PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES

TCC II

Sistema Embarcado de Controle de Navegação Terrestre Aplicativo ‘FuelLog’

ALUNO

Gabriel Vaz de Paula

ORIENTADOR

Prof. Dr. Leonardo Guerra de Rezende Guedes

Goiânia
2023

INTRODUÇÃO

O controle do consumo de combustível é uma preocupação constante para motoristas e gestores de frotas, já que representa um grande custo operacional para empresas e indivíduos. Além disso, a eficiência energética e a redução da emissão de gases poluentes também são preocupações ambientais cada vez mais presentes no setor automotivo. Portanto, a otimização da eficiência do consumo de veículos automotivos assume uma considerável relevância em um contexto global caracterizado pela escassez de recursos naturais e pela crescente demanda dos indivíduos em reduzir seu impacto ambiental, ao mesmo tempo em que buscam economizar diariamente.

Diante desse cenário, a busca por soluções que visem a redução do consumo de combustível e a emissão de gases poluentes se torna cada vez mais relevante. Nesse sentido, o desenvolvimento de um aplicativo Android de controle de consumo de combustível embarcado em um veículo se apresenta como uma solução viável e eficiente para a gestão dos gastos com combustível e para a promoção da eficiência energética e redução da emissão de gases poluentes.

A análise de requisitos baseada na ISO 25010 e a modelagem em casos de uso por UML são técnicas importantes para a elaboração de um aplicativo de qualidade, pois permitem identificar os requisitos essenciais do usuário e modelar as funcionalidades do software de forma clara e objetiva. Dessa forma, o presente plano de trabalho tem como objetivo desenvolver um aplicativo Android de controle de consumo de combustível embarcado em um veículo utilizando estas técnicas, assim como técnicas próprias inspiradas nestas, a fim de fornecer aos usuários uma ferramenta eficiente e fácil de usar para o controle do consumo de combustível, bem como contribuir para a melhoria da eficiência energética e redução da emissão de gases poluentes.

1 JUSTIFICATIVA

De acordo com o estudo "Cenário da Frota Automotiva Brasileira 2021", realizado pela Associação Nacional dos Fabricantes de Veículos Automotores (ANFAVEA), a frota brasileira de veículos atingiu a marca de 108,2 milhões de unidades em 2020, sendo que mais de 90% desses veículos são movidos a combustíveis fósseis (ANFAVEA, 2021). Além disso, segundo a Agência Nacional de Transportes Terrestres (ANTT, 2020), o setor de transporte rodoviário é responsável por cerca de 95% das emissões de CO2 no Brasil.

Diante desse cenário, a busca por soluções que visem a redução do consumo de combustível e a emissão de gases poluentes se torna cada vez mais relevante. Nesse sentido, o desenvolvimento de um aplicativo Android de controle de consumo de combustível embarcado em um veículo se apresenta como uma solução viável e eficiente para a gestão dos gastos com combustível e para a promoção da eficiência energética e redução da emissão de gases poluentes.

Além disso, a utilização de técnicas como a análise de requisitos baseada na ISO 25010 e a modelagem em casos de uso por UML permite a elaboração de um aplicativo de qualidade, que atenda às necessidades e expectativas dos usuários. Dessa forma, o presente plano de trabalho é importante e relevante para o desenvolvimento de soluções tecnológicas que visem a sustentabilidade ambiental e a eficiência energética, além de contribuir para a formação de profissionais capacitados em Engenharia de Software.

1.1 Objetivo geral

Desenvolver um aplicativo Android de controle de consumo de combustível embarcado em um veículo utilizando técnicas de análise de requisitos baseada na ISO 25010 e a modelagem em casos de uso por UML.

1.2 Objetivos específicos

a) Aplicar conceitos de análise de requisitos para propor o aplicativo de controle de consumo de combustível baseado na ISO 25010.

b) Identificar as funcionalidades principais do aplicativo por meio da modelagem em casos de uso por UML.

c) Desenvolver o aplicativo Android de controle de consumo de combustível embarcado em um veículo.

d) Realizar testes e validação do aplicativo para garantir a sua eficiência e confiabilidade.

e) Comparar os resultados obtidos com os requisitos identificados na análise de requisitos baseada na ISO 25010.

f) Elaborar uma monografia de Trabalho de Conclusão de Curso (TCC II) contendo as principais conclusões e contribuições do trabalho.

2 REFERENCIAL TEÓRICO

2.1 Engenharia de Software

Engenharia de Software é a área da computação que trata do desenvolvimento de sistemas de software com o objetivo de produzir software de qualidade de forma eficiente e eficaz. De acordo com Pressman (2022), a Engenharia de Software pode ser definida como "a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de software, incluindo a aplicação de princípios de engenharia para o processo de software".

Segundo Sommerville (2022), a Engenharia de Software pode ser dividida em três áreas principais: desenvolvimento de software, gerência de projetos de software e qualidade de software. O desenvolvimento de software engloba atividades como a análise de requisitos, a modelagem, a codificação e os testes do software. Já a gerência de projetos de software envolve o planejamento, controle e coordenação das atividades do projeto, enquanto a qualidade de software se refere à garantia de que o software atenda aos requisitos definidos e seja confiável, eficiente e seguro.

Para Pressman (2022), a Engenharia de Software tem como objetivo principal a produção de software de qualidade, que atenda aos requisitos do usuário e seja entregue dentro do prazo e do orçamento estipulados. Para isso, a Engenharia de Software utiliza uma série de técnicas e ferramentas, como a análise de requisitos, a modelagem, os testes e a documentação, que permitem o desenvolvimento de software de forma sistemática e disciplinada.

O uso das técnicas e práticas da Engenharia de Software é fundamental para garantir a qualidade, eficiência e segurança do software desenvolvido. Segundo o Relatório de Qualidade de Software de 2021 da empresa americana CAST, o custo anual dos defeitos de software nos Estados Unidos ultrapassou a marca de US$ 2,08 trilhões. Isso representa cerca de 10% do Produto Interno Bruto (PIB) do país (CAST, 2021).

Esse dado reforça a importância da utilização de técnicas e práticas de Engenharia de Software, como a análise de requisitos, a modelagem, os testes e a documentação, que permitem a detecção precoce de defeitos e a garantia da qualidade do software produzido. Os princípios da Engenharia de Software, permitem a entrega de software de qualidade de forma mais rápida e eficiente.

Portanto, a adoção de técnicas e práticas de Engenharia de Software é fundamental para a produção de software de qualidade e para a redução dos custos e riscos associados aos defeitos de software.

2.2 Software product Quality Requirements and Evaluation (ISO 25010:2011)

A norma ISO 25010 (2011) é uma norma que define um modelo de qualidade de software, também conhecido como SQuaRE (Software product Quality Requirements and Evaluation). Essa norma estabelece uma estrutura para avaliar a qualidade de software em termos de suas características e subcaracterísticas.

A ISO 25010 é composta por oito características principais, que são: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade, portabilidade, segurança e compatibilidade. Cada uma dessas características é dividida em subcaracterísticas mais específicas.

A característica de funcionalidade, por exemplo, é dividida em subcaracterísticas como adequação funcional, precisão, interoperabilidade e segurança funcional. Já a característica de confiabilidade é dividida em subcaracterísticas como maturidade, tolerância a falhas, capacidade de recuperação e conformidade.

Essas características e subcaracterísticas podem ser usadas para avaliar a qualidade de um software em diferentes etapas de seu ciclo de vida, desde a definição dos requisitos até a manutenção do sistema. A ISO 25010 é amplamente reconhecida como uma referência para avaliação de qualidade de software e é usada por muitas empresas e organizações em todo o mundo.

2.3 Características da ISO 25010

A seguir veremos cada umas dessas características da ISO 25010, juntamente com exemplos relacionados a um aplicativo Android de controle de consumo de combustível embarcado em um veículo. Deve-se enfatizar que para um aplicativo Android de controle de consumo de combustível, é importante garantir que essas características sejam levadas em consideração para fornecer uma experiência de usuário confiável, fácil de usar e segura.

2.3.1 Funcionalidade

Refere-se às funcionalidades do software e sua capacidade de atender aos requisitos do usuário. No caso do aplicativo Android de controle de consumo de combustível, as funcionalidades incluem

registrar informações de abastecimento, cálculo de consumo de combustível e emissão de relatórios. Um exemplo de requisito funcional seria a capacidade de armazenar e exibir as informações de abastecimento de maneira clara e organizada.

2.3.2 Confiabilidade

Refere-se à capacidade do software de manter o desempenho desejado sob condições específicas por um determinado período. No caso do aplicativo Android de controle de consumo de combustível, a confiabilidade é importante para garantir que os dados registrados de abastecimento sejam precisos e confiáveis. Por exemplo, o aplicativo deve ser capaz de lidar com interrupções de conexão com a internet e garantir que os dados registrados sejam armazenados corretamente.

2.3.3 Usabilidade

Refere-se à facilidade de uso e eficiência do software para alcançar metas específicas com eficácia, eficiência e satisfação do usuário. No caso do aplicativo Android de controle de consumo de combustível, a usabilidade é importante para garantir que os usuários possam facilmente acessar e entender as informações. Por exemplo, a interface do usuário deve ser clara e intuitiva, com ícones e botões facilmente identificáveis.

2.3.4 Eficiência

Refere-se ao desempenho em relação aos recursos utilizados para atender aos requisitos do software. No caso do aplicativo Android de controle de consumo de combustível, a eficiência é importante para garantir que o aplicativo funcione sem problemas e sem causar lentidão ao sistema do dispositivo móvel. Por exemplo, o aplicativo deve ser projetado para consumir recursos mínimos de processamento e memória.

2.3.5 Manutenibilidade

Refere-se à facilidade de manter e modificar o software para atender a requisitos em mudança. No caso do aplicativo Android de controle de consumo de combustível, a manutenibilidade é importante para garantir que o aplicativo possa ser atualizado com facilidade e rapidez, em caso de correções de bugs ou atualizações de recursos. Por exemplo, o código-fonte do aplicativo deve ser organizado e documentado de maneira clara para facilitar a manutenção.

2.3.6 Portabilidade

Refere-se à capacidade do software de ser executado em diferentes plataformas e ambientes de hardware. No caso do aplicativo Android de controle de consumo de combustível, a portabilidade é importante para garantir que o aplicativo possa ser usado em diferentes dispositivos Android com diferentes configurações de hardware. Por exemplo, o aplicativo deve ser projetado para funcionar em diferentes tamanhos de tela e resoluções, bem como diferentes versões do sistema operacional Android.

2.3.7 Segurança

Refere-se à capacidade do software de proteger as informações e processos contra acesso não autorizado e ataques maliciosos. No caso do aplicativo Android de controle de consumo de combustível, a segurança é importante para garantir que as informações do usuário sejam protegidas contra acessos não autorizados. Por exemplo, o aplicativo deve exigir uma senha forte e criptografar as informações do usuário. Também é importante garantir que as comunicações com servidores externos sejam criptografadas para evitar interceptações mal-intencionadas.

2.3.8 Compatibilidade

Refere-se à capacidade do software de funcionar corretamente com outros sistemas ou componentes de hardware e software. No caso do aplicativo Android de controle de consumo de combustível, a compatibilidade é importante para garantir que o aplicativo possa se comunicar com o computador de bordo do veículo e ler as informações de consumo de combustível. Também é importante garantir que o aplicativo funcione corretamente com diferentes versões do sistema operacional Android e com diferentes modelos de smartphones.

2.4 MODELAGEM UML

A Modelagem em casos de uso por UML (Unified Modeling Language) é uma técnica utilizada na Engenharia de Software para a modelagem de sistemas de software. De acordo com Sommerville (2022), a modelagem em casos de uso é uma técnica de modelagem de requisitos que se concentra em descrever as funcionalidades do sistema sob a perspectiva do usuário.

Para Pressman (2021), a Modelagem em casos de uso é

> *"uma técnica de modelagem de requisitos usada para descrever como um usuário ou sistema interage com um sistema em particular".*

Através da Modelagem em casos de uso, é possível identificar as funcionalidades do sistema, as necessidades dos usuários e as restrições de uso do sistema.

Segundo Larman (2022), a Modelagem em casos de uso envolve a criação de diagramas de casos de uso, que descrevem os atores que interagem com o sistema e as funcionalidades que o sistema deve fornecer. Esses diagramas são uma ferramenta eficaz para comunicar os requisitos do sistema para os desenvolvedores e os stakeholders do projeto.

A modelagem em UML (Unified Modeling Language) é uma linguagem de modelagem gráfica utilizada para a visualização, especificação, construção e documentação de sistemas de software.

Com base no aplicativo Android de controle de consumo de combustível embarcado em um veículo, apresentamos exemplos de utilização dos principais elementos da modelagem UML:

2.4.1 Diagrama de Casos de Uso

O diagrama de casos de uso descreve as funcionalidades do sistema em termos das interações entre os atores e o sistema. No caso do aplicativo de controle de consumo de combustível, um exemplo de caso de uso seria "Registrar Abastecimento". Nesse caso, o usuário (ator) interage com o sistema para inserir as informações do abastecimento, como a quantidade de combustível, o valor pago e a quilometragem.

2.4.2 Diagrama de Classe

O diagrama de classe representa as classes do sistema e seus relacionamentos. No caso do aplicativo de controle de consumo de combustível, uma possível classe seria "Abastecimento", que teria atributos como a quantidade de combustível, o valor pago e a quilometragem, além de métodos como, por exemplo, "Registrar Abastecimento".

2.4.3 Diagrama de Sequência

O diagrama de sequência descreve a interação entre os objetos do sistema em uma determinada sequência de eventos. No caso do aplicativo de controle de consumo de combustível, um exemplo de diagrama de sequência seria a interação entre o usuário e o sistema no processo de registro de um abastecimento.

2.4.4 Diagrama de Atividade

O diagrama de atividade descreve as atividades e processos do sistema em uma determinada sequência. No caso do aplicativo de controle de consumo de combustível, um exemplo de diagrama de atividade seria o processo de cálculo da média de consumo, que envolveria atividades como a obtenção dos dados de abastecimento e o cálculo da média.

2.5 Diagrama de Componentes

O diagrama de componentes mostra a estrutura do sistema em termos de seus componentes e suas interações. No caso do aplicativo de controle de consumo de combustível, um exemplo de componente seria o módulo de cálculo da média de consumo, que receberia os dados de abastecimento e realizaria o cálculo da média.

2.5.1 Diagrama de Implantação

O diagrama de implantação descreve a arquitetura física do sistema em termos de seus componentes e sua distribuição em hardware e software. No caso do aplicativo de controle de consumo de combustível, um exemplo de diagrama de implantação seria a distribuição do sistema em um dispositivo móvel (hardware) e o aplicativo (software).

2.5.2 Detalhamento de Casos de Uso

O Detalhamento de Casos de Uso (ou UC) é uma técnica utilizada no desenvolvimento de software para descrever detalhadamente os fluxos de interação entre o sistema e seus usuários ou outros sistemas. O objetivo do detalhamento de casos de uso é definir com clareza as entradas, processamentos e saídas de cada interação do usuário com o sistema, assim como os resultados esperados e os possíveis erros ou exceções que podem ocorrer.

O detalhamento de casos de uso normalmente é feito em etapas, iniciando com uma descrição geral do caso de uso e em seguida detalhando cada etapa da interação do usuário com o sistema. É comum o uso de diagramas de fluxo de dados e diagramas de sequência para ajudar na descrição detalhada dos fluxos de interação.

Alguns elementos que podem ser incluídos no detalhamento de casos de uso são:

a) Nome do caso de uso;

b) Atores envolvidos;

c) Pré-condições e pós-condições;

d) Fluxo básico de interação;

e) Fluxos alternativos ou exceções;

f) Requisitos não-funcionais associados;

g) Diagramas de fluxo de dados, de sequência ou de classes.

Na modelagem em UML, um caso de uso pode ser representado de duas formas: caso de uso completo e caso de uso abstrato.

O caso de uso completo é uma representação detalhada do comportamento do sistema em relação a um determinado requisito ou funcionalidade, que inclui as etapas de fluxo principal, fluxos alternativos e exceções. É uma representação completa e detalhada do comportamento do sistema em relação a um requisito específico.

Por outro lado, o caso de uso abstrato é uma representação mais geral e resumida do comportamento do sistema em relação a um conjunto de requisitos ou funcionalidades. Ele é utilizado para abstrair as funcionalidades comuns a vários casos de uso, evitando a duplicação de esforços na modelagem.

Um exemplo de caso de uso completo para o aplicativo de controle de consumo de combustível embarcado em um veículo seria o caso de uso "Registro de Abastecimento". Este caso de uso descreveria detalhadamente o comportamento do sistema em relação ao registro de um novo abastecimento, incluindo as etapas de validação dos dados, cálculo de média de consumo, atualização do histórico de abastecimentos, entre outras.

Já um exemplo de caso de uso abstrato para o mesmo sistema poderia ser o caso de uso "Gerenciamento de Abastecimentos". Neste caso, o objetivo seria abstrair as funcionalidades comuns a vários casos de uso relacionados a abastecimentos, como a validação dos dados, o cálculo da média de consumo e a atualização do histórico. Isso permitiria evitar a duplicação de esforços na modelagem e simplificar a representação do comportamento do sistema em relação a essas funcionalidades.

Este é um exemplo de sua aplicação para o login em um aplicativo Android:

Tabela 1: Exemplo de Detalhamento de Caso de Uso (Fonte: o Autor)

Caso de uso: Realizar login no aplicativo

Atores: Usuário

Descrição: Este caso de uso descreve o processo de login do usuário no aplicativo.

Pré-condições:

1. O aplicativo está instalado e funcionando corretamente.
2. O usuário possui uma conta válida no aplicativo.
3. O usuário tem acesso à internet.

Fluxo principal:

1. O usuário inicia o aplicativo.
2. O aplicativo exibe a tela de login.
3. O usuário insere seu e-mail e senha.
4. O usuário clica no botão "Entrar".
5. O aplicativo verifica se o e-mail e a senha são válidos.
6. Se as credenciais forem válidas, o aplicativo realiza o login do usuário e exibe a tela principal do aplicativo.
7. Se as credenciais não forem válidas, o aplicativo exibe uma mensagem de erro e permite que o usuário tente novamente.

Fluxos alternativos:

1. No passo 5, se as credenciais não forem válidas, o aplicativo permite que o usuário redefina sua senha por meio de um link enviado por e-mail.
2. No passo 5, se o e-mail do usuário não estiver registrado, o aplicativo exibe uma mensagem de erro e permite que o usuário crie uma conta.

Pós-condições:

1. O usuário está logado no aplicativo.
2. O usuário tem acesso às funcionalidades do aplicativo.

Este é apenas um exemplo simples de como um caso de uso pode ser detalhado para um aplicativo Android. Dependendo da complexidade do aplicativo e do caso de uso em questão, é possível incluir mais detalhes, como diagramas de sequência ou fluxogramas de processos.

2.5.3 Tabela de Estados

Uma tabela de estados é uma representação estruturada de um algoritmo em que cada linha da tabela corresponde a um estado do algoritmo em um determinado momento da sua execução. Cada estado é caracterizado por um conjunto de variáveis e valores associados a elas.

Por exemplo, suponha que temos um algoritmo que realiza uma busca sequencial em uma lista de números inteiros para encontrar um determinado valor. Podemos criar uma tabela de estados para acompanhar a execução do algoritmo, como mostrado abaixo:

Tabela 2: Exemplo de Tabela de Estados de um Algoritmo (Fonte: o Autor)

Etapa	Estado	Valores
Início	lista, valorProcurado	[5, 7, 2, 9, 4, 1], 7
1	i	0
2	-	-
3	-	-
...	...	...
N	-	-
Fim	i, valorEncontrado	1, True

Na tabela acima, a etapa "Início" corresponde ao momento em que o algoritmo é iniciado, com a lista de números inteiros e o valor procurado já definidos. Na coluna "Valores", temos a lista e o valor procurado como variáveis relevantes para essa etapa, enquanto na coluna "Estado", temos o atual estado de execução do algoritmo.

Ao longo da execução do algoritmo, novas etapas são geradas, cada uma com seus próprios valores e estados. Na tabela acima, a etapa "1" corresponde ao momento em que o índice "i" é inicializado com o valor 0. Já a etapa "Fim" corresponde ao momento em que o valor procurado é

encontrado na lista, com o índice "i" correspondente e a variável "valorEncontrado" definida como True.

Uma tabela de estados é uma ferramenta útil para acompanhar a execução de um algoritmo e identificar possíveis erros ou otimizações.

Além desses elementos, a UML também inclui outros elementos como diagramas de estado, de pacotes, de comunicação, entre outros. Esses elementos são utilizados para modelar diferentes aspectos do sistema de software, permitindo uma representação mais completa e precisa do sistema em questão.

Neste plano de trabalho serão eleitos apenas alguns destes elementos que forem suficientes para a boa prática da modelagem do sistema pretendido, dentre eles a o Detalhamento de Casos de Uso e a Tabela de Estados.

3 METODOLOGIA

No plano de trabalho apresentado, o estudante pretende desenvolver um aplicativo Android de controle de consumo de combustível utilizando a análise de requisitos baseada na ISO 25010 e a modelagem em casos de uso por UML (SEIDL; ZANELLA, 2020). Nesse sentido, pode-se classificar os métodos e técnicas de pesquisa a serem utilizados como quantitativos e exploratórios.

O método quantitativo é adequado para o estudo de fenômenos que podem ser medidos objetivamente, o que é o caso do desenvolvimento de um aplicativo Android de controle de consumo de combustível. Através desse método, será possível coletar dados quantitativos sobre as características do software, tais como a eficiência, a confiabilidade e a usabilidade, através da aplicação de questionários ou testes de usabilidade (LAVILLE; DIONNE, 2020).

Por outro lado, o método exploratório é indicado para a investigação de um tema ainda pouco explorado e que requer um aprofundamento na análise de suas características. A utilização da análise de requisitos baseada na ISO 25010 e a modelagem em casos de uso por UML são técnicas exploratórias que permitem o entendimento aprofundado do problema e a identificação das soluções mais adequadas para o desenvolvimento do aplicativo de controle de consumo de combustível. Além disso, a modelagem em casos de uso por UML permitirá a representação gráfica das funcionalidades do aplicativo, auxiliando na identificação de eventuais problemas e na comunicação efetiva com os usuários finais (PRESSMAN, 2020).

3.1 Procedimentos Metodológicos

Serão executados em ordem cronológica os seguintes procedimentos:

a) Levantamento e análise de requisitos baseando-se na ISO 25010;

b) Modelagem em casos de uso por UML segundo elementos eleitos;

c) Implementação do aplicativo Android de controle de consumo de combustível;

d) Testes e validação do aplicativo.

3.2 Mapa Mental/Conceitual

Um Mapa Mental/Conceitual foi elaborado elencando todos os artefatos e elementos a serem abortados neste trabalho. A partir desta Mapa Mental/Conceitual tomaremos as técnicas listadas a seguir nesta Sessão de Metodologia para execução deste Plano de Trabalho.

Figura 1: Mapa Mental/Conceitual (fonte: o Autor)

4 RESULTADOS ALCANÇADOS

4.1 Tabelas de Elicitação de Requisitos

As tabelas de formulário são ferramentas úteis para a elicitação de requisitos em projetos de software. Elas permitem a organização sistemática das informações coletadas dos stakeholders, facilitando a análise e identificação dos requisitos do sistema.

Na apresentação das tabelas de formulário, é importante seguir algumas recomendações para garantir sua eficácia na elicitação de requisitos. Primeiramente, é necessário esclarecer o objetivo da coleta de informações e o contexto do projeto, para que os stakeholders compreendam a importância e relevância de suas respostas.

Além disso, é fundamental explicar a estrutura da tabela de formulário e como as informações serão utilizadas no processo de análise e definição dos requisitos. A apresentação deve ser clara e objetiva, com exemplos de respostas para orientar os stakeholders na elaboração de suas contribuições.

Durante a coleta de informações, é importante que os responsáveis pela elicitação dos requisitos estejam disponíveis para esclarecer dúvidas e garantir que as respostas estejam completas e coerentes. Ao final da coleta, é recomendável realizar uma revisão das informações coletadas para garantir a qualidade e consistência dos requisitos identificados.

A seguir são apresentadas, para cada critérios da ISO 25010 (2011), as tabelas de formulário de elicitação de requisitos que permitiram a coleta organizada e sistemática das informações necessárias para a definição dos requisitos do sistema. Percebe-se que esta técnica é importante para orientar os stakeholders, além de garantir a disponibilidade para esclarecimento de dúvidas e revisão das informações coletadas.

Além disso, há uma coluna de relacionamento com os quesitos elicitados e os casos de USO UML a serem apresentados após as tabelas.

4.1.1 Confiabilidade

Tabela 3: Requisitos não funcionais de Confiabilidade (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Disponibilidade / Recuperabilidade	1. Validação de tempo para retornar ao funcionamento após falha?		1.1. Quanto tempo, em segundos, o sistema deve levar para retornar ao funcionamento?	-
Disponibilidade / Recuperação de desastres	2. Recuperação de funcionamento em caso de falha na multimídia?		2.1. Quanto tempo, em segundos, o sistema deve levar para retornar ao funcionamento?	-
Eficácia	3. 100% de eficácia nos resultados?	3.1. Qual a medida deve ser considerada para o cálculo de eficácia? (Exemplo: Tempo em segundos para obter o resultado)	3.1.1. O cálculo de consumo deve ter 100% de eficácia? 3.1.2. O cálculo de previsão de consumo deve ter 100% de eficácia? 3.1.3. O cálculo de distância deve ter 100% de eficácia?	3, 5
Exatidão / Precisão	4. Casas decimais para cálculo de resultados? 5. Deve ser arredondado?	4.1. Quantas casas decimais devem ser usadas para apresentar dados ao usuário?	4.1.1 Quantas casas decimais devem ser usadas para cálculo de resultados? 5.1. Qual a regra para realizar o arredondamento?	3, 5
Número de defeitos	6. Número de defeitos por linha de código será contabilizado?		6.1. Qual deve ser o número máximo de defeitos para cada mil linhas de código?	-

Maturidade	7. Deve haver redução no número de defeitos estipulados (por mil linhas de código) com o passar do tempo?		7.1. Qual deve ser a taxa de redução?	-
Previsão de confiabilidade	8. Disponibilidade de resultados anteriores armazenados?	8.1. Quais resultados anteriores armazenados o usuário deve ter acesso?	8.1.1. Quantos % do tempo (em um dia) os resultados devem estar disponíveis?	3, 4, 5
Resiliência / robustez / tolerância a falhas	9. Os resultados devem ser armazenados localmente?	9.1. Quais resultados devem estar disponíveis localmente?	9.1.1. Por quanto tempo (em dias) os resultados devem ser armazenados localmente?	2, 3, 4
Tempo médio entre falhas	10. Deve ser analisado o tempo médio entre falhas?		10.1. Qual o tempo máximo aceitável?	-
Frequência e gravidade da falha	11. Deve ser analisada a frequência das falhas? 12. Deve ser analisada a gravidade das falhas?		11.1. Qual a frequência máxima para falhas? 12.1. Quais os níveis de gravidade?	-

4.1.2 Compatibilidade

Tabela 4: Requisitos não funcionais de Compatibilidade (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos De Negócio*	*Requisitos De Usuário*	*Requisitos De Sistema*	*Caso De Uso*
Coexistência	13. Acesso ao sistema fora do ambiente do veículo? 14. Comunicação com outros sistemas?	13.1. O sistema deve ser capaz de se comunicar com um app de smartphone? 13.2. O sistema deve ser capaz de se comunicar com um smartwatch? 14.1. O sistema deve ser capaz de se comunicar com aplicativos de mapas? 14.2. Deve interromper a execução de outros aplicativos?	13.1.1. Tem que se comunicar com Android? 13.1.2. Tem que se comunicar com iOS? 13.2.1. Tem que se comunicar com o Apple OS? 14.1.1. Tem que se comunicar com o Waze? 14.1.2. Tem que se comunicar com o Google Maps? 14.2.2. Tem que manter a execução de aplicativos de música?	1, 3, 5

Interoperabilidade / Interface E Interações Entre Sistemas	15. Limite de tempo de resposta? 16. Tamanho mínimo e/ou máximo de tela? 17. Será possível personalizar o layout principal?	15.1. Qual o limite de tempo de resposta para interações do usuário? 16.1. O layout será diferente para cada tamanho? 17.1. Quais personalizações estarão disponíveis para o usuário?	15.1.1. Quantas interações simultâneas devem ser permitidas? 15.1.2. Qual o tempo máximo aceitável para a inicialização do sistema? 16.1.1. Deve ser mantido um tamanho mínimo para os componentes do layout? 17.1.1. A personalização por parte do usuário deve ser mantida? 17.1.2. Como será feita a restauração do padrão original?	1, 2, 3, 4, 5

Interoperabilidade / Restrições De Formatos	18. Tipo de armazenamento de dados? 19. Formato fixo para inserção de dados pelo usuário?	19.1. Será informado ao usuário o formato de dados ele deve inserir? 19.2. O usuário irá inserir dados que demandam validação lógica? Exemplo: CPF, CNPJ, CEP.	18.1. Qual o tipo de BD será utilizado para armazenar os dados? 19.1.1. O sistema deve ser construído de forma robusta pra evitar inserção de dados em formatos incorretos? 19.1.2. Caso sejam inseridos dados no formato incorreto, qual deve ser a abordagem do sistema? 19.2.1. Dados que demandam validação lógica serão armazenados de forma convencional?	2, 4, 5

4.1.3 Performance / Eficiência

Tabela 5: Requisitos não funcionais de Performance/Eficiência (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Capacidade atual e futura (escalabilidade) / Capacidade dinâmica	20. Capacidade mínima de processamento simultâneo?		20.1. Quantos cálculos simultâneos o sistema deve ser capaz de fazer? 20.2. O sistema deve ser capaz de realizar tipos diferentes de cálculo de forma simultânea?	3, 5
Capacidade atual e futura (escalabilidade) / Capacidade estática	21. Capacidade mínima de armazenamento do sistema (BD)?	21.1. Quanto de espaço estará disponível aos dados do usuário? 21.2. Quanto de espaço será reservado ao sistema?	21.1.1. O usuário deve ser capaz manipular os dados oriundos de seu uso? (Exemplo: Excluir, Alterar, Inserir manualmente) 21.2.1. Este espaço é estático ou há uma previsão de expansão conforme o sistema for sendo incrementado?	2, 3, 4
Rendimento / velocidade / taxa de transferência	22. Quantidade mínima de cálculos por determinado tempo?		22.1. Quais serão os parâmetros (tipo de cálculo X tempo)?	3, 5
Desempenho (tempo de resposta)	23. Tempo de resposta mínimo do sistema?	23.1. Ao interagir com o touch, o sistema deve responder em no máximo quantos milissegundos?	23.1.1. Como o sistema irá lidar com situação de delay de resposta?	1, 2, 3
Acurácia	24. Capacidade máxima de processamento o tempo todo?		24.1. O sistema deve sempre trabalhar com 100% de sua capacidade de memória e processamento?	3, 5

Comportamento em relação aos recursos	25. Uso de recursos fixo ou variável?	25.1. Como será feito o controle de uso de recursos?	3, 5

4.1.4 *Manutenção / Suporte*

Tabela 6:Requisitos não funcionais de Manutenção/Suporte (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Modificabilidade / estabilidade	26. Estabilidade mínima para modificações futuras?		26.1. Qual o número máximo de erros aceitável (por mil linhas de código)?	-
Analisabilidade	27. Como será feita a identificação de falhas?	27.1. O usuário será capaz de reportar erros?	27.1.1. Como será feita a análise dos erros reportados pelo usuário? 27.1.2. Será obtido um log?	-
Extensabilidade	28. Deve receber atualizações?	28.1. O usuário será notificado dessa atualização? 28.2. O usuário será forçado a atualizar o sistema?	28.1.1. Como será feita a instalação dessas atualizações? 28.2.1. O sistema irá atualizar por conta própria?	-
Gerenciamento de falhas	29. Devem ser gerados logs em caso de falhas?	29.1. O usuário será capaz de visualizar log de erro?	29.1.1. Este log será enviado para a nuvem? 29.1.2. Este log será armazenado no BD?	-
Gerência de configuração	30. Como será feito o controle de versões?	30.1. O usuário será capaz de visualizar qual versão está usando?	30.1.1. O sistema deve permitir o uso de até quantas versões anteriores?	-
Possibilidade de serviço	31. Devem ocorrer atualizações automáticas?	31.1. O usuário poderá usar o aplicativo enquanto ele é atualizado?	31.1.1. Ao terminar a atualização, o sistema será reiniciado?	-
Testabilidade	32. Taxa mínima de cobertura de testes automatizados no sistema?		32.1. Qual deve ser essa taxa?	-

Modularidade	33. Terá estrutura modular?	33.1. Quantos módulos? 33.2. Qual a comunicação entre um módulo com cada outro módulo?	-
Reusabilidade	34. Um módulo será considerado o padrão, de onde os demais serão obtidos?	34.1. Qual módulo será este?	-

4.1.5 Portabilidade

Tabela 7: Requisitos não funcionais de Portabilidade (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Adaptabilidade	35. Deve ser capaz de funcionar em diferentes versões do Android? 36. Deve ser capaz de funcionar em diferentes modelos de multimídia? 37. Deve ser capaz de funcionar em diferentes capacidades de memória RAM?		35.1. Quais versões? 36.1. Algum modelo em específico será desconsiderado? 37.1. Qual o mínimo de memória RAM necessária? 37.2. Qual o impacto dessa diferença na usabilidade?	4, 5
Facilidade de substituição	38. Diferentes versões devem ser compatíveis?		38.1. Como é definida essa compatibilidade?	-
Facilidade de instalação	39. Nível de facilidade de instalação?	39.1. É necessária intervenção do usuário?	39.1.1. Onde o sistema estará disponível para download?	-

4.1.6 Usabilidade

Tabela 8: Requisitos não funcionais de Usabilidade (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Acessibilidade	40. Deve ser acessível para pessoas com deficiência?	40.1. Quais tipos de deficiência serão levados em consideração na modelagem?	40.1.1. O sistema deve possuir descrição de conteúdo? 40.1.2. O sistema deve possuir auxílio visual? (Exemplo: aumentar o tamanho do texto) 40.1.3. O sistema deve possuir esquema de cores especiais para daltônicos?	1, 2, 3, 4, 5
Ajuda on-line e contextual	41. Deve possuir apoio on-line?	41.1. Como o usuário deve acessar este apoio?	41.1.1. Este apoio deve estar disponível quantas horas de um dia? 41.1.2. Como deve ser implementado este apoio?	-
Assistentes e agentes	42. Deve haver pessoas disponíveis, virtualmente, para prestar apoio?		42.1. Como será feita a conexão entre as pessoas e o usuário?	-
Consistência na interface do usuário	43. Quais diretrizes devem ser seguidas na criação da interface do usuário? (Exemplo: Conceito de janelas e menus gráficos)	43.1. Como o usuário irá interagir com a interface?	43.1.1. Será feita uma distinção de diretrizes entre diferentes menus?	1, 2, 3, 4, 5

Estética / fatores humanos	44. Quais diretrizes devem ser seguidas na aparência do software? (Exemplo: Fonte Arial 12) 45. Qual o esquema de cores deve ser seguido? (Exemplo: Azul claro e cinza)	44.1. O usuário deverá ser capaz de configurar a aparência do software? 45.1. O usuário deverá ser capaz de personalizar as cores do sistema?	44.1.1. O sistema deve permitir a alteração definitiva para todos os menus? 44.1.2. Como serão armazenadas as preferências do usuário? 45.1.1. Haverá uma lista pré-determinada de opções para o usuário escolher? 45.1.2. Deve haver uma validação das cores da fonte (branco ou preto) dependendo da cor dos campos?	1, 2, 3, 4, 5
Apreendibilidade	46. Deve existir um tutorial ensinando o usuário a usar o sistema? 47. Como será formulado o conteúdo deste tutorial?	46.1. O usuário conseguirá acessar este tutorial diversas vezes?	46.1.1. Como será feita a validação se o usuário já fez o tutorial? 46.1.2. Como deve ser implementada a opção de restauração do tutorial? 47.1. O sistema poderá receber atualizações futuras em seu tutorial?	1, 2, 3, 4

Inteligibilidade	48. Devem ser implementados auxílios fixos? (Exemplo: Tooltips) 49. Como serão identificados pontos que necessitam auxílio?	48.1. Como serão identificados esses auxílios para o usuário?	48.1.1. O sistema deve ser capaz de desabilitar estes auxílios? 48.1.2. Como será feita a implementação destes auxílios? 49.1. Será realizada uma implementação distinta para diferentes tipos de auxílio?	1, 2, 3
Manual do usuário	50. Há necessidade de um manual do usuário? 51. Como será formulado este manual?	50.1. Como o usuário irá acessar este manual?	50.1.1. Como será implementado este manual? 51.1. Este manual poderá receber atualizações futuras?	1, 2, 3
Materiais de Treinamento	52. Haverá material de treinamento?	52.1. Como este material poderá ser acessado pelo usuário?	52.1.1. Como será implementado este manual?	1, 2, 3

Múltiplo (suportar múltiplas empresas e moedas)	53. Deve haver suporte para diferentes moedas? 54. Deve haver suporte para diferentes unidades de medida? (Exemplo: Libra, pés, polegadas)	53.1. Como o usuário poderá alterar a moeda do sistema? 54.1. Como o usuário poderá alterar as unidades de medida do sistema?	53.1.1. Como será feita a implementação de bibliotecas para diferentes moedas? 53.1.2. Alguma fórmula de cálculo será específica para cada tipo de moeda? 54.1.1. Como será feita a implementação de bibliotecas para diferentes unidades de medida? 54.1.2. Alguma fórmula de cálculo será específica para cada tipo de unidade de medida?	2, 3, 4, 5
Internacionalização	55. O sistema deve oferecer suporte para diversas línguas? 56. Qual será a língua padrão do sistema?	55.1. Como o usuário poderá alterar a língua do sistema? 56.1. Como o usuário poderá resetar a língua do sistema para a original?	55.1.1. Como será feita a implementação de bibliotecas para diferentes línguas? 56.1.1. Como será implementada a opção de retornar a língua para a original?	1, 2, 3, 4, 5
Operabilidade / operacionalidade	57. Deve haver consideração pela operabilidade do sistema?	57.1. É possível alterar essa configuração?	57.1.1. Como serão identificadas quais as operações que devem estar em um mesmo menu?	1, 2, 3, 4, 5

Proteção contra erros	58. Será implementada proteção contra erros do usuário no sistema? 59. O BD deve ser protegido contra erros do usuário?	58.1. O usuário será informado que está cometendo um erro? 58.2. O usuário será instruído a como não cometer este erro? 58.3. Será claro para o usuário qual é o erro?	58.1.1. Como será implementada essa proteção? 58.2.1. Como será implementada essa instrução ao usuário? 58.3.1. Como será destacado para o usuário qual o erro? 59.1. Como será implementada essa proteção no BD?	1, 2, 3, 4

4.1.7 Segurança

Tabela 9: Requisitos não funcionais de Segurança (fonte: o Autor)

Requisitos Não Funcionais	*Requisitos de Negócio*	*Requisitos de Usuário*	*Requisitos de Sistema*	*Caso de Uso*
Auditoria e controle	60. Haverá auditoria?			-
Integridade	61. O sistema deve ser íntegro?		61.1. Como será implementada a prevenção de acessos e modificações ao sistema?	-
Contestabilidade e responsabilização / Cronologia (logs)	62. Serão implementados logs?	62.1. O usuário terá acesso a estes logs?	62.1.1. Como será feita a implementação destes logs?	-
Autenticidade	63. Deve haver verificação de autenticidade de recursos?		63.1. Como será feita essa verificação de autenticidade?	-
Confidencialidade / Controle de acesso / Registro de usuário	64. Haverá controle de diferentes usuários?	64.1. Como um usuário será identificado? 64.2. Como será feita a alteração de usuários?	64.1.1. Como será implementado um identificador único para cada usuário? 64.2.1. Como será implementada essa troca de usuário?	1, 2, 3, 4, 5
Confidencialidade / Controle de acesso / Autenticação de usuário	65. Haverá necessidade de senha para acesso?	65.1. Como o usuário irá definir sua senha? 65.2. Como o usuário irá inserir sua senha? 65.3. Como o usuário poderá redefinir sua senha? 65.4. Qual será a senha padrão?	65.1.1. Como será armazenada esta senha? 65.1.2. Como será feita a validação de senha? 65.3.1. Como será feito o reset de senha?	1

Confidencialidade / Controle de acesso / Autorização específica	66. Haverá necessidade de diferentes níveis de acesso entre usuários? 67. Diferentes usuários podem ter acesso a dados dos demais usuários?	66.1. Quais acessos cada nível deve ter? 67.1. Quais dados podem ser acessados? 67.2. Os dados de um usuário podem ser alterados por outro usuário?	66.1.1. Como será feita a implementação destes acessos? 67.1.1. Como será feito o controle de acesso destes dados? 67.2.1. Como será feito o controle de integridade destes dados?	1
Confidencialidade / Controle de acesso / Autorização configurável	68. Haverá personalização de acesso?	68.1. Qual usuário poderá personalizar os acessos?	68.1.1. Como será implementada essa personalização?	1
Aprovação	69. Alguma ação necessitará de aprovação? (Exemplo: Exclusão de registros)	69.1. Como será identificada essa aprovação para o usuário? 69.2. Qual usuário terá o poder de aprovação? 69.3. Em caso de não aprovação, como será feita essa notificação ao usuário que solicitou?	69.1.1. Como será feita a implementação de um usuário com poder de aprovação? 69.2.1. Este usuário deve ser passivo de exclusão ou alteração? 69.3.1. O sistema deve gerar uma notificação?	1

4.2 UML - Casos de Uso e Tabelas de Estado do Sistema

O detalhamento de casos de uso é uma importante ferramenta para garantir que todos os requisitos do sistema foram compreendidos e serão atendidos na implementação. Além disso, ajuda a identificar problemas de design e usabilidade e permite que o desenvolvimento seja feito de forma iterativa e incremental, aprimorando o sistema em cada etapa.

Foram levantados Casos de Uso, em ambos os formatos completo e abstrato. O seguintes Casos de Uso serão detalhados doravante:

a) Cadastrar Usuário;

b) Registrar Abastecimento;

c) Verificar Consumo Atual;

d) Verificar Histórico de Consumo;

e) Verificar Previsão de Tempo e Distância até o Próximo Abastecimento.

Importante destacar que nos detalhamentos de cada Caso de Uso acima estão relacionados os requisitos não funcionais relacionados elicitados anteriormente.

4.2.1 Cadastrar Usuário

Tabela 10: Detalhamento de Caso de Uso: Cadastrar Usuário (Fonte: o Autor)

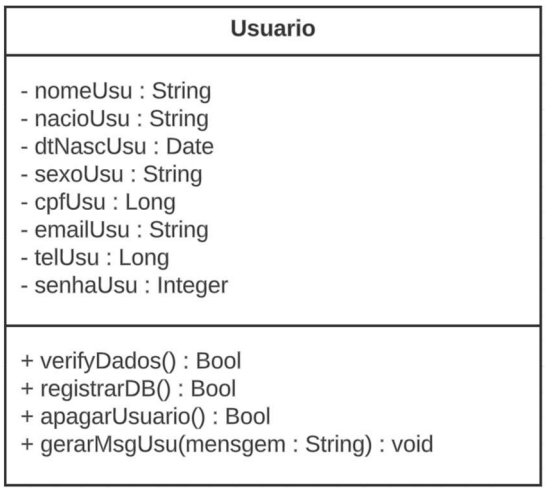
Ator principal	**Usuário**
Descrição resumida	O usuário irá se cadastrar no sistema, inserindo seus dados e registrando uma senha de acesso.
Pré-condições	-
Pós-condições	O usuário terá sido cadastrado no sistema, sendo autenticado por uma senha de sua escolha.
Cenário de Sucesso Principal	
1.	O usuário inicializa o sistema em sua multimídia veicular.
2.	O usuário seleciona a opção de cadastro de usuário na tela principal.
3.	O usuário insere os dados de cadastro e pressiona o botão de confirmação. (Dados obrigatórios para cadastro: Nome, país de origem, data de nascimento, sexo e CPF. Dados não-obrigatórios para cadastro: E-mail e telefone celular).
4.	O sistema solicita que o usuário crie uma senha (PIN - 5 dígitos).
5.	O sistema armazena o usuário criado em seu BD.
6.	É apresentada na tela uma mensagem de sucesso na criação de usuário, contendo o nome do usuário cadastrado.
Fluxos Alternativos:	
a)	Limite de usuários cadastrados atingido
3.	O sistema apresenta uma mensagem ao usuário informando que o limite de usuários foi atingido e sugere que seja apagando outro usuário já cadastrado antes de ser criado um e retorna ao passo 2 do cenário principal.
b)	Dados Inválidos
1.	O usuário insere dados inválidos (Exemplo: CPF inválido).

2.	O sistema informa que o dado inserido está incorreto e solicita que o usuário insira novamente e retorna ao passo 3 do cenário principal.
c)	Já existe algum usuário cadastrado com o mesmo CPF
4.	O sistema informa ao usuário que já existe um usuário cadastrado com o CPF informado e retorna ao passo 2 do cenário principal.
Requisitos não funcionais relacionados	13, 14, 15, 17, 23, 40, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 64, 65, 66, 67, 68 e 69.

Tabela 11: Tabela de Estados: Cadastrar Usuário (Fonte: o Autor)

Estado Atual	Entrada	Estado Seguinte	Saída
Verificando número de usuários cadastrados	Limite de usuários atingido	Aguardando confirmação do usuário	Mensagem de erro: "Limite de usuários atingido. Deseja excluir algum usuário já cadastrado? (SIM/NÃO)"
Verificando número de usuários cadastrados	Limite de usuários não atingido	Aguardando preenchimento do formulário de cadastro de usuário	-
Aguardando preenchimento do formulário de cadastro de usuário	Usuário preenche os campos de cadastro	Verificando os dados do usuário	-
Verificando os dados do usuário	Dados inválidos	Aguardando preenchimento do formulário de cadastro de usuário	Mensagem de erro: "Dados inválidos. Por favor, preencha novamente o(s) campo(s): NOME_CAMPO"
Verificando os dados do usuário	Dados válidos	Aguardando criação de senha	-
Aguardando criação de senha	Código numérico de 5 dígitos (PIN)	Salvando novo usuário	Mensagem de confirmação: "Usuário cadastrado com sucesso"
Salvando novo usuário	-	Menu Principal	-

Diagrama 1: Diagrama UML (Classe): Cadastrar Usuário (Fonte: o Autor)



4.2.2 Registrar Abastecimento

Tabela 12: Detalhamento de Caso de Uso: Registrar Abastecimento (Fonte: o Autor)

Principal	**Usuário**
Descrição resumida	O usuário irá registrar o abastecimento do veículo, inserindo os dados pertinentes.
Pré-condições	O usuário e o veículo já estão cadastrados no sistema.
Pós-condições	Será armazenado o registro de abastecimento na base de dados do sistema.
Cenário de Sucesso Principal	
1.	O usuário inicializa o sistema em sua multimídia veicular.
2.	O usuário seleciona a opção de registrar abastecimento na tela principal.
3.	O usuário insere os dados e pressiona o botão de confirmação. (Dados obrigatórios para registro: Quilometragem atual, Valor, Quantidade abastecida (Ex.: litros), se encheu o tanque e Tipo de combustível).
4.	O sistema registra a data e o horário da operação automaticamente.
5.	O sistema armazena o registro do abastecimento em seu BD.
6.	É apresentada na tela uma mensagem de sucesso no registro de abastecimento.
Fluxos Alternativos:	
a)	Armazenamento (DB) Cheio
1.	O sistema não consegue registrar o abastecimento devido ao armazenamento cheio.
2.	O sistema apaga o registro de abastecimento mais antigo.
3.	O sistema tenta registrar o novo abastecimento novamente.

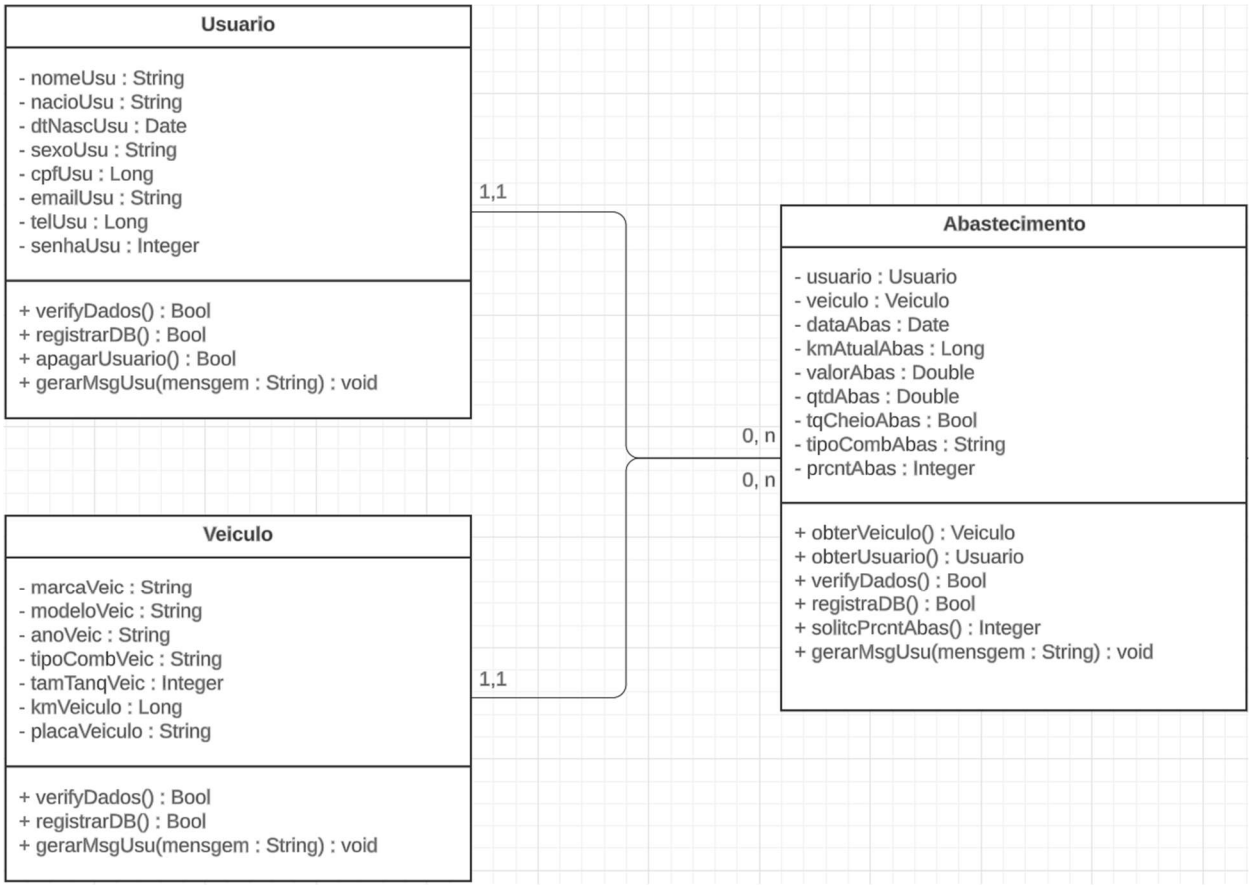
b) Abastecimento parcial

1.	O usuário não marcou que encheu o tanque.
2.	O sistema solicita que o usuário informe a % de combustível no tanque indicada no painel. As possibilidades padrão são de 10% a 90%, em intervalos de 10 em 10.
Requisitos não funcionais relacionados	9, 15, 16, 17, 18, 19, 21, 23, 40, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 e 64.

Tabela 13: Tabela de Estados: Registrar Abastecimento (Fonte: o Autor)

Estado Atual	Entrada	Estado Seguinte	Saída
Aguardando preenchimento dos campos de registro de abastecimento	Usuário preenche os campos de registro de abastecimento	Verificando os dados inseridos	-
Verificando os dados inseridos	Dados inválidos	Aguardando preenchimento dos campos de registro de abastecimento	Mensagem de erro: "Dados inválidos. Por favor, preencha novamente o(s) campo(s): NOME_CAMPO"
Verificando se o campo referente ao tanque cheio foi marcado	O campo não foi marcado	Solicitando que o usuário informe a % de combustível no tanque informada no painel.	-
Solicitando que o usuário informe a % de combustível no tanque informada no painel.	O usuário informa a %.	Verificando os dados inseridos.	-
Verificando se o campo referente ao tanque cheio foi marcado	O campo foi marcado.	Verificando os dados inseridos.	-
Verificando os dados inseridos	Dados válidos	Armazenando o registro de abastecimento	-
Armazenando o registro de abastecimento	-	Menu principal	Mensagem de confirmação: "Abastecimento registrado com sucesso"

Diagrama 2: Diagrama UML (Classe): Registrar Abastecimento (Fonte: o Autor)



4.2.3 Verificar Consumo Atual

Tabela 14: Detalhamento de Caso de Uso: Verificar Consumo Atual (Fonte: o Autor)

Ator principal	**Usuário**
Descrição resumida	O usuário irá verificar como está o consumo atual do veículo, baseado na última vez que abasteceu.
Pré-condições	Já existem pelo menos dois registros de abastecimento no sistema, o mais atual (último feito) e 1 anterior a este.
Pós-condições	Será apresentado ao usuário o cálculo de consumo do veículo baseado nos registros anteriores.
Cenário de Sucesso Principal	
1.	O usuário inicializa o sistema em sua multimídia veicular.

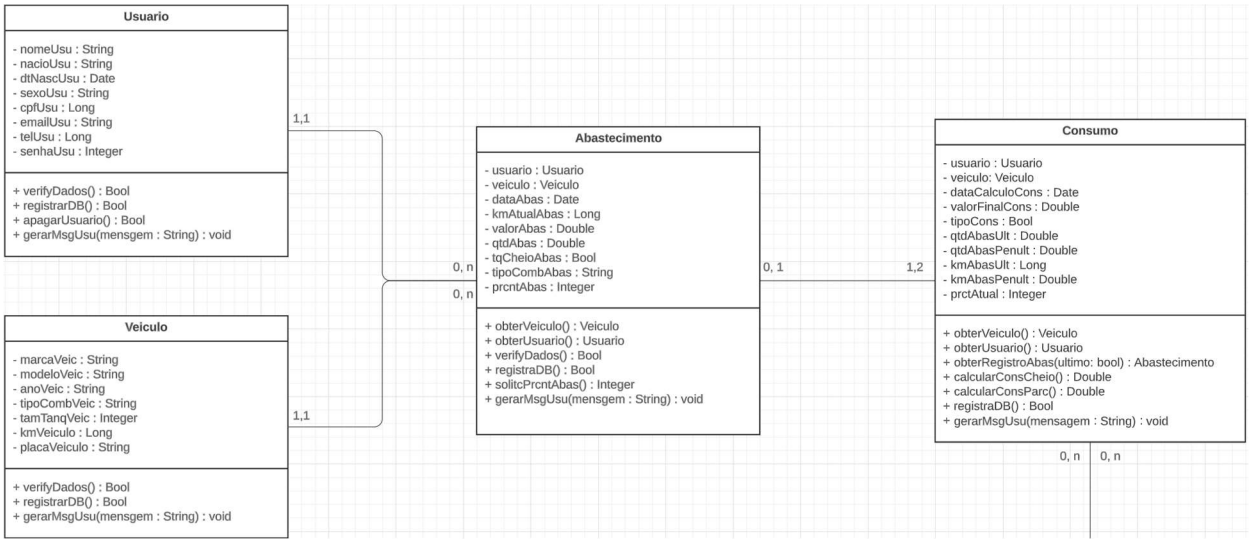
2.	O usuário seleciona a opção de verificar consumo atual na tela principal.
3.	O sistema verifica se os dois últimos registros de abastecimento foram de um tanque cheio.
4.	O sistema realiza o cálculo do consumo baseado nos registros anteriores. O cálculo será feito utilizando a diferença entre a quilometragem dos últimos dois registros e o volume de combustível consumido.
5.	O sistema apresenta este consumo ao usuário.
Fluxos Alternativos:	
a)	Não existem registros anteriores de abastecimento
1.	O sistema não tem dados anteriores de abastecimento para realizar o cálculo de consumo.
2.	O informa ao usuário que ainda não existem registros suficientes para realizar o cálculo e mostra quantos registros ainda faltam (1 ou 2).
b)	Os últimos registros de abastecimento não foram de um tanque cheio
1.	O sistema verifica que os dois últimos registros de abastecimento não foram de um tanque cheio.
2.	O sistema informa ao usuário que o cálculo será feito de forma aproximada, devido aos abastecimentos anteriores não terem sido tanque cheio.
3.	O sistema pergunta ao usuário como está o indicador de combustível no painel. As possibilidades padrão são de 10% a 90%, em intervalos de 10 em 10.
4.	O usuário informa a % de combustível no tanque indicada no painel.
5.	O sistema realiza o cálculo do consumo aproximado baseado nos registros anteriores. O cálculo será feito utilizando a diferença entre a quilometragem dos últimos dois registros e o volume aproximado de combustível consumido (baseado no tamanho do tanque do veículo e a % informada).
Requisitos não funcionais relacionados	3, 4, 5, 15, 16, 17, 21, 22, 23, 24, 40, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 e 64.

Tabela 15: Tabela de Estados: Verificar Consumo Atual (Fonte: o Autor)

Estado Atual	Entrada	Estado Seguinte	Saída
Verificando se existem registros suficientes para o cálculo	Não existem registros suficientes para o cálculo	Menu principal	Mensagem de erro: “Infelizmente não é possível calcular a média de consumo ”

Verificando se existem registros suficientes para o cálculo	Existem registros suficientes para o cálculo	Verificação do tipo de cálculo de consumo	-
Verificação do tipo de cálculo de consumo	Os dois últimos abastecimentos foram tanque cheio	Cálculo exato	-
Verificação do tipo de cálculo de consumo	Os dois últimos abastecimentos não foram de tanque cheio	Cálculo aproximado	-
Cálculo aproximado	-	Apresentação de resultados	-
Cálculo exato	-	Apresentação de resultados	-
Apresentação de resultados	-	Menu principal	Mensagem ao usuário: "Cálculo de consumo (aproximado/exato): VALOR_CALCULADO"

Diagrama 3: Diagrama UML (Classe): Verificar Consumo Atual (Fonte: o Autor)



4.2.4 Verificar Histórico de Consumo

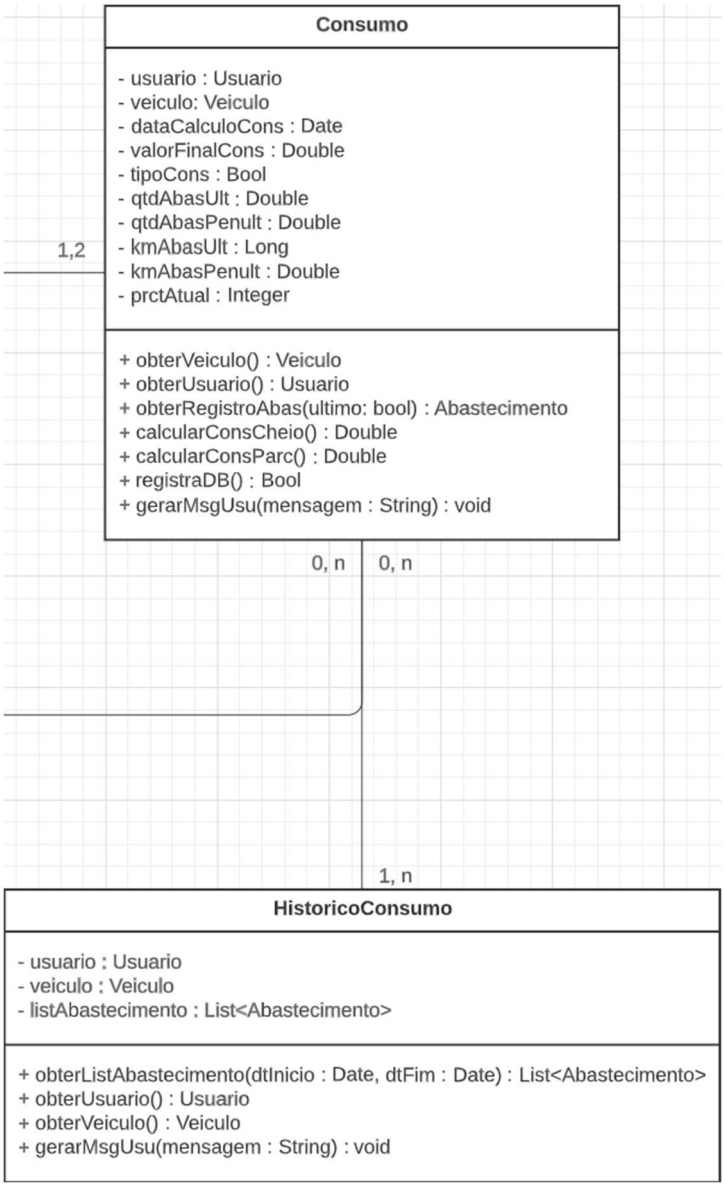
Tabela 16: Detalhamento de Caso de Uso: Verificar Histórico de Consumo (Fonte: o Autor)

Ator principal	**Usuário**
Descrição resumida	O usuário irá verificar o histórico de consumo do veículo, visualizando os registros anteriores armazenados.
Pré-condições	Já existem pelo menos um registro armazenado no sistema.
Pós-condições	Será apresentado ao usuário o histórico de consumo do veículo, listando os registros existentes no BD.
Cenário de Sucesso Principal	
1.	O usuário inicializa o sistema em sua multimídia veicular.
2.	O usuário seleciona a opção de verificar histórico de consumo na tela principal.
3.	O sistema verifica se existe pelo menos 1 registro de consumo.
4.	O sistema apresenta os registros encontrados em formato de lista, organizando por data.
Fluxos Alternativos:	
a)	Não existem registros anteriores de abastecimento
1.	O sistema não tem dados anteriores de consumo para apresentar ao usuário
2.	O sistema informa ao usuário que não existem registros para apresentar.
Requisitos não funcionais relacionados	3, 4, 5, 15, 16. 17, 21, 22, 23, 24, 40, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53. 54, 55, 56, 57, 58, 59 e 64.

Tabela 17: Tabela de Estados: Verificar Histórico de Consumo (fonte: o Autor)

Estado Atual	**Entrada**	**Estado Seguinte**	**Saída**
Verificando se existem registros de consumo para apresentar	Existem registros de consumo armazenados no BD	Listando registros encontrados	-
Verificando se existem registros de consumo para apresentar	Não existem registros de consumo armazenados no BD	Menu principal	Mensagem de erro: “Não foram encontrados registros de consumo.”
Listando registros encontrados	-	Menu principal	Lista de registros encontrados organizada em ordem cronológica

Diagrama 4: Diagrama UML (Classe): Verificar Histórico de Consumo (Fonte: o Autor)



4.2.5 Verificar Previsão de Tempo e Distância até o Próximo Abastecimento

Tabela 18: Detalhamento de Caso de Uso: Verificar Previsão de Tempo e Distância até o Próximo Abastecimento (Fonte: o Autor)

Ator principal	**Usuário**
Descrição resumida	O usuário irá verificar uma previsão de tempo e distância até a próxima vez que terá que abastecer.
Pré-condições	Existe um registro atual de consumo do veículo armazenado e pelo menos 2 registros de abastecimento (parcial ou tanque cheio).
Pós-condições	O usuário será capaz de visualizar uma previsão estimada de quantos quilômetros e quantos dias até a próxima vez que será necessário abastecer. As previsões terão sido armazenadas no BD.
Cenário de Sucesso Principal	
1.	O usuário inicializa o sistema em sua multimídia veicular.
2.	O usuário seleciona a opção de previsão de abastecimento na tela principal.
3.	O sistema verifica se existe um registro atual de consumo no BD.
4.	O sistema realiza o cálculo do tempo estimado para o próximo abastecimento baseado nos registros anteriores. O cálculo será feito utilizando a média de intervalo de tempo entre cada abastecimento.
5.	O sistema realiza o cálculo da quilometragem estimada para o próximo abastecimento baseado no registro anterior. O cálculo será feito utilizando a média da diferença quilometragem entre os abastecimentos anteriores.
6.	O sistema armazena os valores calculados no BD.
7.	O sistema apresenta estas estimativas ao usuário.
Fluxos Alternativos - Não existem registros suficientes	
1.	O sistema não tem dados anteriores de abastecimento para realizar o cálculo das previsões.
2.	O informa ao usuário que ainda não existem registros suficientes para realizar o cálculo e mostra quantos registros ainda faltam (1 ou 2).
Requisitos não funcionais relacionados	3, 4, 5, 8, 13, 14, 15, 16, 17, 18, 19, 20, 22, 24, 25, 35, 36, 37, 40, 43, 44, 53, 54, 55, 56, 57 e 64.

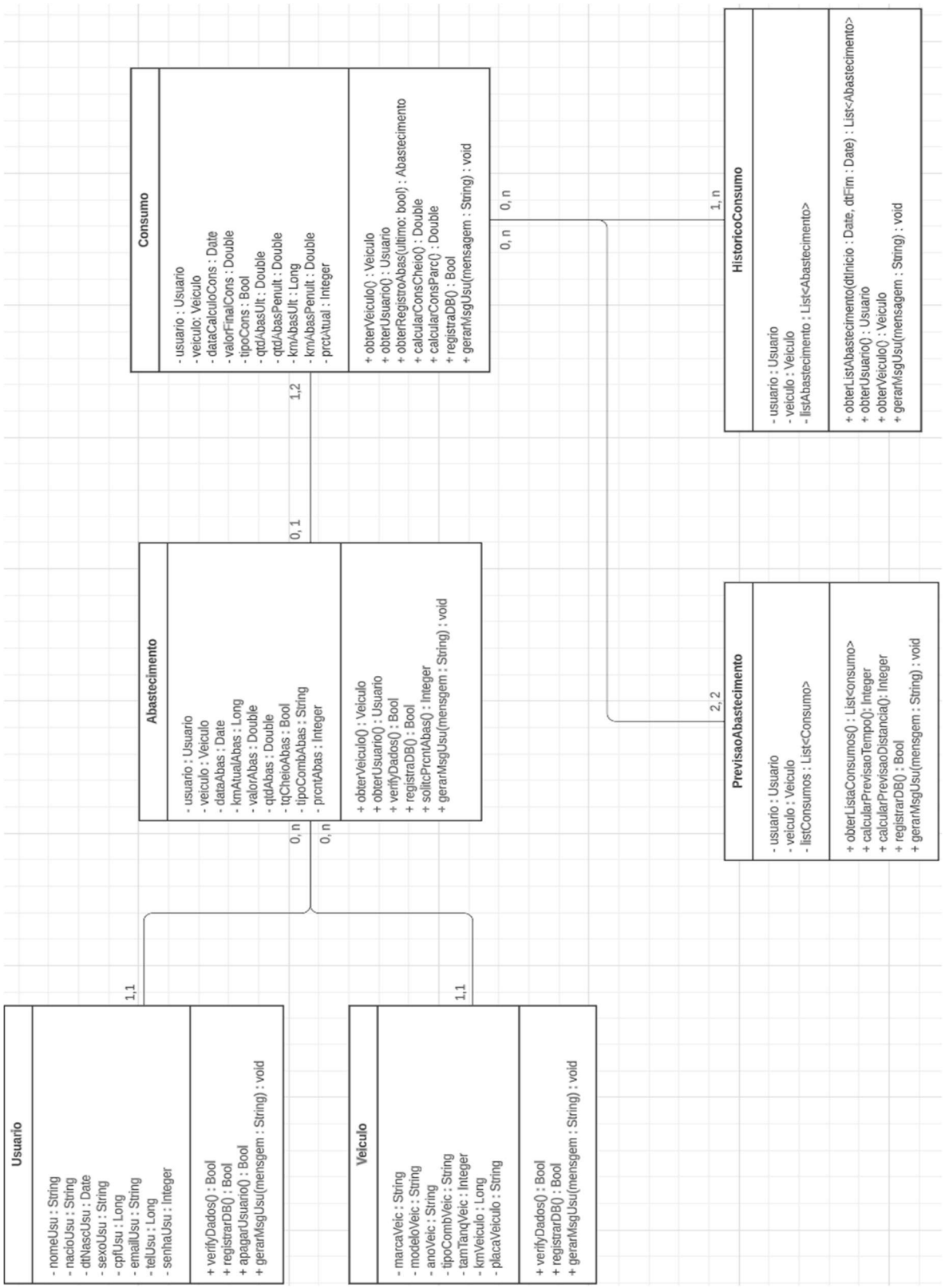
Tabela 19: Tabela de Estados: Verificar Previsão de Tempo e Distância até o Próximo Abastecimento (Fonte: o Autor)

Estado Atual	**Entrada**	**Estado Seguinte**	**Saída**
Verificando se existem registros de abastecimento para cálculo de previsão	Existem registros suficientes armazenados no BD	Cálculo de previsão	-

Verificando se existem registros de abastecimento para cálculo de previsão	Não existem registros suficientes armazenados no BD	Menu principal	Mensagem de erro: "Não foram encontrados registros suficientes para cálculo de previsão. Para que o cálculo seja realizado é necessário mais (1 registro / 2 registros)"
Cálculo de previsão	-	Apresentação de resultados	-
Apresentação de resultados	-	**Menu principal**	**Apresentação dos campos de previsão para o próximo abastecimento: Data e Distância**

Diagrama 5: Diagrama UML (Classe): Verificar Previsão de Tempo e Distância até o Próximo Abastecimento

(Fonte: o Autor)



5 IMPLEMENTAÇÃO

Esse Capítulo tem como objetivo o esclarecimento geral sobre os artefatos e funcionamento do software “FuelLog”, na forma de um aplicativo Android de controle de consumo de combustível para veículos de pequeno e médio porte.

Serão abordados os conceitos técnicos presentes no desenvolvimento do software, como arquitetura, regras de negócio, bibliotecas utilizadas, testes etc. O detalhamento destes conceitos envolve uma explicação breve sobre o código implementado, assim como um fragmento do próprio código fonte sendo citado dentro do contexto explicado.

Com o intuito de tornar a leitura desse documento mais dinâmica e facilitar futuras consultas a ele, os diferentes conceitos técnicos serão separados por tópicos. A sequência determinada para os tópicos é baseada no nível técnico do conceito abordado, começando por suas telas e finalizando com testes planejados.

O foco do aplicativo FuelLog é oferecer uma ferramenta de cálculo de consumo de combustível que seja fácil de usar e que seja útil ao motorista. Logo, ele foi pensado e desenvolvido para que o software possa ser utilizado em multimídias veiculares Android. O FuelLog apresenta telas simples e diretas, com ícones grandes, cores contrastantes e clareza de informações.

Para referência na criação de layouts, foi escolhido o tablet Nexus 9, da empresa HTC. Este modelo foi selecionado devido à suas características similares àquelas encontradas em multimídias veiculares, sendo, principalmente, uma tela de 9 polegadas e uma versão antiga do Android (8.1 Oreo). É importante salientar que o aplicativo irá funcionar normalmente em outros aparelhos, independente do tamanho de sua tela, desde que sua versão do Android não seja inferior à versão 8.1 Oreo.

5.1 Definições de Arquitetura, Linguagem e Bibliotecas

Após definir-se que a plataforma de escolha para o software seria o sistema operacional Android, o sistema foi idealizado em forma de aplicativo. A partir disso foi feita a eleição da arquitetura implementada, da linguagem principal utilizada para o desenvolvimento do software e das bibliotecas necessárias para sua implementação.

Tratando-se de um sistema com 3 partes, sendo delas, leitura e inserção de dados, regras de negócio e apresentação de informações, a arquitetura escolhida foi a MVC. A arquitetura Model-View-Controller (MVC) é especialmente útil em aplicações mobile, onde o front-end e o back-end, muitas

vezes, estão presentes na mesma aplicação. Em poucas palavras, essa arquitetura é implementada de forma que o Controller é um intermediário entre os dados (Model) e o que é apresentado ao usuário (View), fazendo com que a manutenção e a escalabilidade sejam bem mais fáceis.

Por padrão, a linguagem de programação nativa para Android é Java, logo foi a escolhida. Sendo uma linguagem bem difundida, existe um acervo grande de informações, bibliotecas e ferramentas que podem ser, e foram, utilizadas durante o desenvolvimento.

Como exemplo, fez-se o uso de um banco de dados SQLite, em conjunto com a biblioteca SQLiteOpenHelper. Isso tornou o acesso, a leitura, a alteração e o tratamento dos dados bem mais simples e direto. Como parte da biblioteca SQLiteOpenHelper, tem-se os métodos “execSQL” e “getWritableDatabase”, utilizados para executar linhas de código escritas em SQL e realizar a conexão com a base de dados, respectivamente.

5.2 Principais Interfaces Homem-Máquina

5.2.1 Menu Principal

Esse é o menu principal do aplicativo. Nele encontramos o nome do usuário no canto esquerdo superior, configurações no canto direito superior e os 4 menus disponíveis na primeira versão do aplicativo.

Os menus são:

a) Registrar Abastecimento,

b) Próximo Abastecimento,

c) Consumo Atual, e

d) Histórico.

Imagem 1 – Menu Principal

Fonte: o Autor

5.2.2 Registrar Abastecimento

A tela Registrar Abastecimento é a porta de entrada de dados do usuário para o sistema. Utilizando esse menu o usuário irá inserir as informações do abastecimento que está realizando, sendo elas:

a) Km do Veículo,

b) Valor,

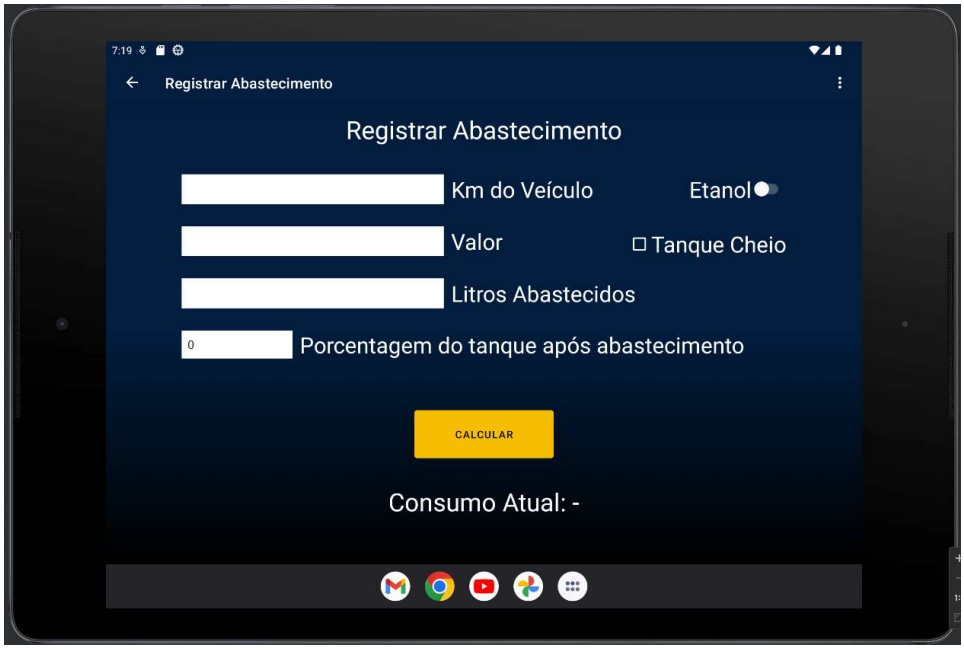
c) Litros Abastecidos,

d) Tipo de Combustível (existem dois diferentes tipos de combustível, Etanol e Gasolina, e duas diferentes formas de abastecimento, Tanque Cheio ou não.),

e) Tanque Cheio, e

f) Porcentagem do Tanque.

Tela 2 – Registrar Abastecimento



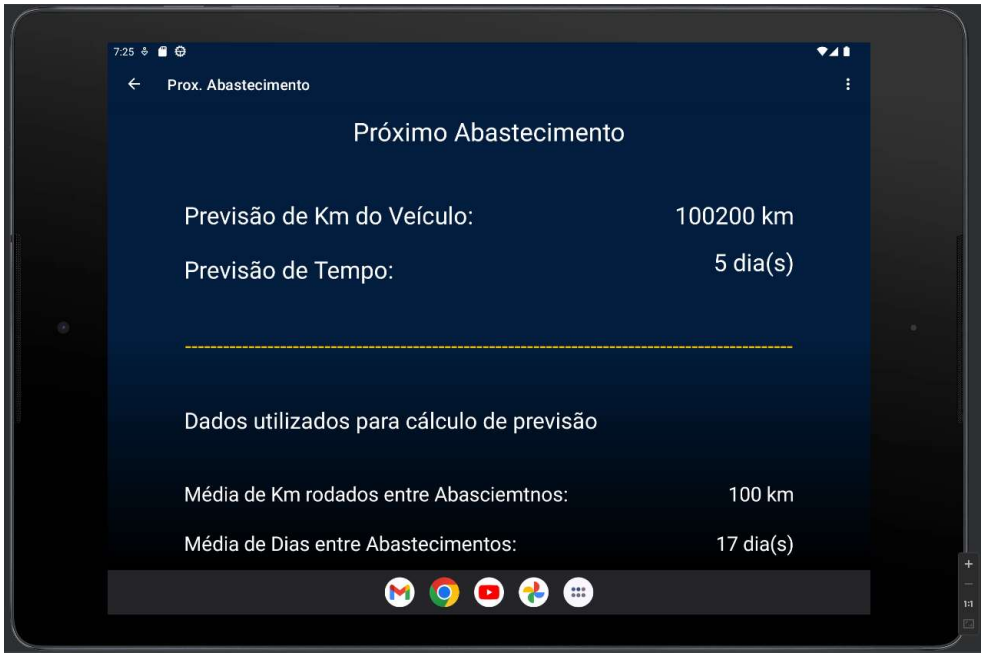
Fonte: o Autor

Conforme o cenário inserido pelo cliente, a tela poderá mudar para ocultar ou apresentar o campo Porcentagem e atualizará o valor do Consumo Atual, no rodapé.

5.2.3 Próximo Abastecimento

Nessa tela o usuário irá consultar informações relativas ao próximo abastecimento, conforme o cálculo do aplicativo baseado em registros anteriores. Este cálculo é apresentado de duas formas, em tempo e em Km do Veículo, sendo que ambos os resultados necessitam de ao menos dois registros de abastecimento anteriores para que sejam apresentados.

Imagem 3 – PRÓXIMO ABASTECIMENTO



Fonte: o Autor

5.2.4 Consumo Atual

Baseando-se em registros de abastecimentos presentes no banco de dados, esse menu irá apresentar os dados de média de consumo. Os campos apresentados são:

a) Média atual,

b) tipo de combustível,

c) data que foi realizado o cálculo,

d) o tipo (Exato ou Aproximado), e

e) uma barra visual para qualificação dos resultados.

Imagem 4 – CONSUMO ATUAL

Fonte: o Autor

5.2.5 Histórico

O objetivo da tela de Histórico é fornecer ao usuário uma lista de registros, para que ele possa manter um controle sobre variações no consumo calculado, os tipos de consumo e as datas que tais consumos foram calculados.

Imagem 5 – HISTÓRICO

Fonte: o Autor

5.3 Implementação dos Indicadores Principais

Visando contemplar diferentes cenários do mundo real, foram criadas 4 (quatro) diferentes fórmulas de cálculo de média. Essas fórmulas se agrupam em dois tipos, EXATA e APROXIMADAS.

- **EXATA**: Fórmula que utiliza dois registros de abastecimento com o tanque cheio para o cálculo, resultando em uma estimativa precisa do consumo de combustível do veículo.
- **APROXIMADAS**: Fórmulas que utilizam um ou dois registros de abastecimento que não foram tanque cheio. Isso torna o resultado uma aproximação, uma vez que se perde o controle da quantidade exata de combustível consumida no período calculado.

5.3.1 Cálculos de Médias Gerais

Para a fórmula EXATA, o cálculo matemático é:

Média = (KmAtual – KmAnterior) / LitrosConsumidos.

E assim é sua implementação no código:

```
if (anterior.getTanqueCheio() == 1 && atual.getTanqueCheio() == 1) {
    consumoCalculado = ((atual.getKmAtual() - anterior.getKmAtual()) / atual.getQuantidadeLitros());
    tipo = "EXATO";
```

Para as fórmulas APROXIMADAS, os cálculos matemáticos são:

Média = (KmAtual – KmAnterior) / (EstLitrosAtual – EstLitrosAnterior)

Média = (KmAtual – KmAnterior) / (TanqueCheio - EstLitrosAtual)

Média = (KmAtual – KmAnterior) / (LitrosAbastecidos – EstLitrosAnterior)

E assim são suas implementações no código:

```
} else if (anterior.getTanqueCheio() != 1 && atual.getTanqueCheio() != 1) {
    consumoCalculado = ((atual.getKmAtual() - anterior.getKmAtual())/ ((tamTanque()*atual.getPercentualTanque()/100) - (tamTanque()*anterior.getPercentualTanque()/100)));

} else if (anterior.getTanqueCheio() == 1 && atual.getTanqueCheio() != 1) {
    consumoCalculado = ((atual.getKmAtual() - anterior.getKmAtual())/ (tamTanque() - (tamTanque()*atual.getPercentualTanque()/100)));

} else {
    consumoCalculado = ((atual.getKmAtual() - anterior.getKmAtual())/ (atual.getQuantidadeLitros() - (tamTanque()*anterior.getPercentualTanque()/100)));
```

5.3.2 Cálculo das previsões de abastecimento

Para o cálculo das previsões de abastecimento, são utilizados os dados de abastecimentos já existentes para se medir os intervalos de tempo e distância entre os abastecimentos. Utilizando desse intervalo, é possível se calcular uma média de dias e de quilômetros entre um abastecimento e outro.

Dada a natureza relativa destes dados, essas previsões têm como objetivo serem mais um lembrete do que um aviso. Para que o cálculo das previsões pudesse ser exato, seria necessário obter dados em tempo real da quantidade de combustível no tanque e do consumo do veículo. Considerando que os veículos mais modernos possuem essa funcionalidade integrada em seu computador de bordo, não se tem como objetivo a substituição de tal ferramenta.

5.3.3 Cálculo das médias de distância e tempo

O cálculo das médias foi implementado da seguinte forma:

```
previsao.setMediaKm((mediaKm.get(mediaKm.size()-1) - mediaKm.get(0))/(mediaKm.size()-1));
previsao.setMediaDia(LocalDate.parse(mediaDia.get(0), formatter).until(LocalDate.parse(mediaDia.get(mediaDia.size()-1), formatter)).getDays());
previsao.setDataCalculo(getDataAtual());
previsao.setDataPrevista(LocalDate.parse(mediaDia.get(mediaDia.size()-1), formatter).until(LocalDate.parse(getDataAtualPrevisao(), formatter)).getDays());
previsao.setKmPrevisto(mediaKm.get(mediaKm.size()-1) + previsao.getMediaKm());
```

O que este bloco de código realiza é o cálculo de média de distância e tempo, baseando-se em registros antigos, e utiliza essa média para estimar a próxima kilometragem do veículo no abastecimento e quantos dias até que esse abastecimento ocorra.

5.4 Testes

Tratando-se de um aplicativo que depende diretamente de registros em seu banco de dados, como parte da estratégia de testes desenvolvida, foram eleitas as ferramentas Mockito e AVD.

- Mockito: O seu nome vem da palavra “Mock”, que, traduzida do inglês, significa algo como “disfarce”. Esse framework utiliza da injeção de dados fictícios como uma forma de validação do fluxo, ou seja, torna dispensável a inserção manual por parte de um usuário. O seu uso para aplicações desenvolvidas em Java é muito comum, uma vez que se consegue isolar frações do fluxo e testar tais frações individualmente e intensamente.

- **AVD**: Sendo parte da IDE (interface de desenvolvimento) oficial do Android, o Android Studio, a AVD é nada mais do que um emulador de dispositivos Android. Dentro das opções de configuração dos dispositivos virtuais, é possível controlar a versão do sistema operacional, sua capacidade de memória RAM, seu espaço interno e todas as configurações comuns de um aparelho Android. Isso torna a validação da implementação muito mais ágil e eficaz, por tornar visual as alterações e permitir *debugar* em tempo real a execução do código.

5.5 Roadmap

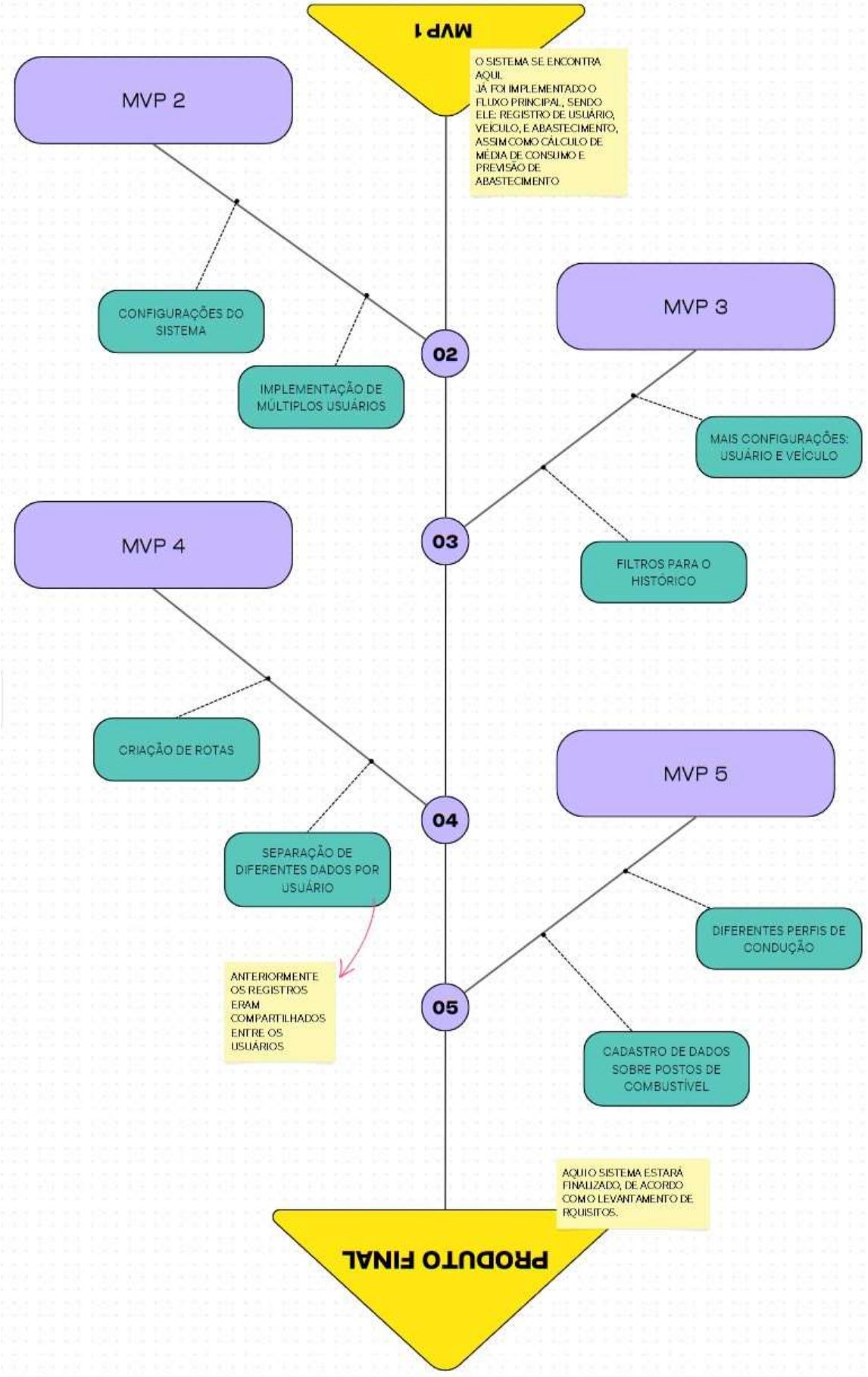
Conforme estabelecido previamente, o sistema será desenvolvido em partes, as chamadas "MVP's". A sigla que significa Produto Mínimo Viável refere-se a versões funcionais do sistema, que podem ser desenvolvidas e anexadas gradualmente, não sendo necessário finalizar todo o escopo do projeto de uma única vez.

Considerando que a entrega da primeira versão do aplicativo FuelLog representa a MVP 1, foi criado um roadmap demonstrando o planejamento de recursos e funcionalidades para cada versão futura. É importante salientar que essa primeira versão é simples, porém completamente funcional. Em futuras MVP's serão implementados outros requisitos previamente levantados, ou seja, mesmo que não estejam presentes nessa MVP 1, não foram desconsiderados ou descartados.

Para expandir o roadmap do aplicativo FuelLog, pode-se considerar a estruturação em fases incrementais. Cada fase, ou MVP (Produto Mínimo Viável), introduzirá novas funcionalidades baseadas em feedback dos usuários e análises de desempenho. A primeira versão, já funcional, satisfez os requisitos essenciais. Para as próximas versões, propõe-se planejar a inclusão de recursos como:

- MVP 2: Implementação de análise preditiva para estimativa de consumo.
- MVP 3: Integração com mapas para otimização de rotas.
- MVP 4: Recursos de gamificação para engajamento do usuário.

Cada etapa traz melhorias contínuas, garantindo que o aplicativo permaneça atualizado e alinhado às necessidades dos usuários. É crucial que cada MVP seja lançada após rigorosos testes para garantir a estabilidade e a confiabilidade do sistema.


MVP 1
O SISTEMA SE ENCONTRA AQUI.
JÁ FOI IMPLEMENTADO O FLUXO PRINCIPAL, SENDO ELE: REGISTRO DE USUÁRIO, VEÍCULO, E ABASTECIMENTO, ASSIM COMO CÁLCULO DE MÉDIA DE CONSUMO E PREVISÃO DE ABASTECIMENTO
MVP 2
CONFIGURAÇÕES DO SISTEMA
IMPLEMENTAÇÃO DE MÚLTIPLOS USUÁRIOS
02
MVP 3
MAIS CONFIGURAÇÕES: USUÁRIO E VEÍCULO
03
FILTROS PARA O HISTÓRICO
MVP 4
CRIAÇÃO DE ROTAS
04
SEPARAÇÃO DE DIFERENTES DADOS POR USUÁRIO
ANTERIORMENTE OS REGISTROS ERAM COMPARTILHADOS ENTRE OS USUÁRIOS
MVP 5
DIFERENTES PERFIS DE CONDUÇÃO
05
CADASTRO DE DADOS SOBRE POSTOS DE COMBUSTÍVEL
AQUI O SISTEMA ESTARÁ FINALIZADO, DE ACORDO COM O LEVANTAMENTO DE REQUISITOS.
PRODUTO FINAL

6 Considerações Finais

Este trabalho, orientado na Pontifícia Universidade Católica de Goiás, culminou no desenvolvimento do aplicativo Android 'FuelLog', visando a otimização do consumo de combustível e a redução de emissões poluentes.

O planejamento da execução do projeto proposto no TCC1 foi orientado pelos objetivos gerais de promover a sustentabilidade e eficiência energética, e aos objetivos específicos de aplicar práticas de engenharia de software e modelagem UML na criação de uma solução tecnológica inovadora.

O desenvolvimento do aplicativo 'FuelLog' se alinhou eficientemente aos objetivos específicos estabelecidos no TCC1:

a) Desenvolvimento de um aplicativo Android para gestão do consumo de combustível: Realizado com sucesso, conforme evidenciado na documentação e prototipagem do aplicativo.

b) Aplicação de técnicas de análise de requisitos baseadas na ISO 25010: Implementadas integralmente, garantindo a qualidade e a usabilidade do aplicativo.

c) Uso de modelagem em casos de uso por UML: Demonstrado nos capítulos de modelagem e design do aplicativo, fornecendo uma base sólida para o desenvolvimento.

d) Promoção de eficiência energética e redução de emissão de gases: Alcançado através das funcionalidades do aplicativo, que incentivam um uso mais consciente do combustível.

e) Implementação de práticas de engenharia de software sustentáveis: Evidenciado pelo processo de desenvolvimento disciplinado e pela atenção à sustentabilidade durante todo o projeto.

Esses objetivos foram cumpridos ao longo do desenvolvimento do projeto, demonstrando a eficácia do aplicativo em atender às necessidades dos usuários e contribuir para uma condução mais sustentável. Os próximos passos incluem aprimoramentos contínuos e expansão das funcionalidades do FuelLog para manter sua relevância e eficácia.

Os resultados demonstraram que o FuelLog atende às necessidades dos usuários ao fornecer um controle eficiente sobre o consumo de combustível, contribuindo significativamente para a conscientização ambiental e a economia operacional. A utilização da ISO 25010 e da modelagem UML garantiu um desenvolvimento estruturado e de alta qualidade.

Para futuras implementações, recomenda-se a integração do aplicativo com sistemas de navegação mais avançados, a inclusão de recursos de aprendizado de máquina para previsões mais precisas do consumo de combustível, e a expansão para outras plataformas. Este passo assegurará que o FuelLog continue evoluindo em resposta às demandas tecnológicas e às necessidades dos usuários, mantendo sua relevância e eficácia em um mercado em constante mudança.

REFERÊNCIAS

ANFAVEA. Cenário da Frota Automotiva Brasileira 2021. São Paulo, 2021. Disponível em: https://www.anfavea.com.br/cenario-da-frota-automotiva-brasileira-2021/. Acesso em: 06 mai. 2023.

CAST. CRASH Report: O Relatório de Qualidade de Software de 2021. Nova York: CAST, 2021. Disponível em: https://www.castsoftware.com/wp-content/uploads/2021/05/crash-report-2021.pdf. Acesso em: 06 mai. 2023.

ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Geneva: International Organization for Standardization, 2011.

LAVILLE, Christian; DIONNE, Jean. A construção do saber: manual de metodologia da pesquisa em ciências humanas. Porto Alegre: Artmed, 2020.

LARMAN, Craig. Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos e ao Processo Unificado. 3. ed. Porto Alegre: Bookman, 2022.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2020.

SEIDL, Juliana; ZANELLA, Luís Fernando. Metodologia de pesquisa em engenharia de software. 1. ed. Rio de Janeiro: Elsevier, 2020.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2022.