# CarView: Criação de uma aplicação para registro de manutenção de veículos

**Matheus A. L. Firmiano, André C. da Silva**

Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia – SP – Brasil

m.firmiano@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

***Abstract.*** *In this work, an application focused on vehicle maintenance was developed, providing users with a simple and robust solution to manage their vehicles' condition. The application allows users to record and monitor maintenance activities, from oil changes to major inspections, and sends automatic reminders to ensure that maintenance is performed periodically. Additionally, the application provides detailed information on average prices for parts and services, helping users to perform maintenance more efficiently.*

***Resumo.*** *Neste trabalho de conclusão de curso, foi desenvolvido uma aplicação voltada para a manutenção de veículos, que fornece aos usuários uma solução simples e robusta para gerenciar o estado de seus veículos. A aplicação permite que usuários registrem e monitorem as atividades de manutenção, desde trocas de óleo até revisões importantes, além de enviar lembretes automáticos com o intuito de garantir que as revisões sejam feitas periodicamente. Além disso, a aplicação fornece informações detalhadas sobre os preços médios de peças e serviços, auxiliando os usuários a realizar manutenções de forma mais eficiente.*

## 1. Introdução

Existem, no Brasil, 464 milhões de computadores, notebooks, *smartphones* e *tablets* (segundo pesquisa da FGV em 2023). Para desenvolver para múltiplos dispositivos há diversas soluções, desde a codificação da aplicação em diferentes linguagens de programação (chamado de código nativo) ao uso de *frameworks* para gerar código para diferentes plataformas (Silva e Prado, 2019).

Adicionalmente, é possível constatar que a manutenção preventiva de veículos apresenta uma relação custo-benefício mais favorável quando comparada à abordagem corretiva (Otani e Machado, 2008). Diante dessa consideração, aliada às ações empreendidas pelos órgãos estatais (Portal Terra, 2017), no sentido de promover a conscientização entre os condutores sobre a relevância da manutenção preventiva como garantia de um tráfego mais seguro, diversas cidades no Brasil elegeram o mês de junho como o "Mês da Conscientização em Prol da Manutenção Preventiva de Automóveis" (Portal Terra, 2017). É perceptível que existe uma certa dificuldade entre os proprietários de veículos em determinar o momento adequado para realizar as manutenções em seus automóveis, pois além de terem de se lembrar da última revisão em determinada parte ou peça, cada parte ou peça pode ter um tempo de vida útil diferente.

Diante desse cenário nota-se que uma aplicação que auxilia o motorista/proprietário a acompanhar o estado em que o veículo se encontra, avise-o de tempos em tempos o que seria

necessário trocar e informá-lo a média do valor que será empenhado para realizar as manutenções necessárias é de grande importância.

O objetivo do projeto é desenvolver um aplicativo PWA (*Progressive Web App*) para auxiliar usuários a registrar a manutenção realizada em seu veículo, receber avisos para realizar manutenções e a consultar a média do preço gasto nas peças pelos demais usuários do aplicativo. Pretende-se, com esse trabalho, proporcionar uma boa experiência para os usuários, seja em um dispositivo portátil móvel (*smartphones* e *tablets*) ou em computadores (*desktops* e *notebooks*). Este trabalho optou pela solução PWA, que utiliza tecnologias da *Web* para dispor uma aplicação em execução em diferentes dispositivos mas, de um modo geral, se assemelhando muito a um aplicativo do dispositivo.

Na Seção 2 é abordado o referencial teórico, apresentando os conceitos que foram utilizados neste trabalho, enquanto que na Seção 3 são apresentados os trabalhos semelhantes ao que esse estudo propõe. Na Seção 4 são abordados a arquitetura que foi elaborada no projeto junto com suas tecnologias e a abordagem do processo de desenvolvimento da aplicação. Na Seção 5 são descritas as etapas do desenvolvimento deste trabalho, seus protótipos e suas funcionalidades. Na Seção 6 é resgatado o objeto principal do trabalho e sua contextualização, os objetivos alcançados, quais as limitações e apontamentos de trabalhos futuros, além de abordar quais disciplinas do curso foram articuladas.

## 2. Referencial Teórico

O objetivo desta seção é apresentar os assuntos de características de aplicações *Web* considerando o cenário de múltiplos dispositivos, o desenvolvimento de aplicações nesse contexto, dando ênfase no PWA.

### 2.1. Características de Aplicações Web

As aplicações web podem ser acessadas de qualquer dispositivo com conexão à *internet* e navegador *web*, permitindo que os usuários as utilizem de qualquer lugar do mundo. Além disso, é necessário projetar interfaces de usuário fáceis de usar e que proporcionam uma experiência agradável aos usuários. Uma característica importante é a capacidade de fornecer informações e atualizações em tempo real, permitindo que os usuários vejam dados e conteúdos em tempo real, sem a necessidade de recarregar a página (Vasconcelos; Brilhante; Lemos, 2020).

Estas aplicações possibilitam interações bidirecionais entre o usuário e o sistema, permitindo atividades como envio de dados e preenchimento de formulários, com respostas imediatas da aplicação (Fêo; da Silva; do Amaral, 2016). Além disso, elas podem se integrar facilmente a diferentes sistemas e serviços por meio de *APIs* (Interfaces de Programação de Aplicações), permitindo a troca eficiente de dados e funcionalidades entre diversas plataformas (Vasconcelos; Brilhante; Lemos, 2020).

A troca de dados via *APIs* ocorre através de *endpoints*, que são pontos finais de comunicação em um sistema de *software*, geralmente *URLs*. Os clientes enviam solicitações e recebem respostas, permitindo interações com serviços da *web*, acesso a recursos ou execução de operações específicas em sistemas distribuídos (GEEWAX, 2021). Esses serviços podem ser empacotados e distribuídos usando tecnologias como o Docker, facilitando sua implantação e gerenciamento em diferentes ambientes de desenvolvimento, teste e produção.

As aplicações *web* também têm a vantagem de armazenar dados na nuvem, garantindo fácil acesso a partir de qualquer dispositivo e evitando perda de dados em caso de falha do dispositivo do usuário (AWS). Além disso, são altamente escaláveis, podendo lidar com um grande número de usuários e se adaptar à demanda sem a necessidade de grandes modificações na infraestrutura (Marques, 2022).

## 2.2. Desenvolvimento de Aplicações para a *Web* e *Mobile*

Existe uma imensidão de tecnologias que podem ser utilizadas para a criação de aplicações *web* e dispositivos móveis (Dzhangarov; Pakhaev; Potapova, 2021), variando conforme o tipo de aplicação e como será utilizada.

Aplicações *web* são divididas em duas partes, o *front-end*, onde o usuário interage com a aplicação, e o *back-end*, onde fica toda a lógica e regra de negócio, ele também processa as solicitações vindas dos *endpoints*, realiza operações e retorna as respostas. O básico para se criar um *front-end* de uma aplicação *web* são *HTML*, *CSS* e *JavaScript*. No lado do *back-end*, utilizam-se linguagens como *Ruby*, *Java* e *Node.js*, assim como bancos de dados relacionais e não relacionais para armazenamento de dados. Para que a aplicação possa ficar acessível para os usuários, ela deve ser hospedada em servidores *web* e vinculada à um domínio para permitir o acesso via navegador (Andrei, 2024).

No universo dos aplicativos móveis existem diferentes sistemas operacionais, os mais populares são *Android* e *iOS* (Silva; Prado, 2019), o desenvolvimento para aplicativos Android é feito com *Java* ou *Kotlin* por exemplo, já para *iOS* é feito com Swift ou *Objective-C*. Além disso, existem frameworks inovadores como o Xamarin, que possibilitam o desenvolvimento para as duas plataformas utilizando um único código-base (*Microsoft*, 2023). Cada plataforma móvel possui suas próprias diretrizes de design, como o *Material Design* no *Android* e o *Human Interface Guidelines* (HIG) no *iOS*. As interfaces do usuário são desenvolvidas usando componentes nativos ou personalizados, adaptando-se às necessidades específicas de cada aplicativo. Após o desenvolvimento, os aplicativos são distribuídos por meio da *Google Play Store* para *Android* e da *App Store* para *iOS* que são lojas onde os usuários podem encontrar aplicativos de forma segura e prática.

### 2.2.1. *Progressive Web App*

Um *Progressive Web App* (PWA) é uma tecnologia híbrida que combina as características de sites e aplicativos móveis para oferecer uma experiência de usuário otimizada e envolvente (Mazzarolo; Silva, 2021); a Figura 1 ilustra essa ideia. PWAs são aplicações *web* desenvolvidas com tecnologias da *web* padrão, mas com funcionalidades adicionais que anteriormente eram exclusivas de aplicativos nativos, como por exemplo acessar a câmera, contatos e a localização do dispositivo móvel.

As principais características dos PWAs incluem: Acessibilidade *Offline* sendo capaz de funcionar *offline* ou em condições de rede instáveis, permitindo que os usuários acessem conteúdo mesmo sem conexão à *internet*; Responsividade sendo capazes de funcionar em qualquer dispositivo ou tamanho de tela, proporcionando uma experiência de usuário consistente em *desktops*, *laptops*, *smartphones* e *tablets*; Atualizações Automáticas: PWAs são atualizados automaticamente, garantindo que os usuários sempre tenham acesso à versão mais recente da aplicação, sem a necessidade de baixar e instalar atualizações manualmente (Sanitá, 2020); Interação com o Usuário sendo capazes de enviar notificações *push*, acessar a câmera, microfone, GPS e outras funcionalidades do dispositivo, proporcionando uma

experiência interativa semelhante à de aplicativos nativos; Segurança sendo servidos através de HTTPS, garantindo a segurança dos dados do usuário e protegendo contra ataques cibernéticos (Mazzarolo; Silva, 2021). Essas características fazem dos PWAs uma opção popular para empresas que desejam oferecer uma experiência de aplicativo aos usuários sem os inconvenientes das lojas de aplicativos tradicionais. PWAs são especialmente úteis em regiões com conexões de internet instáveis, onde a acessibilidade *offline* é crucial.

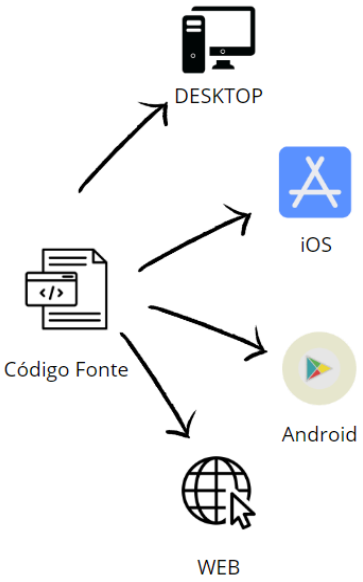



**Figura 1. Código fonte único multiplataformas.**

## 3. Trabalhos Correlatos

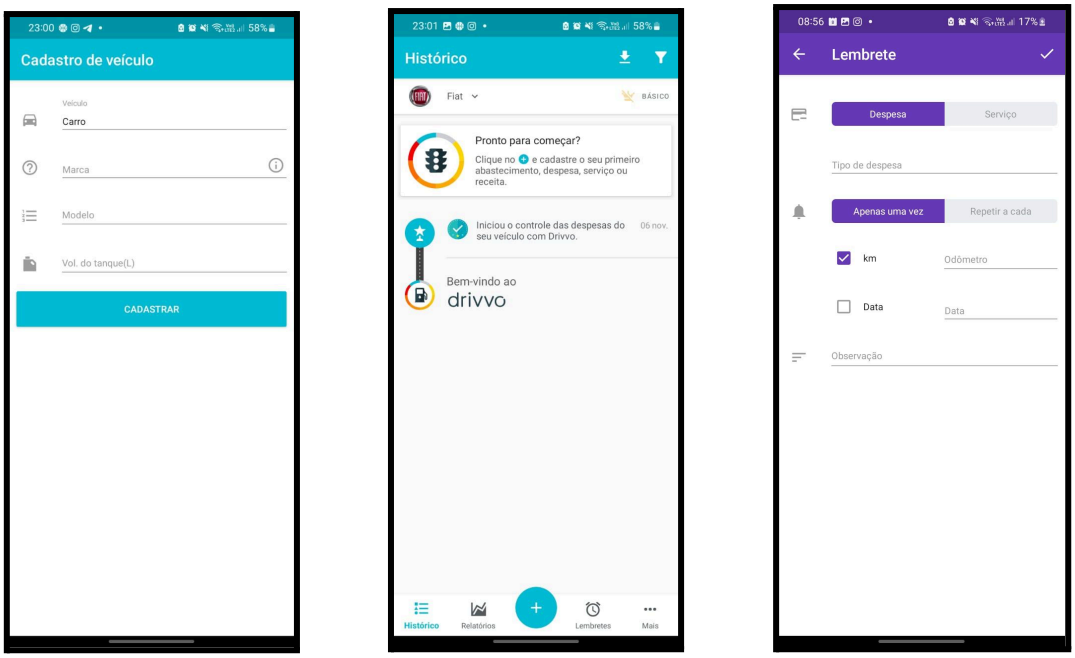
Esta Seção tem como finalidade apresentar as principais funcionalidades de trabalhos semelhantes, que serviram como inspiração para as funcionalidades da aplicação desenvolvida neste trabalho.

### 3.1. Drivvo

Drivvo (Figura 2) é um aplicativo para para gerenciar veículos, os gastos com combustível e com peças trocadas exibidos na tela principal, criação de lembretes. Possui uma versão gratuita do aplicativo para uso pessoal. Ele possibilita que você tenha acesso sobre os custos com o veículo, tais como abastecimentos e manutenções. É um aplicativo gratuito, porém possui alguns planos pagos com alguns benefícios. Está disponível para dispositivos móveis *Android* e *IOS* apenas.

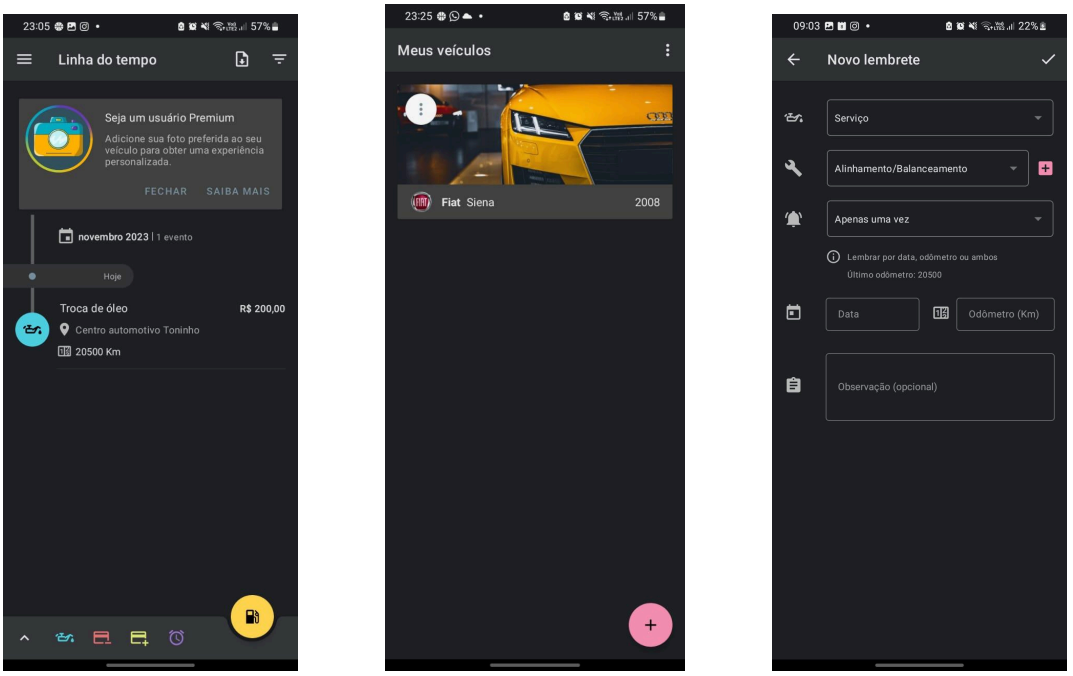
### 3.2. Carango

Carango (Figura 3) é um aplicativo para gerenciar veículos disponibilizado apenas para *Android*. possui um gerenciamento financeiro para veículos, com ele é possível registrar receitas e despesas, criar lembretes manuais de despesas e usar a calculadora *flex* para saber qual combustível colocar de acordo com a diferença do preço entre os combustíveis. É um aplicativo gratuito, porém possui alguns planos pagos com alguns benefícios. Está disponível para dispositivos móveis *Android* e *IOS* apenas.




(a) (b) (c)

**Figura 2. Telas do aplicativo Drivvo para (a) cadastro de veículos, (b) tela principal e (c) criação de lembretes.**

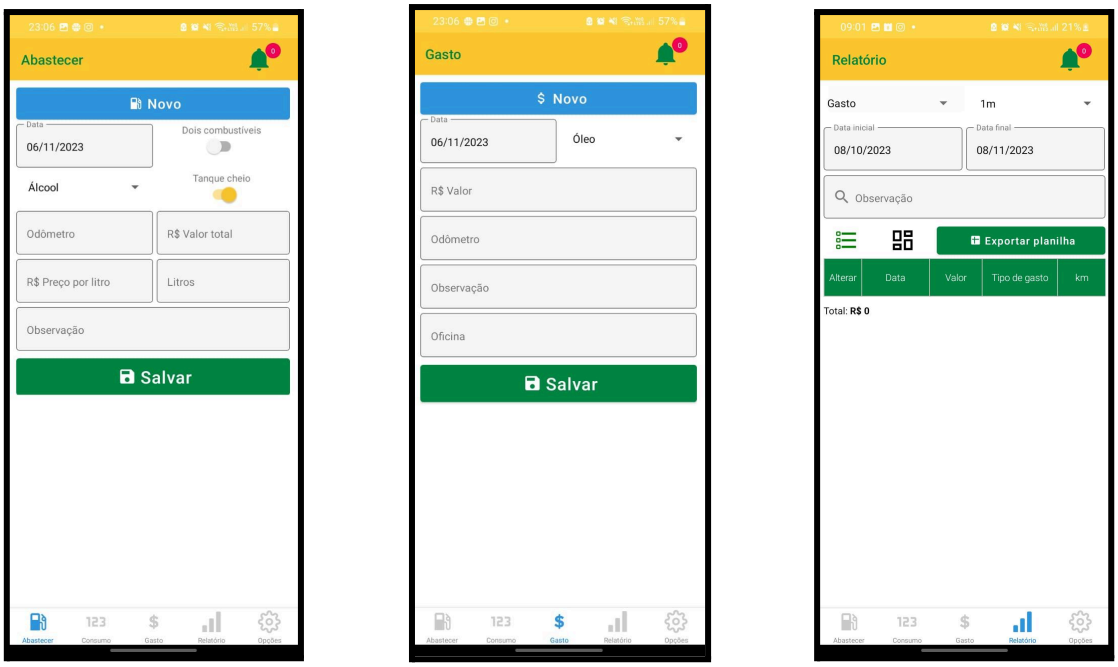



(a) (b) (c)

**Figura 3. Telas do aplicativo Carango para (a) cadastro de despesa, (b) tela inicial e (c) criação de lembretes.**

## 3.3. Meu Possante

Meu Possante (Figura 4) é um aplicativo para gerenciar despesas de veículos, saber o quanto é gasto em combustível. É possível adicionar lembretes para trocas de óleo, revisões e manutenção. É um aplicativo gratuito, porém possui alguns planos pagos com alguns benefícios. Está disponível para dispositivos móveis *Android* e *IOS* apenas.




(a) (b) (c)

**Figura 4. Telas do aplicativo Meu Possante. (a) registro de abastecimento, (b) cadastro de despesa e (c) relatórios.**

### 3.4. Comparação dos Trabalhos Correlatos

Os três aplicativos possuem a mesma solução implementada de formas diferentes, todos os três possuem gráficos de gestão de gastos, relatório de abastecimento, gestão do veículo e histórico de gastos, porém nenhum deles envia notificações sobre as manutenções próximas da validade de forma automática sem a necessidade do usuário ficar criando lembretes, e tampouco exibem o preço médio gasto em cada manutenção ou troca de peça (Figura 5). Dentre os três trabalhos correlatos, o Carango se destaca por mostrar os postos de gasolina nas proximidades.

| Funcionalidade/App | Drivvo | Carango | Meu Possante | Carview |
|---|---|---|---|---|
| Grafico de gestão de gastos | ✓ | ✓ | ✓ | ✗ |
| Relatório de abastecimento | ✓ | ✓ | ✓ | ✗ |
| Mapa com postos de combustíveis próximos | ✗ | ✓ | ✗ | ✗ |
| Histórico de gastos | ✓ | ✓ | ✓ | ✓ |
| Valor médio gastos pelos usuários em determinada manutenção/peça | ✗ | ✗ | ✗ | ✓ |
| Notificar manutenções necessárias | ✗ | ✗ | ✗ | ✓ |
| Compatibilidade | ✓ | ✗ | ✗ | ✓ |
| Segurança de acesso | ✓ | ✓ | ✓ | ✓ |

**Figura 5. Comparação das funcionalidades dos trabalhos correlatos.**

## 4. Metodologia

Inicialmente, para compreensão dos requisitos de um *software* no contexto de registro de manutenções veiculares, será realizada uma pesquisa para identificar as dificuldades de registrar as manutenções realizadas nos veículos. Também serão realizadas análises de *softwares* com o mesmo propósito. A partir dessa pesquisa será feito uma análise para identificar as necessidades essenciais e direcionar os esforços para resolvê-las. Os requisitos identificados serão expostos por meio de histórias de usuário.

Após finalizar a análise, serão identificadas as entidades do sistema que serão traduzidas para tabelas. Para armazenamento dos dados, optamos pelo SGBD de código aberto *PostgreSQL*, conhecido por sua robustez, flexibilidade e licença de código aberto (*PostgreSQL*). Além disso, é uma ferramenta madura e estável, respaldada por uma comunidade ativa que contribui para sua evolução.

O *back-end* será desenvolvido por meio de uma abordagem incremental e abordagem de desenvolvimento de *API*, criando *endpoints* eficientes para comunicação entre *front-end* e *back-end*, permitindo flexibilidade e escalabilidade (Marques, 2022). Como abordagem de desenvolvimento, será empregada a abordagem incremental, sendo o *back-end* desenvolvido em 4 iterações e o *front-end* em 3 iterações. O projeto será construído na arquitetura monolítica, onde o *back-end* e *front-end* estão dentro do mesmo serviço, isso permite que a aplicação seja desenvolvida de maneira mais rápida e menos complexa.

O *front-end* será desenvolvido a partir do *framework* Vaadin, pois esse *framework* é capaz de criar aplicações PWA sem a necessidade de realizar muitas configurações, além de proporcionar uso semelhante a aplicativos nativos, com responsividade para diferentes dispositivos. Desenvolver para múltiplos dispositivos foi uma prioridade, com foco na compatibilidade com vários dispositivos. Essa abordagem busca adaptar interfaces eficazmente a variados tamanhos de tela e sistemas operacionais, garantindo consistência na experiência do usuário por meio de uma solução responsiva. Serão empregadas as tecnologias *HTML, CSS, JavaScript, Java, Docker, Ngrok*, além de *SQL* para interação com o banco de dados.

## 5. Desenvolvimento

Esta Seção tem como objetivo descrever as etapas do desenvolvimento deste trabalho, os protótipos criados e as funcionalidades.

### 5.1. Análise Inicial e Coleta de Dados sobre Manutenção Veicular

Na elaboração deste TCC, primeiramente realizamos uma análise da situação atual, expondo as dificuldades dos proprietários de veículos em centralizar informações de manutenção. Para isso, implementamos um questionário no *WhatsApp* (Figura 6) como parte do processo de coleta de dados. Esse questionário foi muito importante para entender quais funcionalidades seriam cruciais para criar uma solução diferente das já existentes no mercado.

Um fator importante notado neste questionário é a falta de tempo para se registrar as manutenções, portanto a aplicação deve ser simples de se usar, sem a necessidade de realizar configurações complexas e que demandam muito tempo. Outro ponto importante é que o público em geral não tem muito conhecimento sobre como registrar as manutenções, isso reforça a ideia de que essa deve ser uma solução de uso simples para melhor atendê-los.

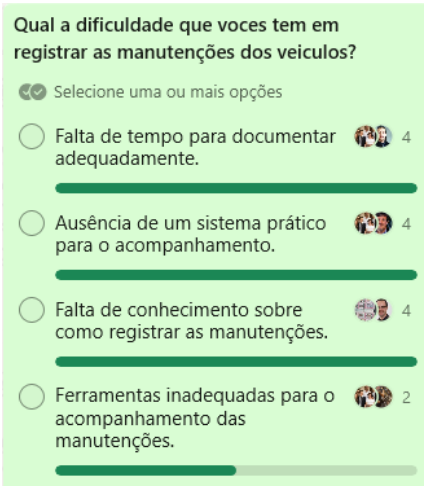



**Figura 6. Questionário referente a dificuldades ao registrar manutenções em veículos.**

## 5.2. Requisitos Funcionais e Não Funcionais

Os requisitos funcionais e não funcionais do sistema foram identificados ao analisar os trabalhos correlatos, e também foram baseados nas minhas próprias experiências e desejos pessoais, refletindo o que considero essencial para um funcionamento eficaz. Para isso, realizei uma auto análise das necessidades que surgem em situações práticas, imaginando como o sistema poderia atender a essas demandas.

Os requisitos funcionais foram delineados a partir de cenários que vivi e que poderiam ser otimizados por meio da tecnologia, enquanto os requisitos não funcionais foram moldados por considerações sobre a usabilidade, desempenho e segurança que considero importantes. Essa abordagem pessoal e reflexiva me permitiu criar uma visão clara do que o sistema deve oferecer, alinhando minhas expectativas às funcionalidades necessárias para uma experiência satisfatória.

Os requisitos funcionais do sistema incluem o cadastro de usuários, que deve permitir o registro com *e-mail* e senha únicos, além de autenticação durante o *login*. Os usuários poderão cadastrar veículos informando dados como marca, modelo, ano e placa, com cada veículo sendo associado ao usuário que o cadastrou. O sistema deve permitir a inserção, edição e exclusão de manutenções realizadas nos veículos, bem como a visualização da lista de veículos e suas manutenções, exibindo detalhes de cada registro. Também é necessário fornecer mensagens de confirmação após ações como cadastro, edição e exclusão, além de solicitar a confirmação do usuário antes de excluir um veículo ou manutenção. O sistema deverá solicitar semanalmente ao usuário a atualização da quilometragem dos veículos cadastrados.

Em relação aos requisitos não funcionais, como essa é uma aplicação PWA, ela deve ser suportada em múltiplos dispositivos; o desempenho do sistema deve garantir tempos de resposta adequados para operações como cadastro, edição e consulta, proporcionando uma experiência fluida. A interface deve ser intuitiva e fácil de navegar, com formulários claros e validação de dados em tempo real. Além disso, a segurança das informações dos usuários deve ser assegurada por meio de criptografia, restringindo o acesso ao sistema a usuários autenticados. A arquitetura do sistema deve ser escalável, suportando um aumento no número de usuários e veículos cadastrados sem perda de *performance*, e deve ser compatível com os principais navegadores *web* e dispositivos móveis.

### 5.3. Histórias de Usuário

Após levantar os requisitos funcionais e não funcionais, foram criadas algumas histórias de usuário para guiar o desenvolvimento da solução. Devido às limitações deste TCC, foram registradas apenas algumas histórias de usuário, a seguir, dois exemplos.

**Cenário:** Como usuário do aplicativo CarView, gostaria de poder cadastrar as manutenções do meu veículo de forma rápida e eficiente. Para alcançar isso, desejo iniciar o processo cadastrando o meu veículo no aplicativo.

1. Abertura do *App*:
    - Ao fazer *login* no aplicativo, sou direcionado para a tela inicial.
2. Cadastro do Veículo:
    - Navego até a seção de "Veículos" pelo menu lateral.
    - Encontro a opção para adicionar um novo veículo e seleciono essa opção.
    - Sou apresentado a um formulário onde posso inserir as informações do meu veículo.
    - Depois de preencher os campos obrigatórios, confirmo o cadastro do veículo.
3. Registro de Manutenção:
    - Retorno à tela inicial e acesso a opção de "Manutenções" do veículo cadastrado.
4. Detalhes da Manutenção:
    - Preencho os detalhes da manutenção realizada.
5. Confirmação e Registro:
    - Reviso todas as informações inseridas para garantir precisão.
    - Ao confirmar, a manutenção é registrada no sistema e associada ao veículo selecionado.

Critérios de Aceitação:

- O aplicativo deve permitir o cadastro fácil e intuitivo de um veículo.
- O usuário deve ser capaz de adicionar uma nova manutenção ao veículo selecionado.
- Todos os detalhes relevantes da manutenção devem ser registrados com precisão.
- O sistema deve garantir a associação correta do veículo ao usuário específico.
- O sistema deve garantir a associação correta da manutenção ao veículo específico.

**Cenário:** Como usuário do aplicativo CarView, desejo ter a capacidade de editar as informações das manutenções do meu veículo.

1. Acesso ao Aplicativo:
    - Ao acessar o aplicativo CarViewe fazer *login*, sou levado diretamente para a tela inicial.
2. Seleção do Veículo para Edição:
    - Navego até a seção de "Veículos" usando o menu lateral.
    - escolho o veículo cuja manutenção desejo editar.
3. Acesso às Manutenções:
    - Na página do veículo selecionado, toco na opção "Manutenções" para visualizar todas as manutenções associadas a esse veículo.
4. Seleção da Manutenção para Edição:
    - Encontro a manutenção que quero editar na lista e toco nela para abrir os detalhes.

    - Na página de detalhes da manutenção, encontro o botão "Editar" e toco nele.
5. Edição dos Detalhes da Manutenção:
    - Sou apresentado a um formulário pré-preenchido com as informações da manutenção.
    - Faço as alterações necessárias nos campos que desejo editar, como tipo de serviço, data, custo, etc.
6. Confirmação e Atualização:
    - Após revisar cuidadosamente as alterações feitas, toco no botão "Confirmar" para salvar as edições.
    - As informações atualizadas são registradas no sistema e associadas à manutenção do veículo selecionado.
7. Confirmação e Atualização:
    - Após revisar cuidadosamente as alterações feitas, toco no botão "Confirmar" para salvar as edições.
    - As informações atualizadas são registradas no sistema e associadas à manutenção do veículo selecionado.

Critérios de Aceitação:

- O aplicativo deve permitir o cadastro fácil e intuitivo de um veículo.
- O usuário deve ser capaz de adicionar uma nova manutenção ao veículo selecionado.
- Todos os detalhes relevantes da manutenção devem ser registrados com precisão.
- O sistema deve garantir a associação correta do veículo ao usuário específico.
- O sistema deve garantir a associação correta da manutenção ao veículo específico.

### 5.4. Elaboração do Projeto da Base de Dados

Com base na análise dos requisitos, identificamos as principais entidades, seus atributos e os tipos de relacionamentos (Figura 7), o que permitiu compreender e definir os requisitos de dados da aplicação. Em seguida, foram estabelecidas as estruturas das tabelas, os tipos de dados e suas restrições. As principais entidades do sistema são Usuários, Carros e Manutenções.

- Usuários: Essa entidade possui como principais atributos o nome, email, usuário e senha, foi decidido persistir as credenciais de acesso junto ao perfil do usuário para não deixar o sistema mais complexo.
- Carro: Essa entidade possui como principais atributos o modelo, fabricante, ano de fabricação e *id* do usuário .
- Manutenções: Essa entidade possui como principais atributos nome da peça, fabricante, estado da peça, preço gasto, vida útil*, id* do veículo e a quilometragem ideal para se trocar a peça.

### 5.5. Configuração do ambiente do projeto de desenvolvimento

Durante essa fase de desenvolvimento, utilizamos um conjunto de ferramentas essenciais para concluir o trabalho. Para a codificação, foi empregado a *IDE IntelliJ*, que facilitou e auxiliou no processo de desenvolvimento. Para garantir o rastreamento das alterações e a segurança no armazenamento dos repositórios, foi utilizado o *GitHub*. Além disso, o *Maven* foi escolhido para uma gestão eficaz das dependências do projeto. Esta ferramenta permite gerenciar todas as bibliotecas e componentes necessários, assegurando a integração e estabilidade das diferentes partes do sistema. Foi utilizado a ferramenta *Docker Desktop* para criar um

*container* para armazenar o banco de dados, facilitando o processo de conexão com o banco e manipulação do mesmo caso seja preciso recriá-lo. Para desenvolver uma PWA foi utilizado o *framework* Vaadin, um *framework* para *Java* e *Spring Boot* que realiza todas as configurações para a aplicação ter as características de uma PWA.

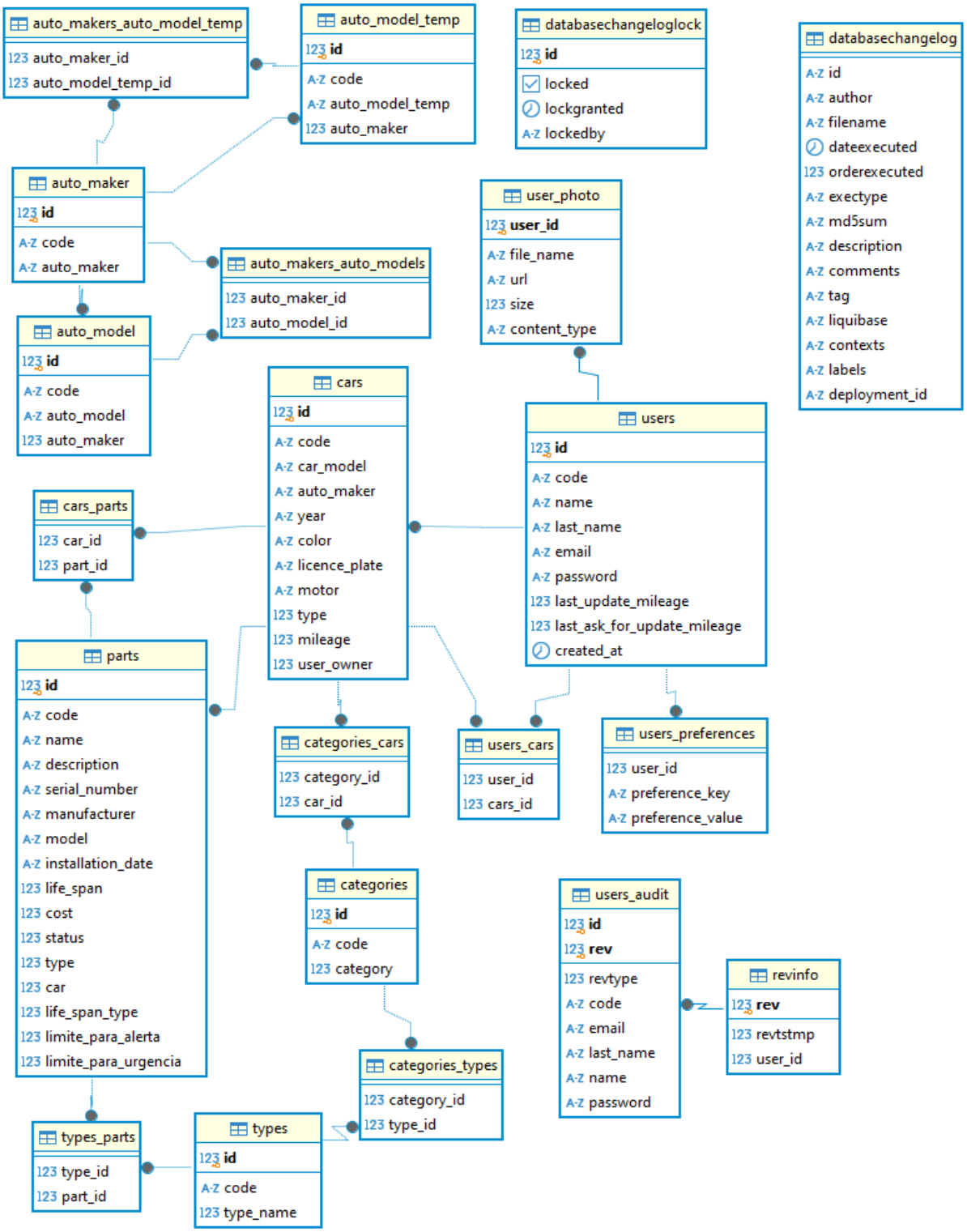



**Figura 7. Diagrama da base de dados para a aplicação CarView.**

## 5.6 Fluxo da aplicação

O fluxo das janelas da aplicação é apresentado na Figura 8. A aplicação inicia na página de *login* onde é feita a autenticação, caso o usuário não tenha realizado o cadastro ele pode criar uma conta e após criar a conta poderá acessar a aplicação, caso tenha um cadastro ele pode acessar a aplicação. Após realizar o *login* o usuário é redirecionado para a tela inicial da aplicação, onde pode acessar a funcionalidade de cadastrar um veículo e cadastrar manutenções realizadas nos veículos cadastrados. Além de cadastrar as manutenções e carros é possível visualizar as manutenções realizadas, caso o usuário não deseje mais realizar nenhuma operação ele encerra a aplicação.

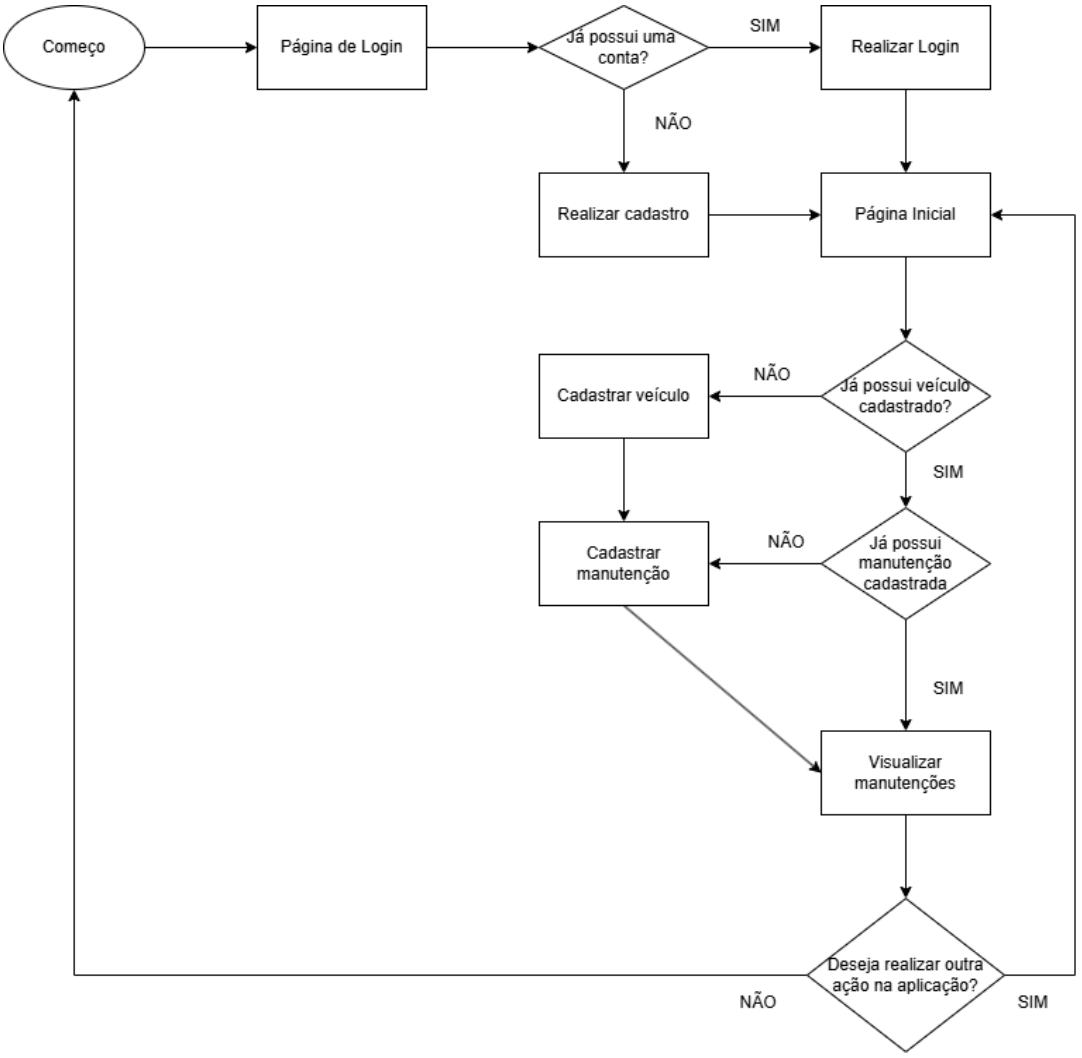



**Figura 8. Fluxo de navegação do usuário para a aplicação CarView.**

## 5.7. Construção do *back-end* para registro de usuários

Nesta fase, foi desenvolvido a lógica para cadastrar um usuário e autenticá-lo para acessar o sistema. Neste trabalho foi utilizado o serviço de autenticação nativo do *Spring Security*, que disponibiliza uma tela de *login* padrão, métodos para garantir que o usuário esteja autenticado e tenha a autorização necessária para acessar o sistema. Os usuários são armazenados no banco de dados e recuperados durante o processo de autenticação. Cada usuário possui um *ID* interno e um *ID* externo (*UUID*), utilizado para operações no sistema. Foi criado um *endpoint* que proporciona acesso às operações disponíveis na funcionalidade de

persistência/recuperação do usuário, seja para fazer *login* ou cadastrar um novo usuário. Nesse incremento foram criados os *endpoints* descritos na Tabela 1.

**Tabela 1. *endpoints* criados para registro de usuários**

| Funcionalidade | *endpoint* | Método HTTP |
|---|---|---|
| Listar todos os usuários | /carview/v1/users/ | *GET* |
| Obter os dados de um usuário com o *ID* externo informado | /carview/v1/users/{id} | *GET* |
| Listar preferências do usuário | /carview/v1/users/{id}/preferences | *GET* |
| Listar carros do usuário | /carview/v1/users/{id}/cars | *GET* |
| Inserir um novo usuário | /carview/v1/users/ | *POST* |
| Atualizar usuário | /carview/v1/users/{id} | *PUT* |
| Atualizar carro por *ID* do usuario | /carview/v1/users/{id}/car | *PUT* |
| Deletar Usuário | /carview/v1/users/{id} | *DELETE* |

## 5.8. Construção do *back-end* para registro do veículo

Nesta etapa, foi desenvolvido a lógica para persistir e recuperar um veículo, associando-o ao usuário que o criou (Figura 9). Isso possibilita a exibição de todos os veículos cadastrados pelo usuário, assim como as manutenções realizadas em cada veículo. Os veículos são armazenados durante o processo de cadastro como apresentado na linha 69 da Figura 9. Cada veículo possui um *ID* interno e um *ID* externo (*UUID*), utilizados para operações no sistema. Criamos alguns *endpoints* que permitem acesso às operações disponíveis na funcionalidade de persistência/recuperação de veículos, seja para criar, atualizar, listar ou excluir um veículo. Esses *endpoints* estão listados na Tabela 2.

```
59      @Transactional
60      public CarEntity insert(CarEntity carEntity, String user) {
61          try {
62              var userDb = userRepositoryFront.findByEmail(user);
63              carEntity.setUsuario(userDb.getId());
64              CarEntity savedCar = carRepository.save(carEntity);
65
66              List<Integer> categoryIds = categoryRepository.findAllIds();
67
68              for (Integer categoryId : categoryIds) {
69                  String sql = "INSERT INTO categories_cars (car_id, category_id) VALUES (?, ?)";
70                  jdbcTemplate.update(sql, savedCar.getId(), categoryId);
71              }
72
73              return savedCar;
74          } catch (DataIntegrityViolationException ex) {
75              throw ExceptionUtils.buildSameIdentifierException(Constants.CAR_DUPLICATED);
76          }
77      }
78
```

**Figura 9. Bloco de código responsável por inserir um novo veículo no banco de dados.**

**Tabela 2. *endpoints* criados para registro de carros**

| Funcionalidade | *endpoint* | Método HTTP |
|---|---|---|
| Listar carro por *ID* | /carview/v1/car{id} | *GET* |
| Listar categorias de carros por *ID* | /carview/v1/car/{id}/categories | *GET* |
| Criar um carro | /carview/v1/car | *POST* |
| Atualizar informações do carro por *ID* | /carview/v1/car{id} | *PUT* |
| Deletar carro por *ID* | /carview/v1/car{id} | *DELETE* |

## 5.9. Construção do *back-end* para registro de categoria e tipo

Nesta etapa, foram criadas duas entidades: Categoria e Tipo, que serão disponibilizadas para o usuário no momento de registrar uma manutenção (Figura 10). Ambas as entidades possuem *Enums* (Tipo de dado que consistem em um conjunto de valores fixos nomeados, geralmente usados para representar categorias, opções ou estados distintos) que apresentam os tipos de manutenção possíveis de serem realizados e suas respectivas categorias (Figura 10). Dessa forma, o usuário não precisa se preocupar em lembrar como a manutenção é categorizada e a qual tipo ela pertence. Os usuários não têm permissão para excluir ou criar novos tipos e categorias de manutenção, sendo possível utilizar apenas os já existentes. Os *endpoints* criados nesse incremento estão listados nas Tabelas 3 e 4.

```
@Getter
public enum CategoryEnum {

    PREVENTIVE((short) 0),
    CORRECTIVE((short) 1),
    PREDICTIVE((short) 2),
    SCHEDULED((short) 3),
    EMERGENCY((short) 4);

    private final short value;

    public static CategoryEnum fromValue(Short categoryCode) {
        var message = MessageResource.getInstance().getMessage( key: "category.invalid");
        return Arrays.stream(CategoryEnum.values()) Stream<CategoryEnum>
                .filter(it -> Short.valueOf(it.getValue()).equals(categoryCode))
                .findFirst() Optional<CategoryEnum>
                .orElseThrow(() -> ExceptionUtils.buildBadRequestException(message));
    }
    CategoryEnum(short value) { this.value = value; }
```

**Figura 10. Bloco de código de uma classe Enum responsável por representar os tipos de categoria.**

**Tabela 3. *endpoints* criados para registro de tipos de manutenção**

| Funcionalidade | *endpoint* | Método HTTP |
|---|---|---|
| Listar todos os tipos de manutenções | /carview/v1/type | *GET* |
| Listar tipo de manutenção por *ID* | /carview/v1/type/{id} | *GET* |

**Tabela 4. *endpoints* criados para registro de categorias de manutenção**

| Funcionalidade | *endpoint* | Método HTTP |
|---|---|---|
| Listar todas as categorias de manutenções | /carview/v1/category | *GET* |
| Listar categorias de manutenção por *ID* | /carview/v1/category/{id} | *GET* |
| Listar tipos de manutenção por *ID* da categoria de manutenção | /carview/v1/category/{id}/types | *GET* |
| Criar uma categoria | /carview/v1/category | *POST* |
| Atualizar informações da categoria por *ID* | /carview/v1/category{id} | *PUT* |
| Deletar categoria por *ID* | /carview/v1/category{id} | *DELETE* |

## 5.10. Construção do *back-end* para registro de manutenção

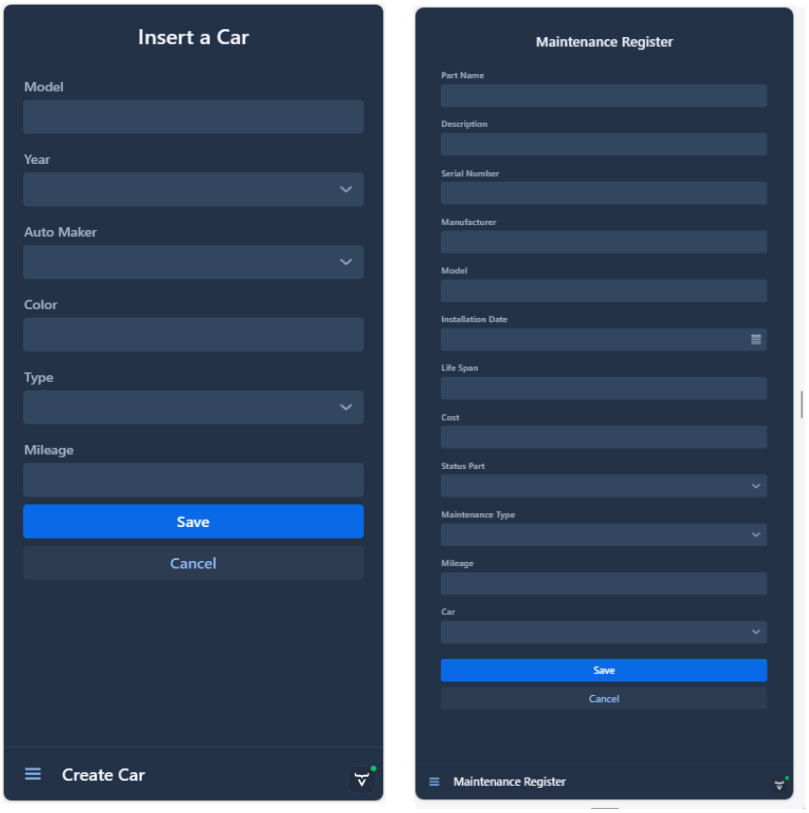
Nesta etapa, foi desenvolvida a lógica para persistir e recuperar as manutenções, associando-as a um veículo. Isso permite a exibição de todas as manutenções realizadas nos veículos. As manutenções são armazenadas e recuperadas durante o processo de cadastro/listagem. Cada manutenção possui um *ID* interno e um *ID* externo (*UUID*), utilizados para operações no sistema. O sistema solicita ao usuário atualizar a quilometragem do veículo ao menos uma vez por semana quando o usuário realiza o login, com o objetivo de manter o registro do carro atualizado e proporcionar uma experiência melhor. Criamos alguns *endpoints* (Tabela 5) que permitem acesso às operações disponíveis na funcionalidade de persistência/recuperação de manutenções, seja para criar, atualizar, listar ou excluir uma manutenção.

**Tabela 5. *endpoints* criados para registro de peças**

| Funcionalidade | *endpoint* | Método HTTP |
|---|---|---|
| Listar todas as peças | /carview/v1/part | *GET* |
| Listar peça por *ID* | /carview/v1/part/{id} | *GET* |
| Criar uma peça | /carview/v1/part | *POST* |
| Atualizar informações da peça por *ID* | /carview/v1/part{id} | *PUT* |
| Deletar peça por *ID* | /carview/v1/part{id} | *DELETE* |

### 5.11. Construção do *front-end* para integrar com *back-end* responsável por manter cadastro de veículo

Nesta etapa, foi construída a página web responsável pelo cadastro de veículos (Figura 11a). A interface foi desenvolvida para ser de fácil utilização através de um formulário simples e objetivo. Na tela de listagem de veículos, o gerenciamento dos veículos é feito de forma centralizada em uma interface web funcional, que permite ao usuário cadastrar, visualizar, editar e excluir veículos de maneira prática e eficiente.Com isso, é fornecida uma maneira funcional para realizar o cadastro e gerenciamento de veículos no sistema.




(a) (b)

**Figura 11. Telas do aplicativo CarView para (a) cadastro de veículo e (b) cadastro de manutenção.**

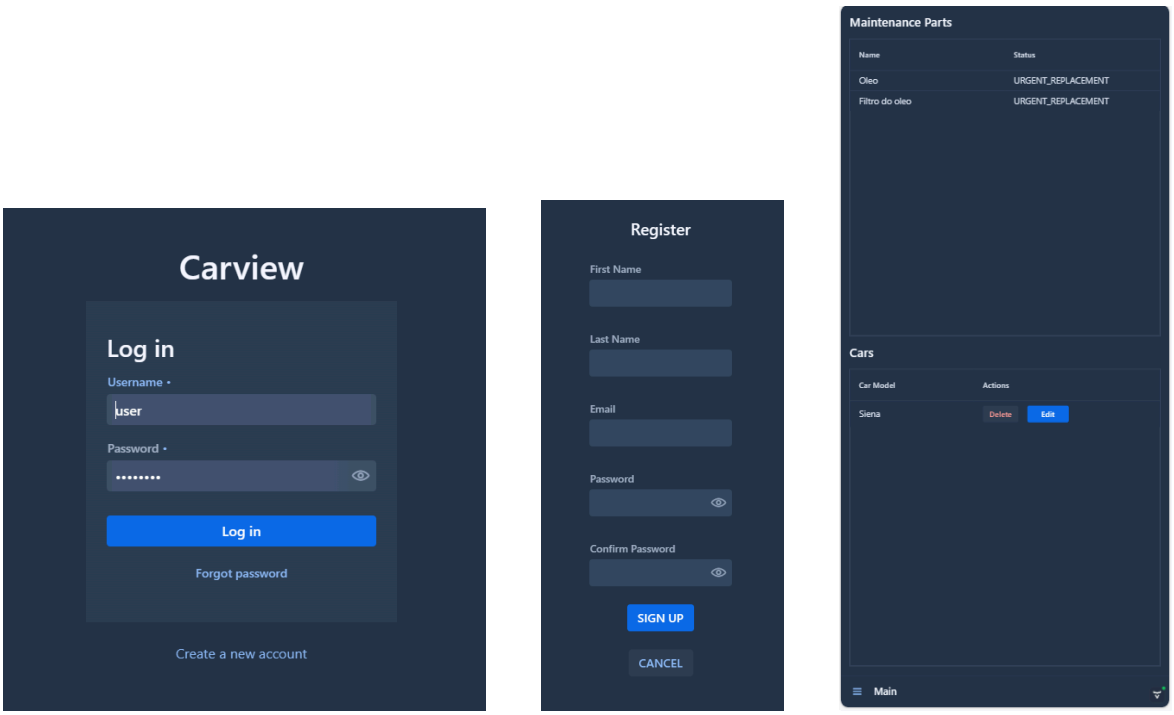
### 5.12. Construção do *front-end* para integrar com *back-end* responsável por manter cadastro de manutenção

Nesta etapa é seguido a mesma lógica da interface anterior, mantendo a fluidez e garantindo a interação eficaz através de formulários simples e objetivos, para o cadastro das manutenções dos veículos (Figura 11b). Essa página foi construída para ser interativa, ou seja, a medida que o usuário vai preenchendo os campos disponíveis, novos campos são adicionados à tela para camuflar a grande quantidade de campos que devem ser preenchidos. Além de cadastrar a manutenção, é possível editá-la e excluí-la. Com isso, é fornecida uma maneira funcional para realizar o cadastro e gerenciamento das manutenções no sistema.

## 5.13. Construção do *front-end* para integrar com *back-end* responsável pelo cadastro de usuários e Login

Nesta etapa, foram construídas duas telas: uma destinada a realizar o *login* do usuário (Figura 12a) e outra para o cadastro do usuário no sistema (Figura 12b). Quando o usuário acessa a aplicação, ele é redirecionado para a tela de *login*. Caso o usuário não possua um cadastro, é fornecida uma opção para realizar o cadastro através de um botão que redireciona para a tela de cadastro.

Os usuários acessam o sistema através do *e-mail* e da senha cadastrados. Dessa forma, é aceito apenas um e-mail único, não sendo possível cadastrar dois usuários com o mesmo endereço de *e-mail*, pois a coluna de *e-mail* no banco de dados é uma chave primária, o que não permite a inserção de *e-mails* duplicados. Além disso, as telas da aplicação foram projetadas para se conectar eficientemente ao *back-end*. Cada tela utiliza métodos encapsulados em classes devidamente injetadas, promovendo uma arquitetura modular e coesa.




(a) (b) (c)

**Figura 12. Telas do aplicativo CarView para (a) login, (b) cadastro de novo usuário e (c) inicial do aplicativo CarView.**

## 5.14. Construção do *front-end* para integrar com *back-end* responsável pela tela principal

Nesta etapa, foi construída a tela principal, à qual o usuário tem acesso após fazer *login* (Figura 12c). Nesta tela, o usuário pode visualizar todas as manutenções necessárias e o preço médio gasto por outros usuários. Por meio do menu lateral, é possível acessar a página de cadastro de veículos, o registro de manutenções, fazer *logout* e retornar ao menu principal caso esteja em outra tela. Semanalmente, quando o usuário acessar a aplicação, será solicitado ao usuário atualizar a quilometragem dos veículos cadastrados e vinculados a ele.

## 6. Conclusões

Este trabalho teve como foco o desenvolvimento de uma aplicação voltada para auxiliar proprietários de veículos no gerenciamento da manutenção de seus automóveis. Através da criação de um aplicativo PWA, buscamos oferecer uma solução abrangente e acessível em múltiplos dispositivos, permitindo o registro detalhado de manutenção, o acompanhamento da quilometragem e o acesso a informações sobre custos médios de peças e serviços.

Acreditamos que o objetivo principal do trabalho foi alcançado com sucesso. A aplicação desenvolvida oferece funcionalidades essenciais para o gerenciamento da manutenção veicular, como o cadastro de veículos, o registro detalhado de manutenção, incluindo data, tipo de serviço, peças utilizadas e custos, além da visualização do histórico de manutenções realizadas. O código da aplicação está disponível no *GitHub* no repositório: https://github.com/fgost/carview .

Apesar dos resultados positivos, identificamos pontos que podem ser aprimorados em trabalhos futuros. A adição de um sistema de notificações *push* para lembrar os usuários sobre manutenções seria uma funcionalidade útil. Além disso, integrar o aplicativo com oficinas mecânicas e lojas de autopeças permitiria agendar serviços e adquirir peças diretamente. Outras melhorias incluem o registro de abastecimentos para calcular o consumo médio de combustível e a inclusão do custo médio de cada tipo de manutenção nas peças.

Este trabalho integrou diversos conhecimentos adquiridos ao longo do curso de Análise e Desenvolvimento de Sistemas. Foram aplicados conceitos de modelagem e implementação de banco de dados, adquiridos nas disciplinas Banco de Dados I e II, além de tecnologias *web*, como *HTML, CSS* e *JavaScript*, abordadas em Desenvolvimento *Web* e Desenvolvimento de Sistemas *Web*. No *back-end*, utilizamos *Java* e *Spring Boot*, conhecimentos explorados em Programação Orientada a Objetos e Linguagem de Programação I, II e III. Também foram essenciais os fundamentos de Algoritmos e Programação, Estruturas de Dados I e II, Análise Orientada a Objetos e Arquitetura de *Software*. As disciplinas de Qualidade de *Software* e Gestão de Projetos contribuíram para garantir a eficiência e organização do desenvolvimento, enquanto o uso de metodologias ágeis, estudadas em Metodologias Ágeis, foi fundamental para a estruturação e entrega do projeto.

Em suma, este trabalho de conclusão de curso representa uma contribuição relevante para a área de desenvolvimento de aplicativos, oferecendo uma solução prática e eficiente para o gerenciamento da manutenção veicular. Acreditamos que a aplicação desenvolvida tem potencial para auxiliar proprietários de veículos a manterem seus automóveis em bom estado, garantindo maior segurança e economia.

Para aprimorar e expandir o escopo do projeto abordado neste trabalho, são sugeridas as seguintes recomendações para trabalhos futuros:

1. **Criação de Testes Unitários**: A implementação de testes unitários é fundamental para garantir a confiabilidade do sistema. Testes unitários permitem identificar e corrigir erros, além de assegurar que novas funcionalidades não irão comprometer o funcionamento das já existentes.
2. **Integração com Oficinas Mecânicas**: É interessante a integração com oficinas mecânicas para proporcionar um valor significativo ao sistema, oferecendo aos usuários uma maneira prática de agendar serviço e acompanhar o serviço realizado.

Esta funcionalidade permitiria a comunicação direta com oficinas parceiras, possibilitando a troca de informações sobre a condição dos veículos e o agendamento de serviços.

3. **Funcionalidade para Definir Lembretes de Manutenção Próximas**: A implementação de uma funcionalidade para definir lembretes de manutenções próximas ajudaria os usuários a manterem seus veículos com a manutenção em dia, prevenindo problemas inesperados. Essa funcionalidade poderia enviar notificações antecipadas sobre a necessidade de manutenção, com base em parâmetros como quilometragem ou tempo de uso.
4. **Funcionalidade para Sugerir Manutenções de Acordo com o Padrão do Usuário**: Desenvolver uma funcionalidade que sugira manutenções personalizadas com base no histórico e nas preferências do usuário seria um grande avanço. Essa funcionalidade poderia analisar os padrões de uso e as características individuais do veículo para oferecer recomendações mais precisas e úteis, aumentando a eficiência e a satisfação do usuário com o sistema.
5. **Recriação do *Front-End* para Ser Desacoplado do *Back-End***: Para melhorar a escalabilidade e a manutenção do sistema, é recomendada a recriação do *front-end* de forma desacoplada do *back-end,* ou seja como um micro serviço separado do *back-end*. Utilizar uma arquitetura baseada em *APIs* permitiria uma separação mais clara entre as camadas de apresentação e lógica de negócio. Isso não só facilita a atualização e a manutenção de cada componente de forma independente, mas também oferece maior flexibilidade para futuras melhorias e integrações.

Estas recomendações visam não apenas resolver limitações identificadas no sistema, mas também torná-lo mais robusto, eficiente e adaptável às necessidades dos usuários. Implementar essas melhorias contribuirá significativamente para a evolução contínua do sistema e a satisfação dos usuários.

## Referências


AWS. O que é o armazenamento em nuvem?, 2023 Disponível em: https://aws.amazon.com/pt/what-is/cloud-storage/#:~:text=armazenamento%20em%20nuvem%3F-,O%20que%20%C3%A9%20o%20armazenamento%20em%20nuvem%3F,conex%C3%A3o%20de%20rede%20privada%20dedicada. Acesso em: 19 dez. 2023.

ANDREI L. O Que é Hospedagem de Site? Guia Prático com Tipos de Hospedagem e Mais. 10 abr. 2024. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-hospedagem-de-site . Acesso em: 11 dez. 2024.

CARANGO App. Disponível em: https://play.google.com/store/apps/details?id=br.com.carango. Acesso em: 24 mai. 2023.

DRIVVO App. Como você gerencia o seu veículo ou a sua frota?, 2023. Disponível em: https://www.drivvo.com/pt-BR. Acesso em: 29 mar. 2024.

DZHANGAROV, A.; PAKHAEV, K.; POTAPOVA, N. *Modern web application development technologies*. **IOP Conference Series**: Materials Science and Engineering, v. 1155, 2021. Disponível em: https://www.researchgate.net/publication/352276087_Modern_web_application_development_technologies. Acesso em: 10 Set. 2023.

GEEWAX, J. J. API Design Patterns. New York: Manning Publications Co., 2021.

MARQUES, S. O que é escalabilidade de software e por que ela é tão importante?. 2022. Disponível em: https://uds.com.br/blog/o-que-e-escalabilidade-de-software/. Acesso em: 19 dez. 2023.

MAZZAROLO, V.; SILVA, R. S. da. Progressive Web Apps: Uma nova abordagem no desenvolvimento de aplicações Web. 2021. Trabalho de conclusão de curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Veranópolis, 2021. Disponível em: https://ifrs.edu.br/veranopolis/wp-content/uploads/sites/10/2022/04/Artigo-TCC-Vinicius-Mazzarolo-2021-Pub.pdf. Acesso em: 10 set. 2023.

OTANI, M.; MACHADO, W. V.. A proposta de desenvolvimento de gestão da manutenção industrial na busca da excelência ou classe mundial, **Revista Gestão Industrial**, v. 4, ed. 2, 2008. Disponível em: https://periodicos.utfpr.edu.br/revistagi/article/view/16 .

PORTAL TERRA. Junho é o mês da conscientização pela manutenção preventiva de automóveis, 10 Jun 2017. Disponível em: https://www.terra.com.br/noticias/junho-e-o-mes-da-conscientizacao-pela-manutencao-preventiva-de-automoveis,c3ff43d6a00b9b65789585f6d5e9db19lfs7qqk6.html. Acesso em: 19 dez. 2023.

POSTGRESQL. The World's Most Advanced Open Source Relational Database, 2023. Disponível em: https://postgresql.org. Acesso em: 29 mar. 2023.

SILVA, F. A. DE S.; PRADO, E. F. DO. Análise teórica sobre o desenvolvimento de aplicativos nativos, híbridos e webapps, **Revista EduFatec: educação, tecnologia e gestão**, v. 2, n. 1, p. 1-18, 1 jun. 2019. Disponível em: https://revistaedufatec.fatecfranca.edu.br/wp-content/uploads/2019/09/AN%C3%81LISE-TE%C3%93RICA-SOBRE-O-DESENVOLVIMENTO-DE-APLICATIVOS-NATIVOS-H%C3%8DBRIDOS-E-WEBAPPS.pdf . Acesso em: 24 maio 2023.

VAADIN FRAMEWORK. Build Java web applications faster with Vaadin, 2023. Disponível em: https://vaadin.com/docs/latest/overview. Acesso em: 14 jun. 2023.

VASCONCELOS, L. C.; BRILHANTE, I.; LEMOS, S. Entenda como funciona streaming de dados em tempo real, 2020. Disponível em: https://www.insightlab.ufc.br/entenda-como-funciona-streaming-de-dados-em-tempo-real-2/. Acesso em: 19 dez. 2023.

# Documento Digitalizado Público

## Anexo I - artigo - TCC

**Assunto:** Anexo I - artigo - TCC
**Assinado por:** Andre Constantino
**Tipo do Documento:** Relatório Externo
**Situação:** Finalizado
**Nível de Acesso:** Público
**Tipo do Conferência:** Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 16/02/2025 19:55:40.

Este documento foi armazenado no SUAP em 16/02/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

**Código Verificador:** 1940693
**Código de Autenticação:** b0e8c20887