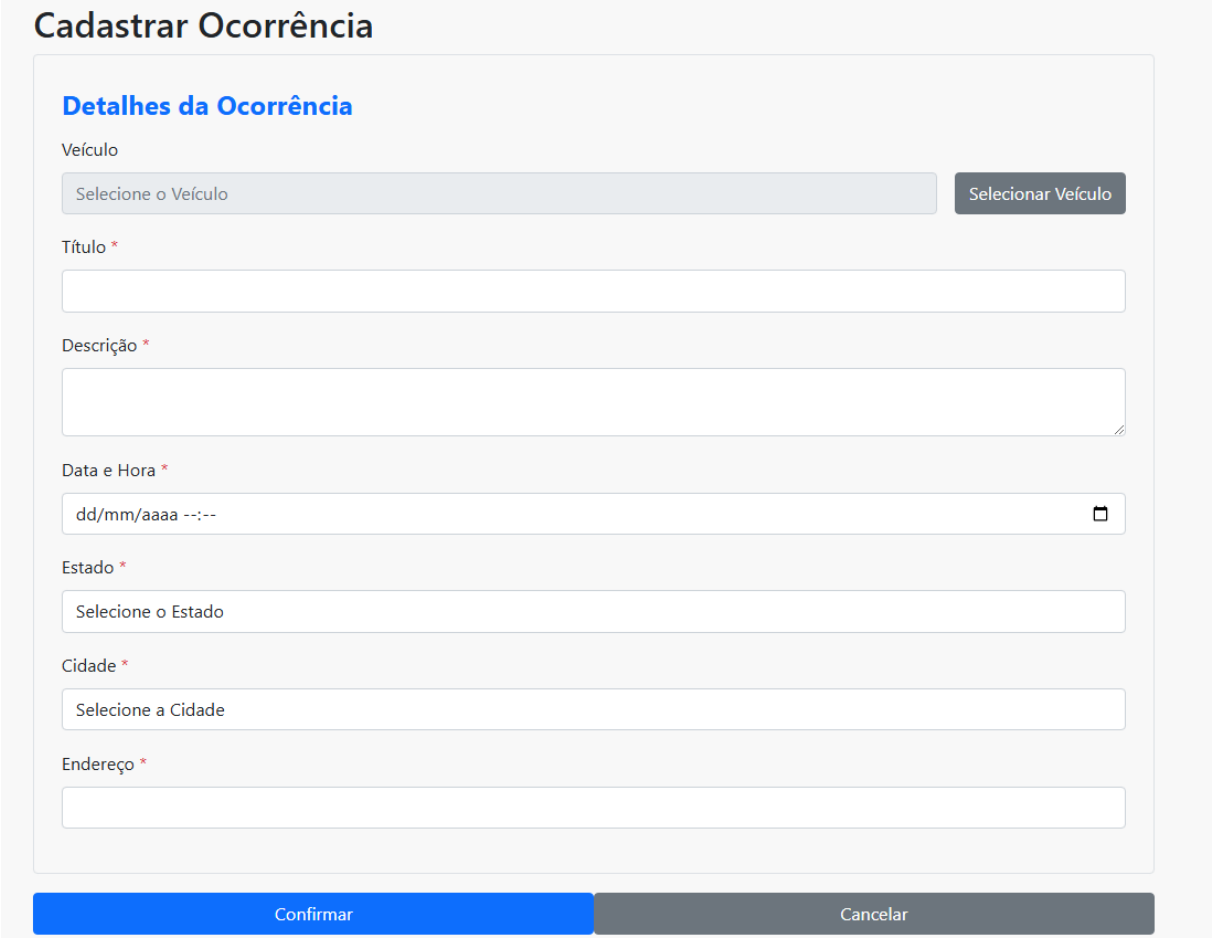
Figura 29 - Ocorrências do Veículo (RF14)



Fonte: acervo do autor (2024).

Cada ocorrência registrada deve conter os campos obrigatórios, como título, descrição, data/hora, cidade, endereço e o veículo. O protótipo (conforme a Figura 30), facilita a organização dessas informações permitindo consulta às ocorrências registradas, e oferece opções para alterar ou excluir as informações conforme necessário.

Figura 30 - Ocorrências do veículo (RF14)



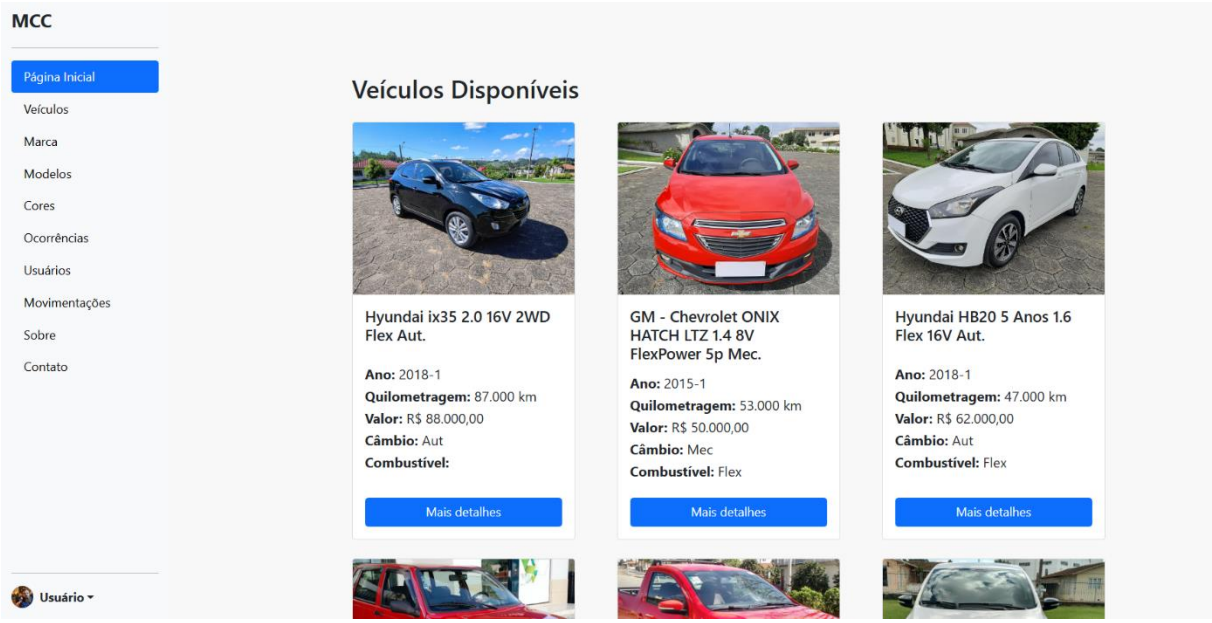
Fonte: acervo do autor (2024).

A página inicial do ambiente administrativo, conforme demonstrado na Figura 31, exibe uma lista dos veículos disponíveis na loja, permitindo ao lojista visualizar inicialmente informações importantes do veículo como marca, modelo, ano, quilometragem, valor, câmbio e tipo de combustível, em conformidade com o RF15. A página inicial oferece um ponto de partida para navegação pelo sistema, proporcionando acesso fácil às demais telas e funcionalidades do protótipo.

Além disso, ao clicar no botão "Mais Detalhes" localizado abaixo de cada veículo, o usuário será redirecionado para a tela de detalhamento do veículo selecionado, onde poderá visualizar ou alterar informações do veículo.

O ambiente administrativo também conta com um menu principal localizado na lateral esquerda da tela (Figura 31), atendendo ao RF16. Este menu oferece acesso rápido e direto a todas as rotinas do sistema.

Figura 31 - Página Inicial (RF15) e Menu Principal (RF16)



Fonte: acervo do autor (2024).

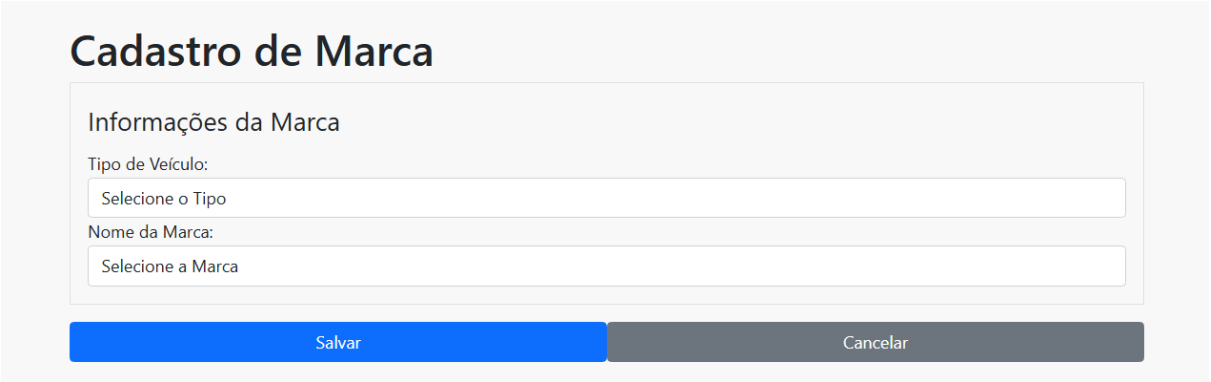
A funcionalidade de "Manutenção de Marca" (Figura 32) permite o a consulta, cadastro, atualização e exclusão de marcas de veículos. Esta funcionalidade cumpre os requisitos definidos no RF17, proporcionando ao administrador do sistema o controle sobre as marcas disponíveis na loja.

Figura 32 - Manutenção de Marca (RF17)

Fonte: acervo do autor (2024).

Para permitir que as informações de marca estejam sempre atualizadas e corretas, o preenchimento do nome da marca utiliza a *API* da FIPE, conforme Figura 33. Isso permite que o sistema traga automaticamente os dados de marcas diretamente da tabela FIPE, facilitando o cadastro e visando uma melhor precisão das informações.

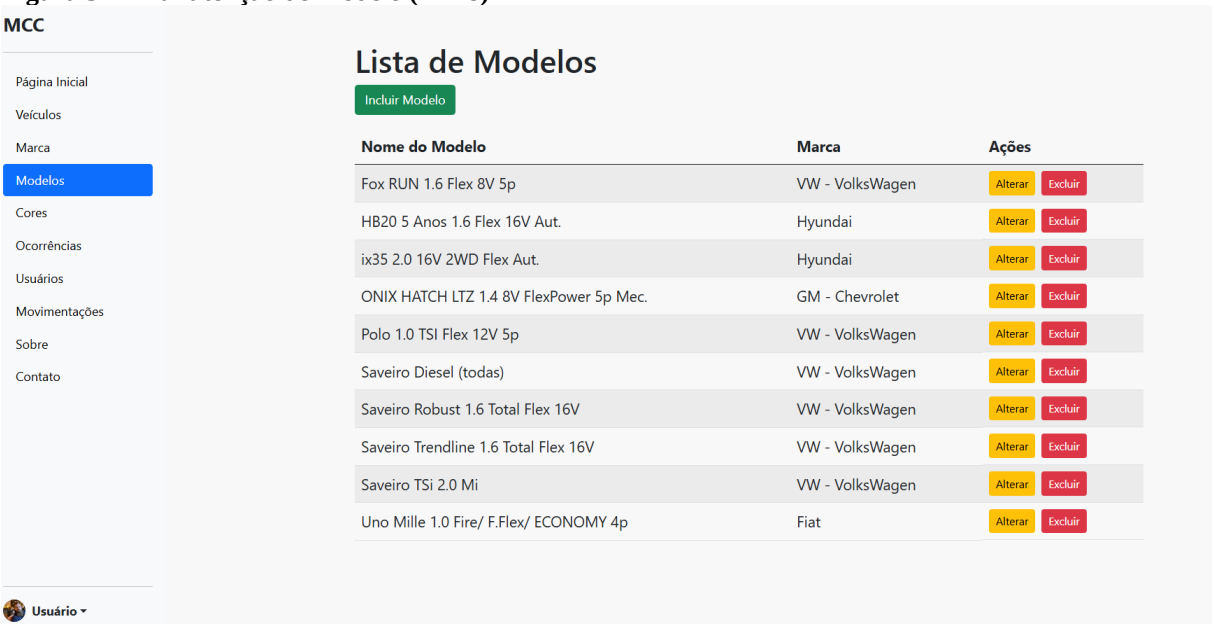
Figura 33 - Manutenção de marca (RF17)



Fonte: acervo do autor (2024).

A funcionalidade de "Manutenção de Modelo" no protótipo, conforme a Figura 34, permite o cadastro, atualização, exclusão e consulta dos modelos de veículos. Esta funcionalidade atende o RF18, tal qual, possibilita ao administrador a gestão dos modelos de veículos disponíveis na loja.

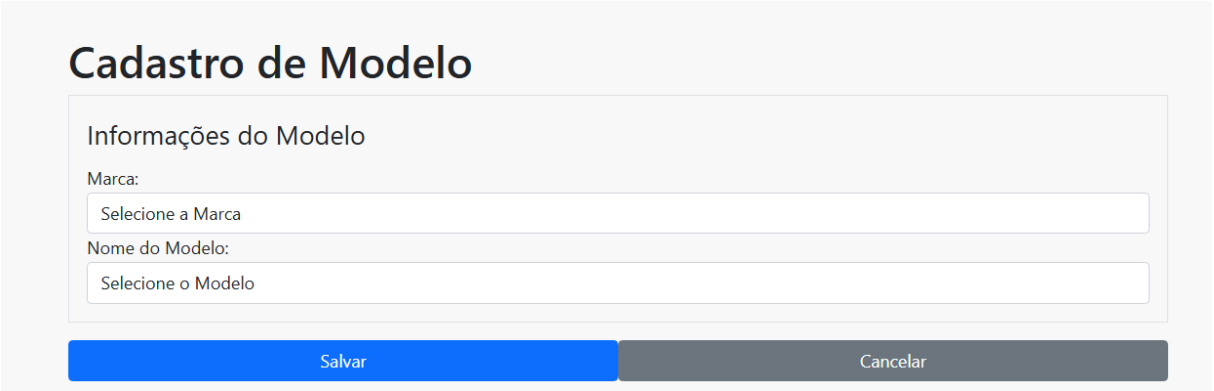
Figura 34 - Manutenção de Modelo (RF18)



Fonte: acervo do autor (2024).

Para cadastro do modelo o protótipo utiliza a *API* da FIPE (Figura 35). Isso permite que os dados dos modelos sejam obtidos diretamente da tabela FIPE, visando garantir que os modelos cadastrados estejam atualizados com as informações mais recentes do mercado.

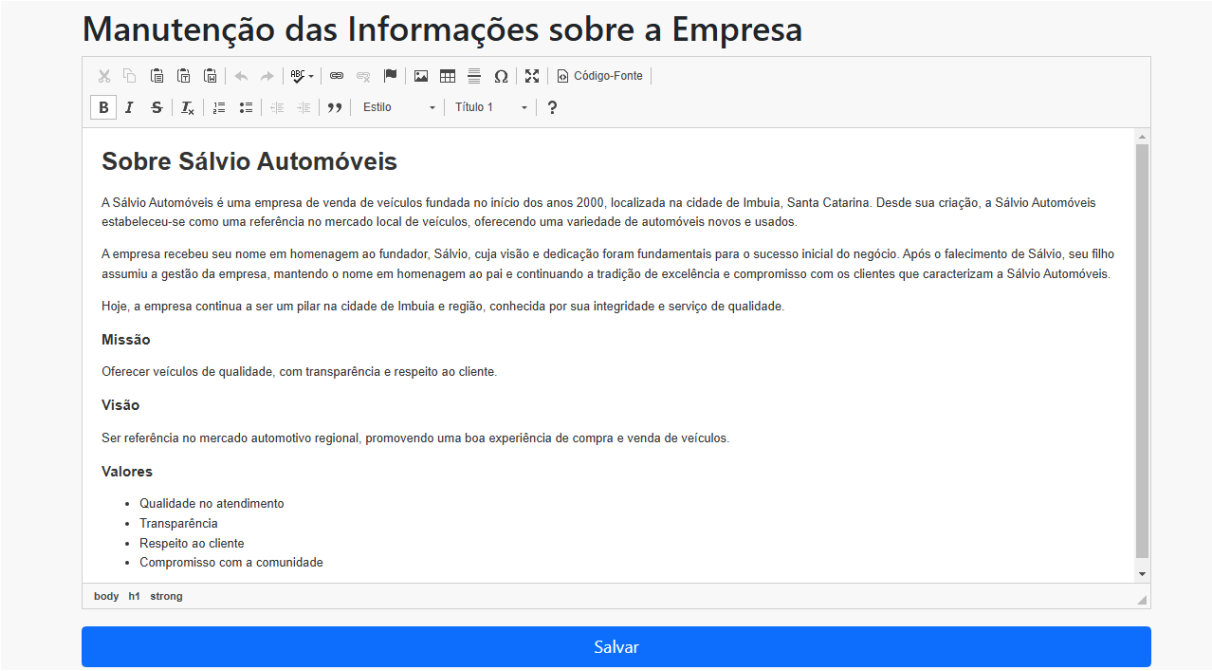
Figura 35 - Manutenção de modelo (RF18)



Fonte: acervo do autor (2024).

A funcionalidade de "Manutenção Sobre" do ambiente administrativo, conforme Figura 36, permite que o administrador do sistema gerencie as informações institucionais da empresa, tais como a história, missão, visão e valores. Esta rotina atende ao RF19, oferecendo um editor de texto com opções de formatação que facilitam a edição e atualização dos dados.

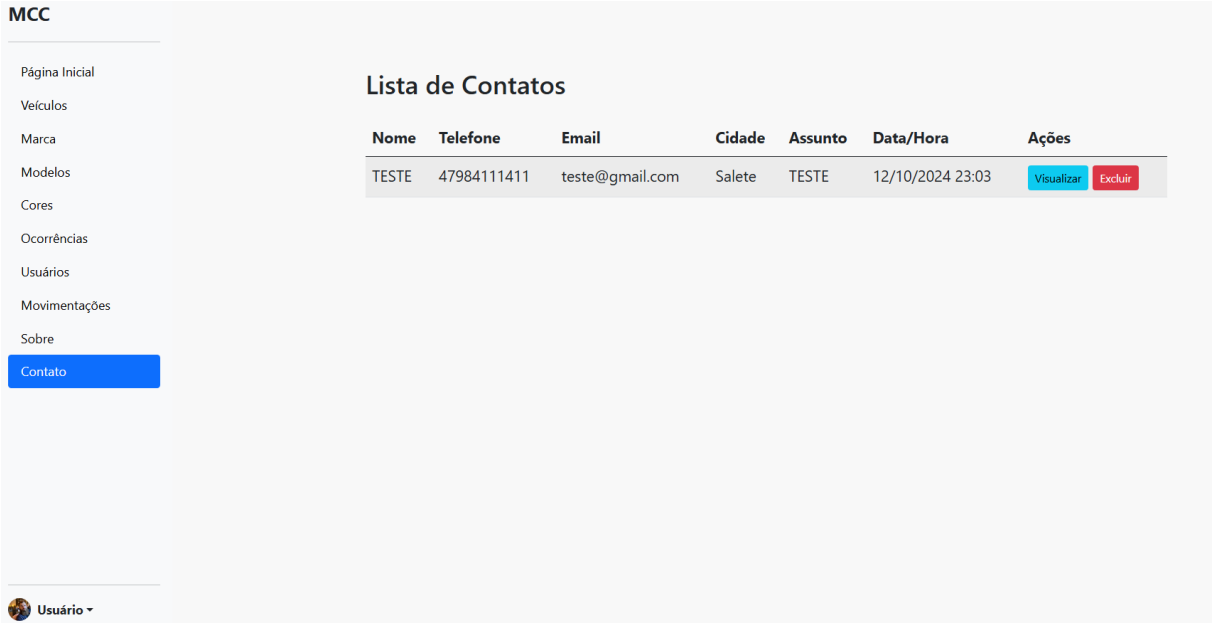
Figura 36 - Manutenção do Sobre (RF19)



Fonte: acervo do autor (2024).

De acordo com o RF20, o protótipo deve permitir a manutenção das informações de contato da empresa, incluindo a visualização dos dados enviados pelo formulário de contato no website, conforme a Figura 37. A tela exibe uma tabela com detalhes como nome, telefone, *e-mail*, cidade, assunto, data/hora e opções de "Visualizar" ou "Excluir".

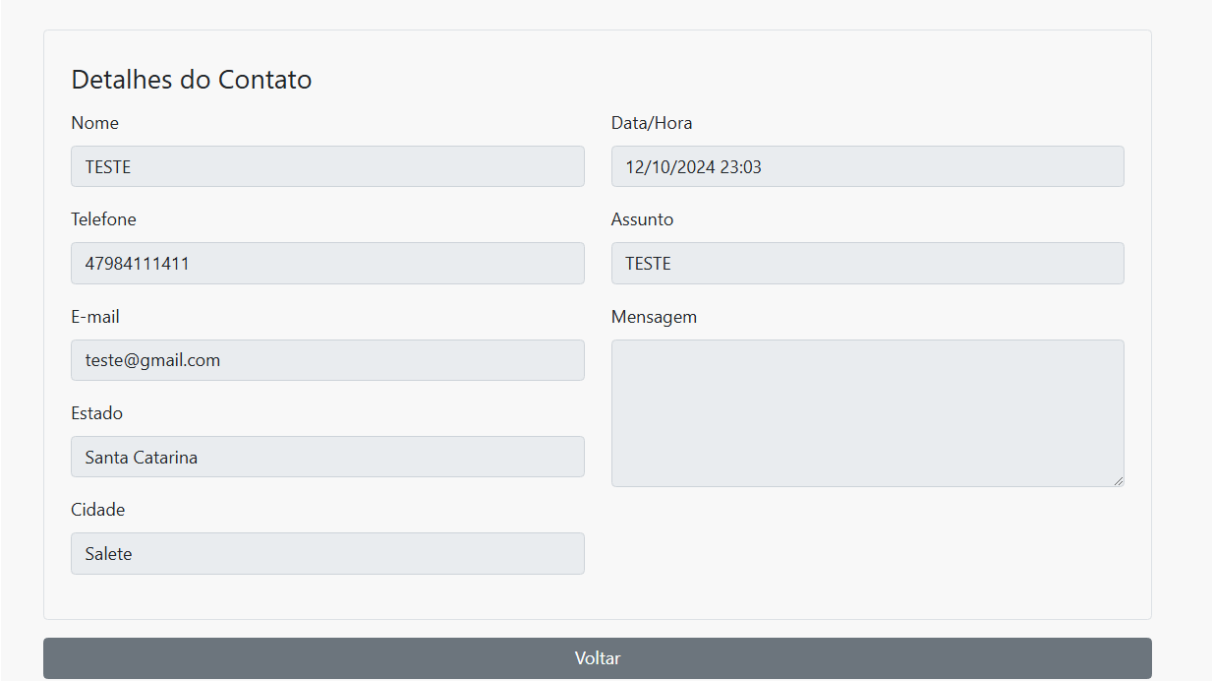
Figura 37 - Manutenção de Contato (RF20)



Fonte: acervo do autor (2024).

A tela de visualização do contato (conforme a Figura 38), denominada de "Detalhes do Contato", é possível visualizar todas as informações do contato selecionado, como nome, telefone, *e-mail*, cidade, estado e mensagem enviada.

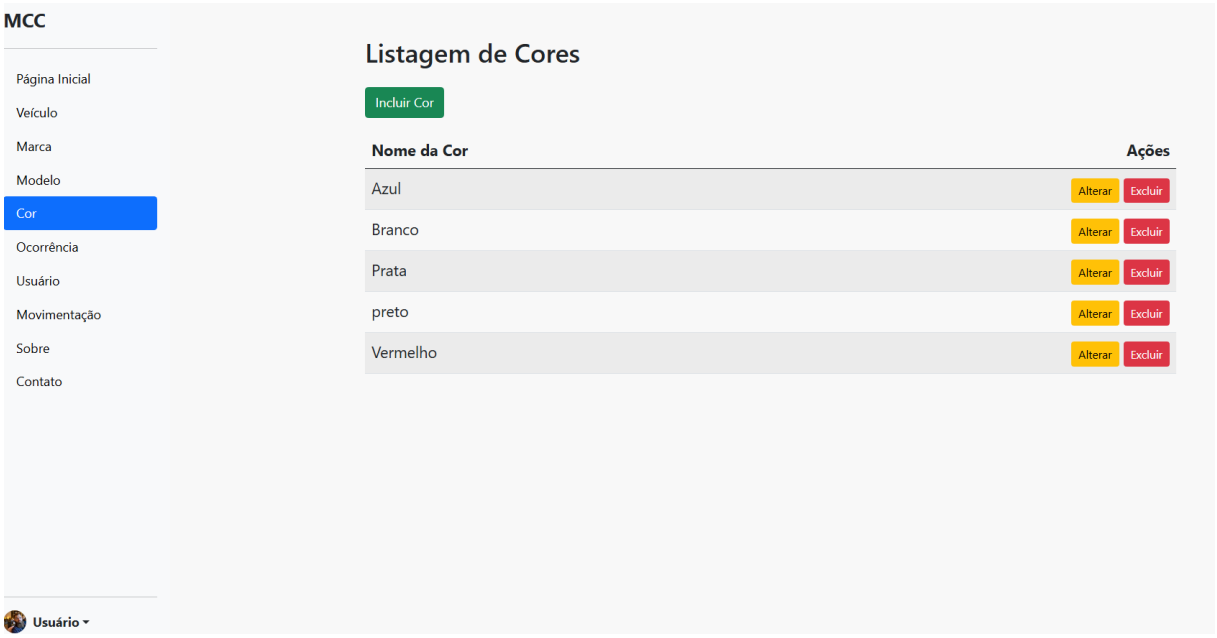
Figura 38 - Manutenção de Contato (RF20)



Fonte: acervo do autor (2024).

A funcionalidade de "Manutenção de Cor" no protótipo, como demonstrado na Figura 39, permite que o lojista gerencie as cores disponíveis para os veículos cadastrados. Essa rotina atende ao RF21, oferecendo opções de cadastro, atualização e exclusão das cores dos veículos.

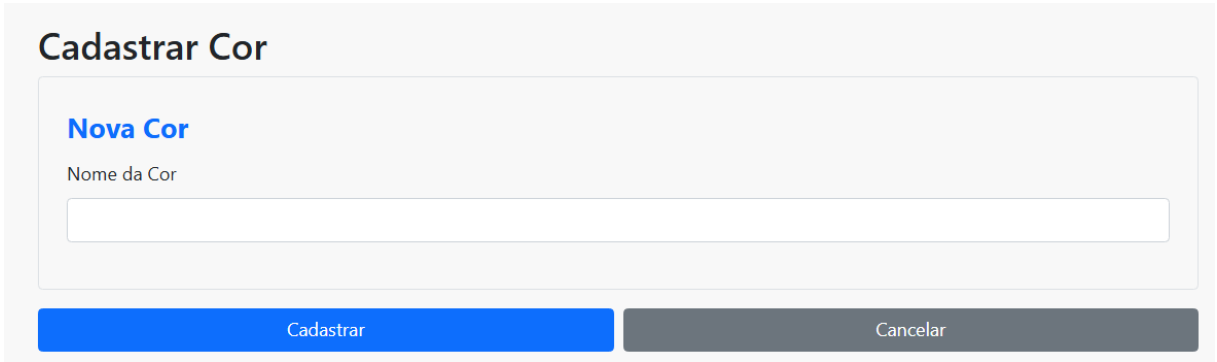
Figura 39 - Manutenção de Cor (RF21)



Fonte: acervo do autor (2024).

Ao clicar em "Incluir Cor", o usuário pode adicionar uma nova cor ao sistema, conforme demostrado na Figura 40. O protótipo também oferece a opção de alterar ou excluir cores já existentes na lista, conforme mencionado anteriormente. Essa funcionalidade é essencial, pois visa garantir que todas as cores disponíveis estejam corretamente cadastradas e gerenciadas, permitindo ao administrador manter o controle sobre essa informação no protótipo para posteriormente associar a um veículo.

Figura 40 - Manutenção de Cor (RF21)

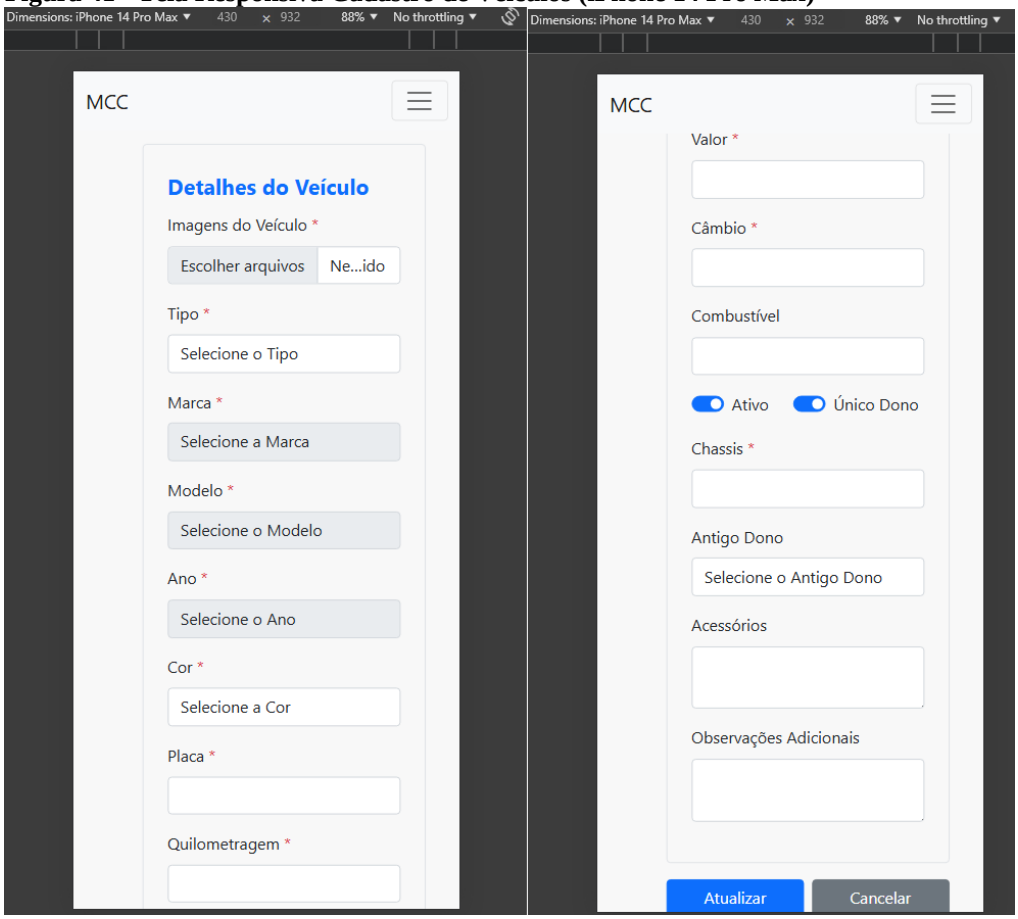


Fonte: acervo do autor (2024).

Conforme mencionado, o protótipo foi desenvolvido para ser totalmente responsivo (RNF5), adaptando-se a diferentes resoluções de tela. Na Figura 41, pode-se observar a

visualização do formulário de Cadastro de Veículos (RF11) em um dispositivo móvel, exemplificado com a tela de um iPhone 14 Pro Max.

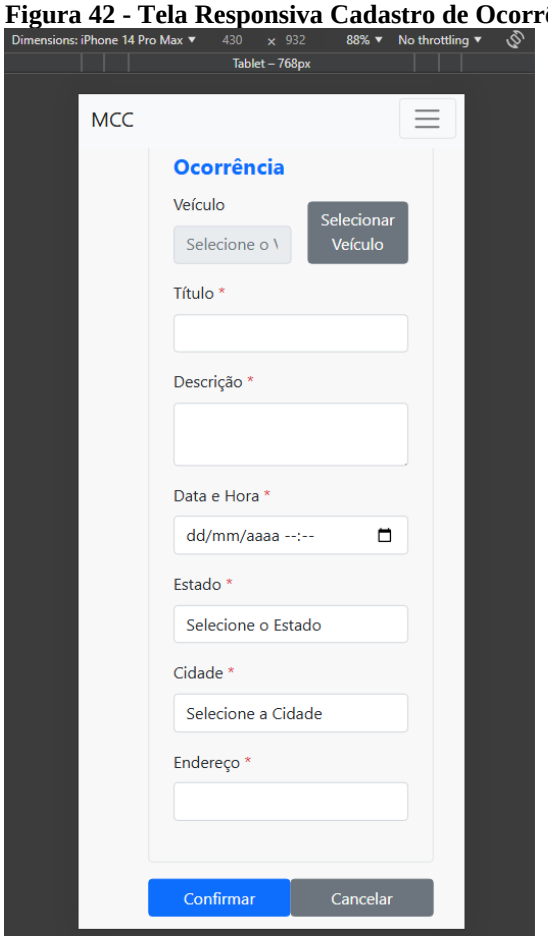
Figura 41 - Tela Responsiva Cadastro de Veículos (iPhone 14 Pro Max)



Fonte: acervo do autor (2024).

Na Figura 42, observa-se o Formulário de Cadastro de Ocorrências (RF14) em um dispositivo móvel, neste exemplo, utilizando a tela de um iPhone 14 Pro Max.

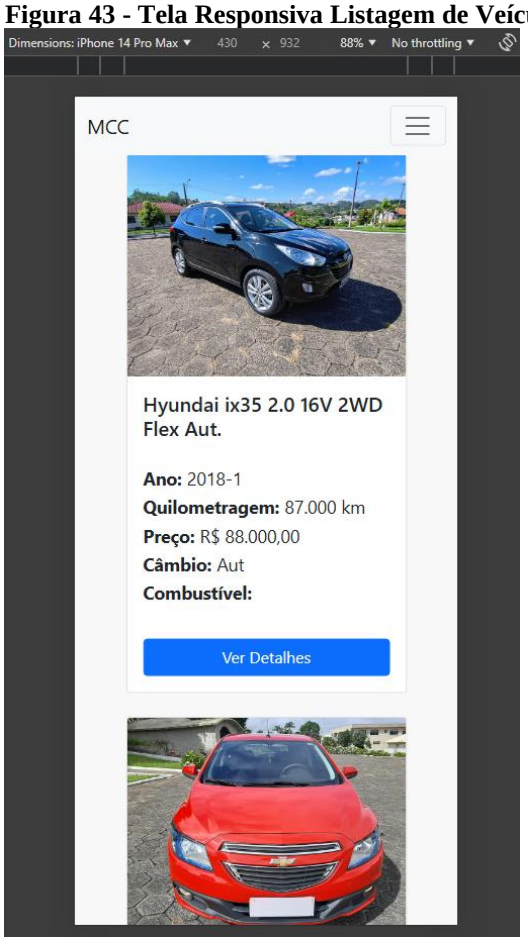
Figura 42 - Tela Responsiva Cadastro de Ocorrências (iPhone 14 Pro Max)



Fonte: acervo do autor (2024).

A Figura 43 apresenta a tela de listagem de veículos do website visualizada em um dispositivo móvel (RF3), mais especificamente em um iPhone 14 Pro Max.

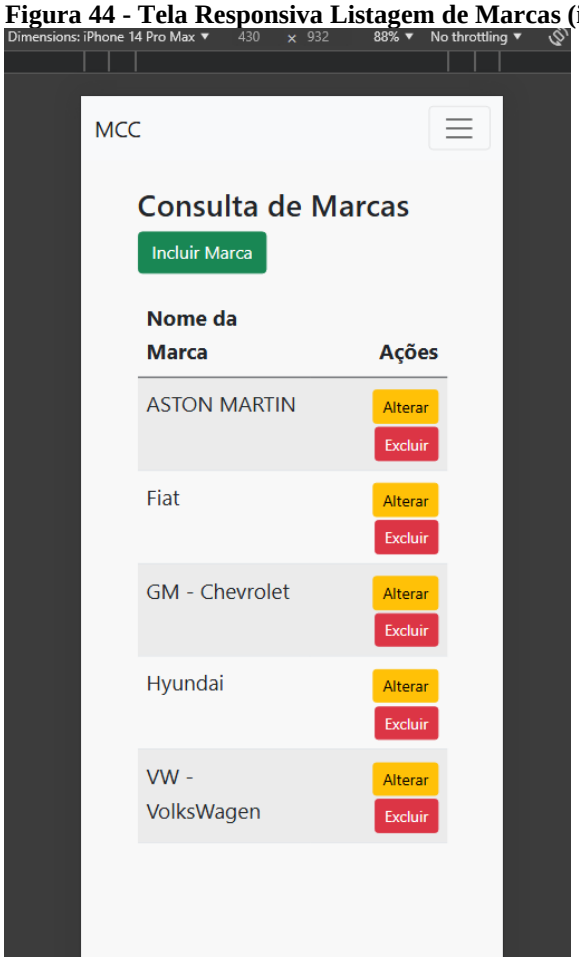
Figura 43 - Tela Responsiva Listagem de Veículos (iPhone 14 Pro Max)



Fonte: acervo do autor (2024).

A Figura 44 apresenta a tela com a listagem de marcas no ambiente administrativo, visualizada em um dispositivo móvel (RF17), mais especificamente em um iPhone 14 Pro Max.

Figura 44 - Tela Responsiva Listagem de Marcas (iPhone 14 Pro Max)



Fonte: acervo do autor (2024).

O próximo capítulo apresenta as considerações finais sobre o trabalho, destacando as contribuições do protótipo para o gerenciamento de veículos em concessionárias e sugerindo melhorias para trabalhos futuros.

5. CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso teve como objetivo desenvolver o MCC (*Munsfeld Car Control*), um protótipo de software web para gerenciar a entrada e saída de veículos em lojas de automóveis. Baseado em uma pesquisa aplicada e descritiva, o protótipo foi projetado para otimizar o controle operacional e a gestão de estoque, oferecendo uma interface prática para lojistas.

As tecnologias adotadas e a linguagem de programação escolhida foram adequdas para atender o desenvolvimento do protótipo. Para a implementação do protótipo foi utilizado no *frontend* a biblioteca *Bootstrap*, que dispõe de componentes estilizados e ajustáveis, simplificando a construção das interfaces com *HTML* e *CSS*. O *JavaScript*, junto com a biblioteca *jQuery*, permitiu a criação de interações mais dinâmicas, enquanto o *SweetAlert* foi integrado para fornecer notificações ao usuário. No *backend*, o PostgreSQL foi escolhido visando a garantia de um armazenamento seguro e eficiente dos dados, proporcionando uma solução robusta e escalável. Para comunicação com as *APIs* foi utilizado o formato de dados *JSON*. Além disso, o *PHP* foi a principal ferramenta utilizada para o desenvolvimento do *backend*, responsável pelo processamento dos dados e comunicação com o banco de dados.

Em relação aos objetivos específicos do trabalho, o primeiro objetivo visava detalhar os conceitos e as tecnologias empregadas no desenvolvimento do protótipo, este foi alcançado através do conteúdo apresentado na revisão da literatura.

O segundo objetivo consistia em analisar softwares semelhantes ao protótipo proposto. Durante o desenvolvimento deste objetivo foram identificadas quatro ferramentas com funcionalidades similares, sendo estas demonstradas na seção do estado da arte. As aplicações demonstradas foram: Autoconf, Altimus, BorbaCar e Encontre Veículos. Sendo que, nas duas primeiras foi possível detalhar melhor os recursos de um ambiente administrativo, e com as outras duas ferramentas foi possível explorar os recursos do website.

O terceiro objetivo, visava listar os requisitos funcionais, não funcionais e as regras de negócio do protótipo. Este objetivo foi cumprido com a seção de análise, onde foram apresentados os requisitos funcionais, não funcionais e as regras de negócio, tanto do *website*, quanto do ambiente administrativo.

O quarto e último objetivo, consistia em empregar no desenvolvimento do protótipo as tecnologias escolhidas. O objetivo foi cumprido e apresentado na seção de implementação, onde foram detalhadas como as tecnologias selecionadas foram utilizadas, bem como, foram

demonstrados todos os requisitos desenvolvidos no protótipo, contendo a explicação de cada uma delas.

Com a utilização do protótipo desenvolvido as lojas de veículos, especialmente aquelas que ainda não possuem um sistema automatizado de gestão como a Sálvio Automóveis, poderão melhorar seu controle e administração dos veículos em suas lojas. O protótipo permite que os lojistas registrem, atualizem e gerenciem informações dos seus veículos de forma centralizada, proporcionando uma melhor organização e otimização dos processos operacionais. A funcionalidade de consulta automática à tabela FIPE integrada ao protótipo facilita ainda mais a vida dos usuários, pois elimina a necessidade de acessar o site da FIPE separadamente para verificar o valor atual dos veículos, economizando tempo e esforço.

Por fim, embora todos os objetivos específicos definidos inicialmente para o protótipo tenham sido alcançados com sucesso, assim como o objetivo geral, proporcionando uma solução prática para o gerenciamento de lojas de veículos, durante o desenvolvimento novas ideias surgiram, mas devido às limitações de escopo e tempo pré-definidos, não puderam ser implementadas. No entanto, essas ideias são apresentadas como sugestões para futuros aprimoramentos e expansões do protótipo, visando torná-lo mais completo e robusto.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Para o aperfeiçoamento e continuidade do protótipo, algumas funcionalidades foram identificadas para serem implementadas em futuras versões. A primeira recomendação é o desenvolvimento de um aplicativo móvel específico para o ambiente administrativo, que não dependeria de um navegador para acesso, proporcionando uma experiência otimizada e fluída para o usuário. Com isso, lojistas teriam maior autonomia no gerenciamento dos veículos, podendo acessar o ambiente de administração diretamente de seus *smartphones* ou *tablets*.

A segunda recomendação seria a implementação de envios de *e-mails* automáticos, que notificaria o lojista sempre que uma nova solicitação de contato fosse recebida pelo *website*. Essa funcionalidade tornaria a comunicação com os clientes mais ágil e eficiente, permitindo um atendimento mais rápido às solicitações.

A terceira recomendação seria a criação de uma rotina de configurações dentro do ambiente administrativo. Essa funcionalidade permitiria, por exemplo, que o lojista configurasse a exibição do preço FIPE dos veículos no website. Com essa opção, o preço da tabela FIPE só seria mostrado ao cliente final na parte do detalhamento do veículo caso a opção

estivesse marcada, garantindo maior flexibilidade para o lojista sobre o que deseja apresentar ou não, de acordo com suas preferências e necessidades.

Adicionalmente, propõe-se a inclusão de funcionalidades para o registro e controle de despesas relacionadas aos veículos, tais como despesas gerais, preço de compra e preço de venda. Esse recurso seria um facilitador aos lojistas, para que pudessem acompanhar de forma detalhada e eficiente os custos associados a cada veículo, contribuindo para uma gestão financeira mais precisa.

Outra melhoria sugerida é a implementação de uma funcionalidade que permita o acompanhamento do tempo entre a entrada (data de aquisição) e a saída (venda) dos veículos no sistema. Essa métrica seria útil para avaliar a eficiência da loja de converter o estoque em vendas, identificando os veículos que permanecem mais tempo no pátio. Quanto menor o tempo médio entre entrada e saída, melhor será o desempenho do negócio, permitindo que os lojistas tomem decisões estratégicas para acelerar o ciclo de vendas.

Além disso, recomenda-se a realização de um processo de validação e homologação do protótipo com a Sálvio Automóveis, de modo a garantir que todas as funcionalidades implementadas estejam atendendo as necessidades do cliente. Esse processo poderia ser realizado com a participação de outros lojistas em ambientes reais ou através de testes simulados, a fim de identificar possíveis melhorias antes da utilização de fato. Essas implementações sugeridas visam otimizar ainda mais o sistema, tornando-o mais robusto e adequado às necessidades operacionais das lojas de veículos.

REFERÊNCIAS

ABRAHAM, Nikhil. **Codificação Para Leigos**: Os primeiros passos para o sucesso. Editora Alta Books, 2019. E-book.

ALTIMUS. **Altimus**. Disponível em: https://altimus.com.br. Acesso em: 31 ago. 2024.

ALVES, William P. **Banco de Dados**. Editora Saraiva, 2014. E-book.

ALVES, William P. **HTML & *CSS***: aprenda como construir páginas web. Editora Saraiva, 2021. E-book.

AUTOCONF. **Autoconf**. Disponível em: https://autoconf.com.br. Acesso em: 31 ago. 2024.

BORBA CAR. **Borba Car**. Disponível em: https://borbacar.com.br. Acesso em: 31 ago. 2024.

DEITEL, Paulo; DEITEL, Harvey; WALD, Alexandre. **Android 6 para programadores**. Grupo A, 2016. E-book.

DUCKETT, Jon. **PHP&MYSQL**: desenvolvimento web no lado do servidor. Editora Alta Books, 2024. E-book.

ENCONTRE VEÍCULOS. **Encontre Veículos**. Disponível em: https://www.encontreveiculos.com.br. Acesso em: 31 ago. 2024.

FIPE API. **FIPE API.** Disponível em: https://fipeapi.com.br/. Acesso em 01 set. 2024.

FLANAGAN, David. **JavaScript**: o guia definitivo. Porto Alegre Bookman 2014. Ebook.

FREITAS, Pedro Henrique C.; BIRNFELD, Karine; SARAIVA, Maurício de O.; MARIANO, Diego César Batista; SILVA, Fabricio Machado. **Programação Back End III**. Grupo A, 2021. E-book.

HEUSER, Carlos A. **Projeto de banco de dados** - UFRGS. V.4. Grupo A, 2011. E-book.

HIRAMA, Kechi. **Engenharia de Software**. Rio de Janeiro: GEN LTC, 2011. E-book. p.VIII. ISBN 9788595155404. E-book.

IBGE (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA). **API de Localidades**. Disponível em: https://servicodados.ibge.gov.br/api/docs/localidades. Acesso em: 08 de nov. 2024.

IBM. **O que é PostgreSQL?** Disponível em: https://www.ibm.com/br-pt/topics/postgresql. Acesso em: 24 de abril de 2024.

LOBO, Edson JR. **Criação de sites em PHP**. Rio de Janeiro: Brasport, 2007.

MACHADO, Felipe Nery R. **Banco de Dados:** Projeto e implementação. SRV Editora LTDA, 2020. E-book.

MDN. **Glossário**. 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary. Acesso em: 24 de abril de 2024.

MILETTO, Evandro M.; BERTAGNOLLI, Silvia C. **Desenvolvimento de software II**: Introdução ao desenvolvimento web com HTML, *CSS*, javascript e PHP. (Tekne). Grupo A, 2014. E-book.

OLIVEIRA, Cláudio Luís V.; ZANETTI, Humberto Augusto P. **JavaScript Descomplicado:** Programação para web, IOT e dispositivos móveis. Editora Saraiva, 2020. E-book.

PAULA FILHO, Wilson de Pádua P. **Engenharia de Software – Produtos**: Vol.1. Grupo GEN, 2019. E-book.

PINHO, Diego Martins de. **Alertas bonitos, responsivos e customizados com o SweetAlert2.** 2018. Disponível em: https://medium.com/code-prestige/alertas-bonitos-responsivos-ecustomizados-com-o-sweetalert2-8db930038137. Acesso em: 09 de nov.2024.

POSTGRESQL. 2024a. **A Brief History of PostgreSQL**. Disponível em: https://www.postgresql.org/docs/current/history.html. Acesso em: 01 de abril de 2024.

POSTGRESQL. 2024b. **PostgreSql Documentation**. Disponível em: https://www.postgresql.org/docs/. Acesso em: 01 de abril de 2024.

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

PRESSMAN, Roger S. **Engenharia de software**: Uma abordagem profissional. Porto Alegre: AMGH, 2021. Ebook.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software**. Grupo A, 2021. E-book.

ROBBINS, Jennifer Niederst. **Learning Web Design**. Sebastopol: O'REILLY MEDIA, 2018. Ebook.

RODRIGUES, Thiago Nascimento; SILVA, Lídia Patrícia Cruz; NEUMANN, Fabiano Berlinck; LEOPOLDINO, Fabrício Leonard; SANTOS, Marcelo da Silva dos; TAVARES, Jenifer Vieira Toledo; BEZERRA, Whesslley Rimar. **Integração de aplicações**. Porto Alegre: Grupo A, 2020. Ebook.

SILVA, Luiz FC.; RIVA, Aline D.; ROSA, Gabriel A.; e outros. **Banco de Dados Não Relacional**. Grupo A, 2021. E-book.

SOMMERVILLE, Iam. **Engenharia de software**. São Paulo: Pearson Prentice Hall, 2011. Ebook.

TURINI, Rodrigo. **PHP e Laravel**: crie aplicações web como um verdadeiro artesão. [S. l.]: Casa do Código, 2015.

VETORAZZO, Adriana S. **Engenharia de software**. Grupo A, 2018. E-book.

ZABOT, Diego; MATOS, Ecivaldo de S. **Aplicativos com Bootstrap e Angular**: Como desenvolver apps responsivos. Editora Saraiva, 2020. E-book.

ANEXOS

ANEXO I – TERMO DE AUTORIZAÇÃO PARA USO DE NOME EMPRESARIAL E DADOS

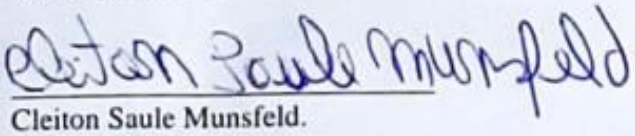
CENTRO UNIVERSITÁRIO PARA DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ

TERMO DE AUTORIZAÇÃO PARA USO DE NOME EMPRESARIAL E DADOS

Ilmo. Sr (a): Cleiton Saule Munsfeld.
Rio do Sul, 26 de outubro de 2023.

Eu, Micael Munsfeld matriculado no curso de Sistemas de Informação, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí com a orientação do professor M.e Jullian Hermann Creutzberg, venho solicitar a V. Sa. A autorização para uso do nome da empresa "Sálvio Munfeld" e eventual coleta de dados com a finalidade de levantamento de requisitos e desenvolvimento de Trabalho de Conclusão de Curso do no curso de Sistemas de Informação, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí com o titulo "MCC: PROTÓTIPO DE SOFTWARE WEB PARA GERENCIAR LOJAS DE VEÍCULOS". O nome da empresa será citado no decorrer do mesmo, como potencial público-alvo da implantação do protótipo. Assumo o compromisso de utilizar os dados obtidos somente para fins acadêmicos e científicos, bem como de compartilhar os resultados obtidos ao final da pesquisa. Agradecemos antecipadamente e esperamos contar com a sua colaboração.

Atenciosamente,
Micael Munsfeld



Cleiton Saule Munsfeld.
Responsável.
Sálvio Automóveis.