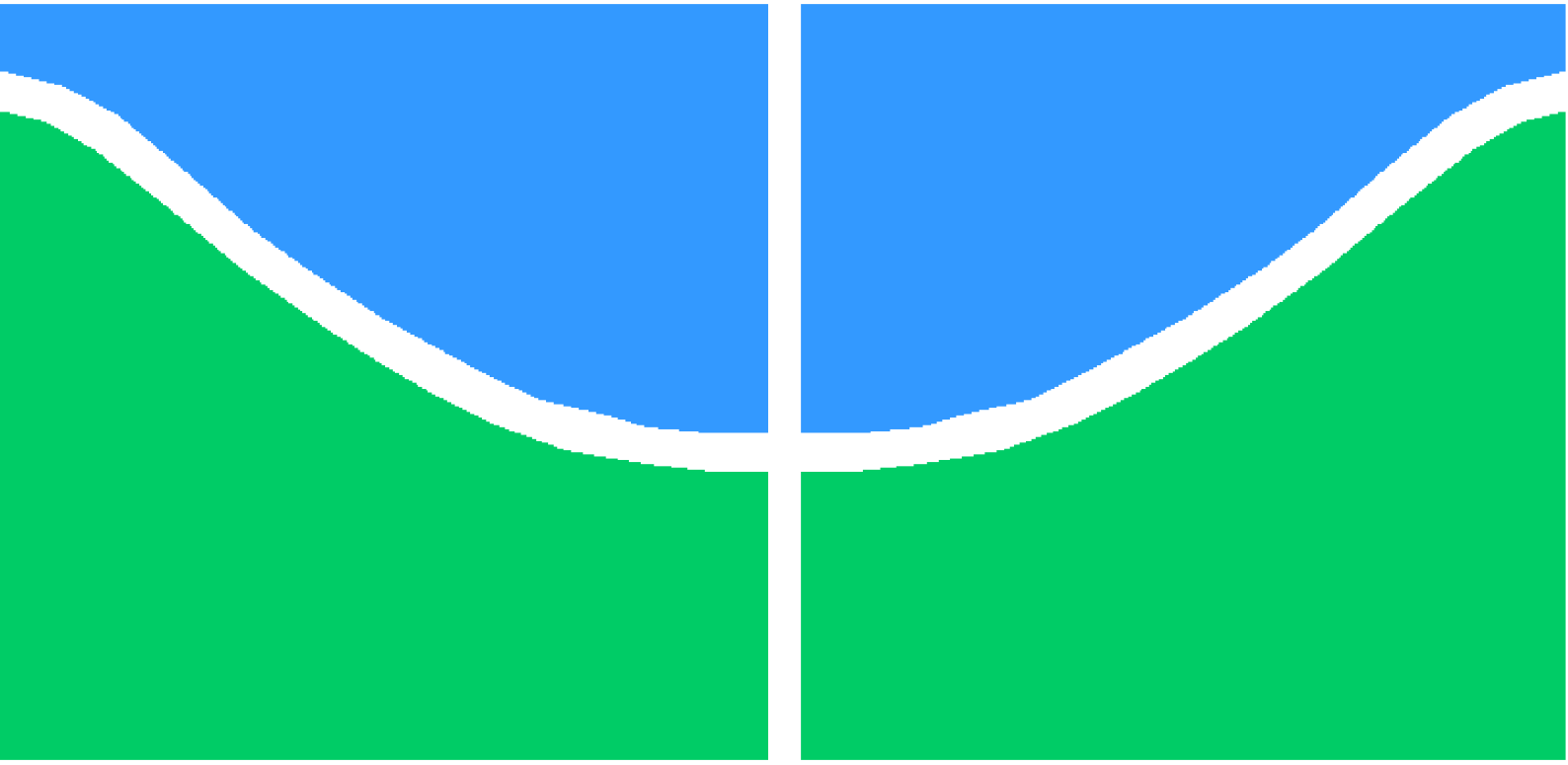


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Aplicativo para controle de viagens, gastos e receitas de caminhoneiros

Autor: Igor Batista Paiva e Thiago Aparecido Lopes Santos

Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF

2023

Igor Batista Paiva e Thiago Aparecido Lopes Santos

Aplicativo para controle de viagens, gastos e receitas de caminhoneiros

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF

2023

Igor Batista Paiva e Thiago Aparecido Lopes Santos
Aplicativo para controle de viagens, gastos e receitas de caminhoneiros/ Igor Batista Paiva e Thiago Aparecido Lopes Santos. – Brasília, DF, 2023-
112 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. Caminhoneiro. 2. Controle Financeiro. I. Prof. Dr. Vandor Roberto Vilardi Rissoli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicativo para controle de viagens, gastos e receitas de caminhoneiros

CDU 02:141:005.6

Igor Batista Paiva e Thiago Aparecido Lopes Santos

Aplicativo para controle de viagens, gastos e receitas de caminhoneiros

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 19 de julho de 2023 – Data da aprovação do trabalho:

Prof. Dr. Vandor Roberto Vilardi Rissoli
Orientador

Prof. Dr. André Barros de Sales
Convidado 1

Prof. Dr. Ricardo Matos Chaim
Convidado 2

Prof. MSc. Jair Alves Barbosa
Convidado da Universidade Católica de Brasília

Brasília, DF
2023

Agradecimentos

Eu, Igor, agradeço a minha noiva, por ser uma pessoa incrível que sempre me apoia, me dá força e ânimo para seguir em frente. Agradeço aos meus familiares e amigos, que sempre me apoiam e proporcionam momentos sublimes.

Eu, Thiago, agradeço à minha família por todo o suporte que me proporcionou em minha vida acadêmica e fora dela. À minha namorada, que esteve comigo durante todo o período da graduação, sempre me motivando a continuar com meus objetivos. Agradeço, também, aos amigos que fiz enquanto na universidade, por sempre estarem ao meu lado nos momentos mais difíceis.

Agradecemos a todos os professores que tivemos durante a graduação e ao nosso orientador, Vandor Roberto Vilardi Rissoli, por sua orientação preciosa e por compartilhar seu conhecimento e experiência conosco para a realização deste trabalho.

Resumo

Este trabalho teve como objetivo principal desenvolver uma aplicação para dispositivos móveis, utilizando uma abordagem híbrida de desenvolvimento, que permita aos caminhoneiros registrar suas viagens, gastos e receitas de forma simples e ágil, contribuindo para um controle sistematizado de seu trabalho e finanças. O processo de desenvolvimento de software adotou as metologias *Extreme Programming* (XP) e *DevOps*, bem como se baseia nos padrões arquiteturais *Representational State Transfer* (REST), *Model-View-Controller* (MVC) e Active Record.

Palavras-chave: Caminhoneiro. Controle Financeiro. Aplicativos Móveis. Abordagem Híbrida.

Abstract

This paper aimed primarily at developing a mobile application using a hybrid development approach, enabling truck drivers to record their trips, expenses, and revenues in a simple and agile manner, contributing to a systematic control of their work and finances. The software development process adopted the methodologies of Extreme Programming (XP) and DevOps, as well as being based on the architectural patterns Representational State Transfer (REST), Model-View-Controller (MVC), and Active Record.

Key-words: Mobile App. Truck Driver. Financial Management. Hybrid Approach.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ACID	*Atomicity, Consistency, Isolation, Durability*(acrônimo)
ANTT	Agência Nacional de Transportes Terrestres
API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CBO	Classificação Brasileira de Ocupações
CLT	Consolidação das Leis do Trabalho
CNT	Confederação Nacional do Transporte
CSS	*Cascading Style Sheets*
DE-R	Diagrama Entidade-Relacionamento
DLD	Diagrama Lógico de Dados
DT-e	Documento Eletrônico de Transporte
ETC	Empresa de Transporte Rodoviário de Cargas
GPS	*Global Positioning System*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IBGE	Instituto Brasileiro de Geografia e Estatística
IHC	Interação Humano-computador
INSS	Instituto Nacional do Seguro Social
IPCA	Índice Nacional de Preços ao Consumidor Amplo
IPEA	Instituto de Pesquisa Econômica Aplicada
JSON	*JavaScript Object Notation*
MVC	*Model View Controller*
PWA	*Progressive Web App*

REST	*Representational State Transfer*
RF	Requisito Funcional
RNF	Requisito Não Funcional
RNTRC	Registro Nacional de Transportadores Rodoviários de Cargas
RWD	*Responsive Web Design*
SDK	*Software Development Kit*
SGBDR	Sistema de Gerenciamento de Bancos de Dados Relacionais
SGBD	Sistema Gerenciador de Banco de Dados
SO	Sistema Operacional
TAC	Transportador Autônomo de Cargas
TCC	Trabalho de Conclusão de Curso
TRC	Transporte Rodoviário de Cargas
UI	*User Interface*
UX	*User Experience*
XP	*Extreme Programming*
VSCode	Visual Studio Code
IDE	*Integrated Development Environment*
PaaS	*Platform as a Service*

Sumário

1 Introdução

O objetivo deste capítulo é apresentar o cenário no qual este trabalho está inserido, sendo apresentada a contextualização, questão de pesquisa, justificativas, objetivos, metodologias e organização do trabalho.

1.1 Contextualização

A greve dos caminhoneiros ocorrida em 2018 deixou claro como o trabalho exercido pelos caminhoneiros no Brasil é de extrema importância para todos os setores do país. Após o início da greve, uma crise de abastecimento começou, ocasionando escassez e aumento dos preços de diversos produtos (LOURENçO, 2018).

Lourenço (2018) mostrou o impacto econômico da greve a partir da análise de alguns indicadores, como o Índice Nacional de Preços ao Consumidor Amplo (IPCA), volume de produção industrial e volume de vendas do comércio varejista. Esses indicadores sofreram uma queda durante o período da greve e começaram a voltar ao nível anterior após o fim da paralisação.

Conforme a nota técnica da logística dos transportes no Brasil de 2014 realizada pelo IBGE (2014), as principais estruturas de transporte do país são rodovias, ferrovias, hidrovias, aeroportos e portos. Cerca de 61,1% da carga transportada em 2009 foi por meio de rodovias. Este dado traz uma informação importante acerca da predominância do transporte rodoviário em relação aos demais, o que ressalta ainda mais a importância deste segmento para o Brasil.

Bueno (2010) afirma que a educação financeira permite melhor aproveitamento do fluxo de caixa familiar, acesso a produtos financeiros, traz um padrão de vida desejável e proporciona sua manutenção. De acordo com a Exame (2015), o Brasil está em 74º lugar no ranking global de Educação Financeira, atrás de alguns dos países mais pobres do mundo, como Madagascar, Togo e Zimbábue. Isso comprova a falta de atenção à educação financeira no país, o que corrobora com os índices apontados por Oliveira (2022, p. 3–4):

> "48% dos brasileiros não controlam o próprio orçamento pessoal e, dentre os que controlam suas finanças (52%), a frequência com que esses anotam e analisam suas despesas não é adequada.".

A Pesquisa Nacional por Amostra de Domicílios Contínua, realizada pelo IBGE (2022), traz informações sobre a posse de telefone móvel celular e utilização da Internet

para uso pessoal. A parcela dos domicílios com posse de aparelho celular em 2021 era de 96,3%, e a Internet era utilizada em 90% dos domicílios. Na população com 10 anos ou mais de idade, 84,4% possuíam telefone móvel celular para uso pessoal, e a fração que possuía acesso à Internet por meio desse aparelho era de 94,8%.

Tendo em vista o problema de educação e saúde financeira em que o Brasil se encontra, e considerando a importância dos caminhoneiros no transporte de cargas do país, além dos gastos substanciais advindos do uso e manutenção de caminhões, um software para controle de gastos e análise de indicadores seria de grande valia para esses importantes profissionais. Isso permitiria um controle mais acurado e preciso de suas finanças em viagens a serem realizadas ou em curso.

Considerando a agenda, geralmente, exacerbada e difícil dos caminhoneiros, além do volume elevado de brasileiros em posse de aparelhos telefônicos com conexão a Internet, a elaboração de um software para dispositivos móveis, que pudesse auxiliar no controle e gestão dos recursos desses profissionais, contribuiria com as suas atividades cotidianas de trabalho, permitindo maior foco nas demais tarefas a serem realizadas pelos caminhoneiros.

1.2 Questão de Pesquisa

Para o desenvolvimento do trabalho, é necessário entender como auxiliar os caminhoneiros do Brasil. Dessa forma, a questão de pesquisa a ser argumentada neste trabalho seria:

> Como auxiliar os caminhoneiros do Brasil na gestão de viagens, gastos e receitas por meio de um software utilizado em um dispositivo móvel?

1.3 Justificativa

Uma pesquisa realizada pela Confederação Nacional do Transporte (CNT) sobre o perfil dos caminhoneiros no Brasil, mostra que 87,7% deles utilizam a Internet (CNT, 2019). Ademais, entre 2016 e 2021 o percentual de domicílios que possuem um microcomputador exibiu um padrão de declínio. Em 2021, o valor foi de 40,7% das residências (IBGE, 2022), indicando uma dificuldade para o uso de softwares em computadores para mais da metade da população.

O perfil dos caminhoneiros brasileiros levantado pela CNT (2019) demonstra que a média de idade é de 44,8 anos. Portanto, é importante ao projetar um sistema para este público, considerar questões de usabilidade para usuários de meia-idade. Lara (2012) ressalta a importância da acessibilidade para pessoas mais velhas, podendo dar mais

autonomia no acesso à informática, e, consequentemente, melhorar a qualidade de vida na velhice. A autora reforça ainda que há uma defasagem entre o conhecimento adquirido pelos jovens e a meia/maior idade, visto que quase não há familiaridade do público mais velho com ícones, teclas, botões e mouse, enquanto os jovens cresceram expostos a tais estímulos. Dessa maneira, é imprescindível que haja uma atenção especial à acessibilidade do software proposto para satisfazer as necessidades dos perfis de usuários mais comuns.

O desenvolvimento de uma aplicação que responda à questão de pesquisa levantada é importante por permitir que essa classe de trabalho, extremamente relevante para a sociedade nacional, seja capaz de controlar suas viagens, gastos e receitas relacionadas com as viagens de maneira mais simples, utilizando recursos comuns em seu dia a dia. Por não existir uma forte cultura de educação financeira no Brasil atualmente, uma solução que apresente maneiras amigáveis de gerir finanças poderia contribuir diretamente para com a saúde financeira dos caminhoneiros usuários desse software.

Estando presente em um dispositivo móvel e possuindo foco em usabilidade, esta aplicação conseguiria fornecer aos caminhoneiros uma forma prática de melhorar sua saúde financeira e facilitar o desempenho de sua profissão.

O tema deste trabalho se mostrou mais relevante entre os demais temas elucidados por propor uma forma de auxiliar uma classe de trabalho essencial para o país, fornecendo uma forma simples e sistemática de controle de gastos, receitas e viagens. Este controle é fundamental para atingir uma vida saudável financeiramente e torna mais fácil a realização dos objetivos desses profissionais.

1.4 Objetivos

1.4.1 Objetivo geral

O objetivo principal deste trabalho é desenvolver uma aplicação para dispositivos móveis, fornecendo aos caminhoneiros uma forma simples e ágil de registro de viagens, gastos e receitas, contribuindo com um controle sistematizado de seu trabalho e suas finanças.

1.4.2 Objetivos específicos

Para atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Prover aos caminhoneiros uma forma de controle de viagens, gastos e receitas;
- Fornecer aos caminhoneiros indicadores acerca de suas viagens, gastos e receitas;
- Garantir que o software funcione adequadamente na rotina de um caminhoneiro;

- Promover o acesso no nível adequado ao perfil de usuário do caminhoneiro, considerando boas práticas de Interação Humano-Computador, para proporcionar uma experiência de uso agradável e eficiente;
- Prover aos caminhoneiros uma maneira de gerir suas viagens, gastos e receitas, por meio de um software que funcione em dispositivos móveis, garantindo a consistência e a integridade dos dados;
- Aplicar boas práticas da engenharia de software no desenvolvimento da proposta e do software, visando garantir maior qualidade, a segurança e a confiabilidade da aplicação.

1.5 Metodologia

As metodologias utilizadas na elaboração deste trabalho estão dividas em duas categorias: Metodologia de Pesquisa e Metodologia de Desenvolvimento de Software.

1.5.1 Metodologia de pesquisa

Gerhardt e Silveira (2009) categorizam os diferentes tipos de pesquisa em relação à sua abordagem, sua natureza, seus objetivos e seus procedimentos, enquanto Serzedello e Tomaél (2011) trazem o conceito de produção tecnológica como um procedimento de pesquisa. Dessa forma, a metodologia de pesquisa deste trabalho corresponde às classificações indicadas na Tabela 1, detalhadas na Seção 3.1.

Tabela 1 – Metodologia de Pesquisa.

Abordagem	Natureza	Objetivos	Procedimentos
Qualitativa	Aplicada	Exploratório	Pesquisa bibliográfica Produção tecnológica

Fonte: Autoria própria.

Na Tabela 1 é possível observar que a metodologia de pesquisa adota é de abordagem qualitativa, natureza aplicada, com objetivos exploratórios e adota os procedimentos de pesquisa bibliográfica e produção tecnológica.

1.5.2 Metodologia de desenvolvimento de software

A metodologia de desenvolvimento utilizada neste trabalho é uma adaptação da metodologia de desenvolvimento de software ágil *Extreme Programming*, em conjunto

com algumas práticas de DevOps. Estas metodologias são abordadas com mais detalhes na Seção 3.2.

Em relação ao *Extreme Programming*, utilizando os termos usados por Beck e Andres (2004), serão utilizados os seguintes valores, princípios e práticas:

- Valores
  - Comunicação
  - Simplicidade
- Princípios
  - Qualidade
  - *Baby steps* (pequenos passos)
- Práticas
  - Ciclo semanal
  - Integração contínua
  - Código compartilhado
  - Código e testes

E no que diz respeito ao *DevOps*, seguindo os termos definidos por Kim et al. (2021), o foco se concentra nas seguintes práticas:

- Integração contínua
- Infraestrutura como código
- Testes automatizados rápidos e confiáveis

1.6 Organização do trabalho

- Capítulo 1 – **Introdução**: capítulo inicial que aborda em síntese contextualização do projeto proposto, sua questão de pesquisa, justificativa, objetivos e metodologias;
- Capítulo 2 – **Referencial Teórico**: neste capítulo é apresentada uma fundamentação essencial à compreensão da contextualização resultante do levantamento que se relaciona com o tema discutido e os recursos tecnológicos que poderiam contribuir com a metodologia de trabalho;

- Capítulo 3 – **Metodologia**: capítulo que esclarece as metodologias utilizadas na elaboração deste trabalho, detalhando as atividades desenvolvidas, tecnologias utilizadas, requisitos definidos e arquitetura do software e seus recursos;
- Capítulo 4 – **Desenvolvimento**: capítulo que detalha os aspectos mais relevantes no processo de desenvolvimento do software utilizando as metodologias adotadas;
- Capítulo 5 – **Conclusão**: capítulo que encerra o Trabalho de Conclusão de Curso, apresentando os resultados obtidos e objetivos alcançados durante a execução do trabalho, além de alguns possíveis trabalhos futuros.

2 Referencial Teórico

Neste capítulo serão detalhadas as fundamentações necessárias para um melhor entendimento do contexto do trabalho, de forma a explicar conceitos relevantes para a elaboração do mesmo, sendo eles: aplicativos móveis, possíveis tecnologias a serem exploradas, características dos caminhoneiros no Brasil e aspectos da interação humano-computador.

2.1 Aplicativos móveis

O objetivo desta seção é discorrer sobre o conceito de aplicativos móveis, a fim de fundamentar a realização do trabalho.

2.1.1 Contextualização

Uma aplicação móvel pode ser definida como uma aplicação de software que opera em um dispositivo móvel, sendo ele um *smartphone*, *tablet*, etc., e que possua um sistema operacional (SO) que suporte softwares autônomos (WANG; LIAO; YANG, 2013).

De acordo com os dados apresentados pelo StatCounter (2022) [1], existem dois SOs dominantes no mercado de dispositivos móveis, sendo eles o Android, que faz parte do Google e concentra 72,37% dos usuários, e o iOS, desenvolvido e gerido pela Apple, concentrando outros 26,98%. Assim, mais de 99% de todos os consumidores estão divididos entre as duas gigantes tecnológicas.

Aplicativos móveis são tipicamente desenvolvidos por terceiros, podendo eles serem fábrica de softwares ou indivíduos desenvolvedores. As lojas funcionam como mercados on-line bilaterais, que geram vantagens mútuas para todos os envolvidos (ROMA; RAGAGLIA, 2016). Ambas as plataformas iOS e Android adotam essa estratégia como meio principal de obtenção de aplicativos desde à consolidação dos *smartphones*. Junto ao lançamento do primeiro iPhone, a Apple trouxe a App Store, e no mesmo ano, o Google respondeu com o Android Market, que posteriormente viria a se chamar Google Play.

Ter uma plataforma específica para fazer o *download* de aplicativos pode ser benéfico em vários aspectos. Uma vantagem é que existe um processo claro e uma expectativa definida no que se diz respeito à capacidade de se localizar e expor os aplicativos (BUETTNER; SIMMONS, 2011).

As lojas de aplicativos têm mecanismos de buscas e listas dos melhores aplicativos, que ajuda os usuários a encontrarem o que precisam. Também possuem um modelo de

[1] Dados do último bimestre de 2022.

monetização desenvolvido e o mecanismo de pagamento já está implementado, na maior parte dos casos, o usuário realiza a compra através da própria loja de aplicativos, então os desenvolvedores do aplicativo só se preocupam na precificação dos produtos de software (BUETTNER; SIMMONS, 2011, tradução nossa).

Desde a criação da App Store e da Android Store em 2008, houve um crescimento exorbitante no mercado de aplicativos móveis. Em 2010, havia uma previsão de aumento de receita global de US$1.94 bilhões em 2009 para US$15.65 bilhões em 2013 (LIU; AU; CHOI, 2012). Em 2021, de acordo com o portal MacMagazine (2021), contabilizando somente o terceiro bimestre do ano, o gasto com aplicativos móveis foi de US$35,7 bilhões.

Dehlinger e Dixon (2011) afirmam que os dispositivos móveis são a plataforma de computação que mais cresce no mundo. Associado a isso, Andrade, Agra e Malheiros (2013) alegam que o uso de dispositivos móveis no Brasil tem crescido em taxas mais altas do que as médias mundiais nos últimos anos.

Tais fatos demonstram que os dispositivos móveis, juntamente com os aplicativos, vêm se tornando cada vez mais populares e essenciais à vida humana.

2.1.2 Desenvolvimento

Em relação ao desenvolvimento de aplicativos móveis, não existe uma abordagem perfeita. Apesar de existirem milhões de aplicativos no mercado, ainda é muito desafiador desenhar e desenvolver um aplicativo móvel que pode ser executado entre plataformas diferentes utilizando o mesmo código. Há poucos anos, utilizar o *kit* de desenvolvimento nativo de cada plataforma era a única maneira de se desenvolver aplicativos, sendo diferente esta realidade atualmente (HUYNH; GHIMIRE; TRUONG, 2017).

2.1.2.1 *Websites* Móveis

Quando o número de usuários de *tablets* e *smartphones* começou a crescer, *web designers* geralmente adotavam uma estratégia de *websites* móveis. O objetivo era criar um *site* que permitisse que os usuários conseguissem acessar o conteúdo de páginas *web*. O ponto chave dessa estratégia era a capacidade de trazer uma boa experiência tanto para usuários de dispositivos móveis quanto de computadores de mesa (*desktop*), porque cada página *web* era projetada especificamente para seu respectivo dispositivo (HUYNH; GHIMIRE; TRUONG, 2017).

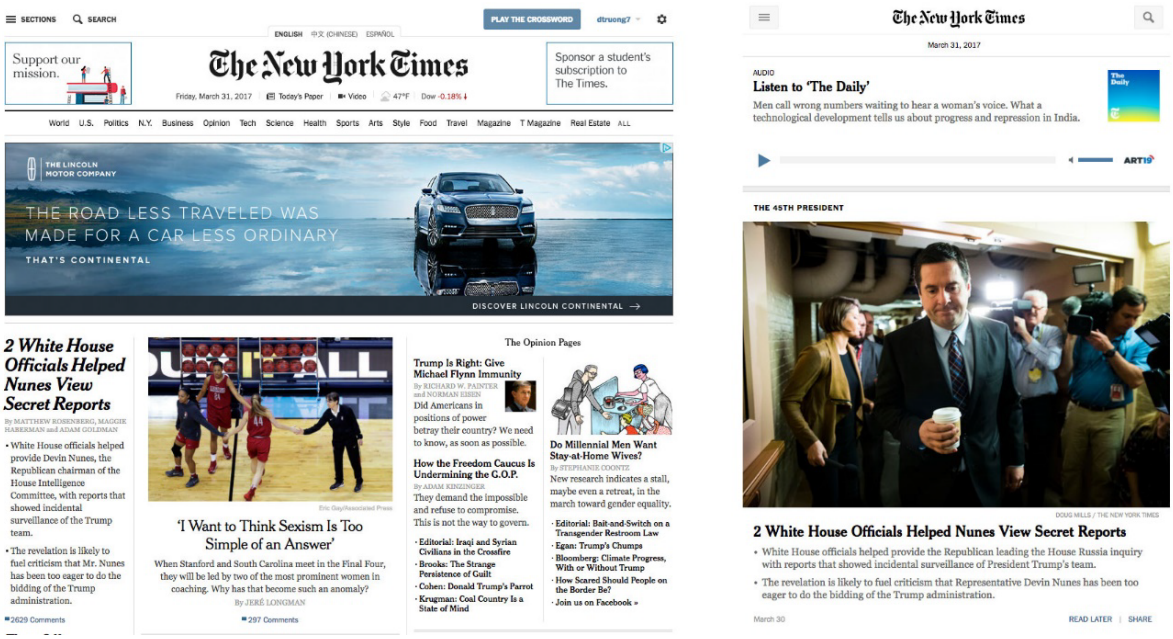
Um *website* móvel é um outro termo para aplicação *web* móvel genérica. Existem diversas maneiras de se desenvolver versões móveis de *websites*. Todavia, a premissa habitual é que a versão *desktop* do *website* identifica dispositivos móveis através do identificador de agente de usuário (*user-agent*) do navegador. Uma vez que o dispositivo móvel é detectado, o *user-agent* é redirecionado a um *website* móvel criado para aquele dispositivo

específico, ou a um *website* que utiliza técnicas de *web design* responsivo para prover o mesmo conteúdo para tipos diferentes de dispositivos (JOBE, 2013).

No entanto, essa abordagem exige o projeto, desenvolvimento e manutenção de dois ou mais *websites*, logo uma consequência é o aumento no custo, além do esforço necessário tanto para desenvolver quanto para manter vários *sites*. Outro fator importante é que a versão móvel do *site* dependia da capacidade do servidor *web* de detectar um navegador *mobile* através de um código JavaScript integrado.

Este código de detecção deveria ser atualizado regularmente para ser possível detectar novos dispositivos, caso contrário, esses novos dispositivos no mercado poderiam não ser detectados, e então não seriam redirecionados para suas respectivas versões móveis. Devido a essas desvantagens, esse tipo de estratégia evoluiu gradualmente ao *Responsive Web Design* (RWD) que perdura até os dias de hoje (HUYNH; GHIMIRE; TRUONG, 2017).

Figura 1 – Contraste entre uma página *web* para *desktop* à esquerda e uma página *web* móvel à direita.



Fonte: Retirado de Huynh, Ghimire e Truong (2017).

A Figura 1 demonstra duas versões de um mesmo *website*, sendo um projetado para *desktop* e o outro para dispositivos móveis.

2.1.2.2 RWD - *Responsive Web Design*

Na abordagem RWD o objetivo é otimizar os acessos e telas dos conteúdos existentes no *website* para múltiplos tipos de dispositivos, incluindo *smartphones*, *tablets*, *notebooks* e *desktops*. A chave é fazer com que o *site* seja responsivo a diferentes tama-

nhos de telas, sem que seja necessária a criação de múltiplas versões de um mesmo *website* (HUYNH; GHIMIRE; TRUONG, 2017).

O RWD é o conceito de utilizar CSS (*Cascading Style Sheets*), que é uma linguagem de folhas de estilo para descrever a apresentação de páginas *web* e realizar consultas de mídia para determinar a resolução do dispositivo que está sendo utilizado, além de ajustar como o conteúdo do *website* será apresentado de acordo (JOBE, 2013).

O maior atrativo do RWD está na simplicidade do código, assim como na sua manutenção. No RWD só é necessário criar e manter uma versão do *website*, reduzindo o gasto de tempo e recursos. No entanto, o RWD herda todas as limitações encontradas em um *website* comum. O seu funcionamento depende de um servidor *web* e de um navegador no lado do cliente. Além disso, adicionar novas funcionalidades para um *website* RWD por vezes pode não ser viável, devido à limitação dos recursos do *framework* de desenvolvimento (HUYNH; GHIMIRE; TRUONG, 2017).

Figura 2 – Ilustração que exemplifica o uso de RWD.

Fonte: Disponível em: <https://www.temperim.com>. Acesso em: 14 jan. 2023.

A Figura 2 exemplifica a mesma página *web* sendo mostrada corretamente de maneiras diferentes de acordo com o dispositivo utilizado.

2.1.2.3 Aplicativos Móveis Nativos

Com o crescimento desenfreado do mercado de dispositivos móveis, e muitos usuários até substituindo computadores de mesa e até *notebooks*, houve um grande aumento na demanda de acesso de conteúdo na *web*, ou até trabalhar e ter a possibilidade de utilizar recursos *offline*. Dessa forma, aplicativos projetados especificamente para dispositivos

móveis entram em cena, sendo o mais dominante deles, os aplicativos nativos (HUYNH; GHIMIRE; TRUONG, 2017).

Aplicativos móveis nativos se referem a aplicações especificamente escritas e desenvolvidas para SOs específicos. Características chaves de aplicações nativas são o seu acesso irrestrito ao *hardware* dos dispositivos e o suporte a todos os recursos relacionados à interface do usuário e suas interações disponíveis em cada SO dos dispositivos (JOBE, 2013).

Aplicações nativas têm muito a oferecer aos desenvolvedores e aos usuários. Em termos de performance, aplicativos nativos são os mais rápidos e eficientes. Aplicações nativas também fornecem ao usuário a possibilidade de acessar conteúdo sem estar necessariamente conectado a Internet. Além disso, um aplicativo nativo permite tirar vantagem de atributos do dispositivo como a câmera, álbum de fotos, GPS, etc. que podem oferecer mais funcionalidades ao usuário. Por esses motivos, a escolha por aplicações nativas podem ser muito atrativas (BUETTNER; SIMMONS, 2011).

Apesar dessa abordagem de desenvolvimento ser dominante atualmente, ela não é perfeita por alguns motivos como: cada SDK (*Software Development Kit*) de um aplicativo móvel está atrelado a uma plataforma específica, por exemplo, um SDK para o iOS é totalmente diferente do que se encontra para o Android. Outra razão seria que o iOS e o Android são todos baseados em plataformas de *hardware* proprietários. Qualquer mudança no SO ou no *hardware* pode fazer com que os aplicativos acabem se tornando incompatíveis (HUYNH; GHIMIRE; TRUONG, 2017).

2.1.2.4 Aplicativos móveis híbridos

Uma alternativa aos aplicativos nativos são os aplicativos móveis híbridos. Huynh, Ghimire e Truong (2017) afirmam que essa abordagem permite aos desenvolvedores criarem apenas um aplicativo móvel se utilizando de padrões *web* e distribuí-lo, consistentemente, para múltiplas plataformas com pouca (ou nenhuma) mudança.

Uma aplicação híbrida é basicamente um *website* com HTML, CSS e JavaScript. A diferença é que ele funciona em uma camada isolada do navegador e tem acesso à camada nativa do SO. Para funcionar como um aplicativo nativo ele depende de uma interface, como o Cordova (HUYNH; GHIMIRE; TRUONG, 2017). Esta interface é um *wrapper* nativo, que permite que interfaces incompatíveis trabalhem em conjunto (GAMMA et al., 2007).

Portanto, uma aplicação híbrida oferece um mecanismo para transformar aplicativos *web* baseados em navegadores em aplicativos móveis que podem ser compilados em códigos binários executáveis, podem ser baixados de lojas de aplicativos e funcionam tanto *on-line* quanto *offline* (HUYNH; GHIMIRE; TRUONG, 2017).

A abordagem híbrida não é praticada sem a existência de *frameworks*. Aqueles que integram o uso de HTML, CSS e JavaScript são de grande valia, pois permitem que os desenvolvedores criem aplicativos com visual nativo, mas não necessitam do uso do Java (Android) ou Objective C (iOS). Além disso, tais *frameworks* também permitem que os desenvolvedores maximizem o impacto do seu esforço aplicando o conceito de "escreva uma vez, execute em qualquer lugar" (HUYNH; GHIMIRE; TRUONG, 2017, p. 54, tradução nossa).

2.1.2.4.1 React Native

O React Native é um *framework* JavaScript criado para escrever aplicações móveis renderizadas nativamente para iOS e Android. Ela é baseada no React, uma biblioteca criada pelo Facebook para a criação de interfaces de usuário, mas que foca em aplicações móveis ao invés do navegador. Dessa forma, possibilita desenvolvedores *web* a elaborarem aplicativos que se parecem nativos. Além disso, devido à possibilidade da maior parte do código escrito ser compartilhado entre plataformas, o React Native facilita o desenvolvimento simultâneo para Android e iOS (EISENMAN, 2015).

O fato do React Native renderizar utilizando a API de renderização nativa de cada plataforma faz com que ele tenha destaque em relação à maioria dos *frameworks* híbridos do mercado, como o Cordova ou Ionic. Tais *frameworks* tipicamente realizam a renderização utilizando *webview*, que acarretam um pior desempenho, além de tentar imitar os elementos de *User Interface* (UI), tarefa que pode levar muito tempo e no final pode trazer a sensação de que algo está estranho para o usuário. (EISENMAN, 2015).

2.1.2.4.2 Ionic

O Ionic é um *framework* de código aberto para criação de aplicativos móveis utilizando tecnologias *web* (HTML, CSS e JavaScript). Ele foca no que se diz respeito às interações de UI e *User Experience* (UX) do *front-end* de um aplicativo. Além disso, o Ionic permite que os desenvolvedores criem aplicativos que funcionam em todas as maiores lojas de aplicativos com apenas um código fonte base. O Ionic emula as orientações de UI nativas e utiliza SDK nativos, trazendo padrões de UI e funcionalidades de aplicações nativas. (IONIC, 2011)

Aplicações feitas utilizando o Ionic podem ser implantadas nativamente por meio dos *frameworks* Capacitor ou Cordova, ou pode ser executado em um navegador como um PWA (*Progressive Web App*) (IONIC, 2011).

2.2 Caminhoneiros no Brasil

O objetivo desta seção é apresentar características relevantes acerca dos caminhoneiros no Brasil, servindo de embasamento para a realização deste trabalho.

2.2.1 Contextualização

A Classificação Brasileira de Ocupações (CBO, 2022, n.p) define caminhoneiros e motoristas de veículos de cargas da seguinte forma:

> "Transportam, coletam e entregam cargas em geral; guincham, destombam e removem veículos avariados e prestam socorro mecânico. Movimentam cargas volumosas e pesadas, podem, também, operar equipamentos, realizar inspeções e reparos em veículos, vistoriar cargas, além de verificar documentação de veículos e de cargas. Definem rotas e asseguram a regularidade do transporte. As atividades são desenvolvidas em conformidade com normas e procedimentos técnicos e de segurança."

Portanto, para realização deste trabalho, é necessária a locomoção do veículo, guiado pelo caminhoneiro, até um destino, podendo este trajeto levar dias para ser concluído. O motorista deve parar para abastecer o veículo, fazer suas necessidades fisiológicas, se alimentar, se higienizar e também descansar e dormir. Devido a essas necessidades da profissão, os caminhoneiros estão sujeitos a todos os tipos de problemas que podem ocorrer em uma estrada (SILVA, 2015).

Silva (2015) aborda a jornada de trabalho de caminhoneiros e a categoriza como exaustiva. Também ressalta a heterogeneidade desta categoria de trabalho, dado os diferentes tipos de trabalho, veículos, cargas, rotas e características individuais. O autor acentua a diferença no que concerne a organização da jornada de trabalho. Alguns caminhoneiros fazem poucas paradas com jornadas de 8, 12, 16 ou até mais horas consecutivas, enquanto outros costumam parar em horários determinados por si mesmo ou quando sentem a necessidade de realizar alguma atividade. Uma diferença que o autor traz entre esses dois casos é que, na maioria, os que param com maior frequência possuem caminhão próprio, e, por outro lado, alguns caminhoneiros têm locais e horários pré-estabelecidos pelas empresas.

O Instituto de Pesquisa Econômica Aplicada (IPEA) afirma que a maior carga horária média de trabalho no ano de 2007 foi no setor de transporte, totalizando 46,2 horas semanais (IPEA, 2009). Por conta dessa jornada de trabalho exaustiva e as demais condições de trabalho, os caminhoneiros precisam adotar hábitos que impactam diretamente em sua saúde. Alessi e Alves (2015) trazem alguns hábitos de vida adotados pelos caminhoneiros que podem prejudicar a própria saúde:

- Alimentação de alto valor calórico e baixo valor nutritivo;

- Horários de descanso insuficientes;
- Sedentarismo;
- Tabagismo;
- Uso de álcool;
- Uso de substâncias psicoativas;
- Ausência de controle periódico em saúde;
- Exposição a doenças sexualmente transmissíveis;

2.2.2 Relações de trabalho

Silva (2015) identifica três tipos de relações de trabalho na categoria de motoristas de caminhões: empregado registrado segundo as normas da Consolidação das Leis do Trabalho (CLT), motorista autônomo com caminhão próprio e motorista sem caminhão próprio que trabalha para um autônomo.

Os caminhoneiros em regime CLT trabalham para fabricantes e comerciantes que possuem uma frota própria, ou para empresas de transporte. Esses profissionais possuem um vínculo empregatício com as empresas, portanto gozam dos benefícios e seguranças das leis trabalhistas e contribuem para a previdência social (INSS), além de possuírem menos obrigações com o veículo e dívidas (SILVA, 2015).

Na Lei n.º 11.442 de 2007, que dispõe sobre o Transporte Rodoviário de Cargas (TRC) por conta de terceiros e mediante remuneração, o artigo 2º define:

> I - Transportador Autônomo de Cargas - TAC, pessoa física que tenha no transporte rodoviário de cargas a sua atividade profissional;
> II - Empresa de Transporte Rodoviário de Cargas - ETC, pessoa jurídica constituída por qualquer forma prevista em lei que tenha no transporte rodoviário de cargas a sua atividade principal. [...] (BRASIL, 2007, p. 1)

A forma de prestação de serviço, definida em contrato, entre a ETC e o TAC categoriza o transportador como agregado ou independente, ademais é permitido a cessão do veículo do TAC para outro profissional, chamado de auxiliar (BRASIL, 2007). Os parágrafos 1º, 2º e 3º, do artigo 4º, da Lei n.º 11.442 de 2007, declaram:

> § 1º Denomina-se TAC-agregado aquele que coloca veículo de sua propriedade ou de sua posse, a ser dirigido por ele próprio ou por preposto seu, a serviço do contratante, com exclusividade, mediante remuneração certa.
> § 2º Denomina-se TAC-independente aquele que presta os serviços de transporte de carga de que trata esta Lei em caráter eventual e sem exclusividade, mediante frete ajustado a

> cada viagem.
> § 3º Sem prejuízo dos demais requisitos de controle estabelecidos em regulamento, é facultada ao TAC a cessão de seu veículo em regime de colaboração a outro profissional, assim denominado TAC - Auxiliar, não implicando tal cessão a caracterização de vínculo de emprego. (BRASIL, 2007, p. 1)

No parágrafo primeiro do artigo 2º, a lei mencionada também afirma que o TAC deve:

> I - comprovar ser proprietário, co-proprietário ou arrendatário de, pelo menos, 1 (um) veículo automotor de carga, registrado em seu nome no órgão de trânsito, como veículo de aluguel;
> II - comprovar ter experiência de, pelo menos, 3 (três) anos na atividade, ou ter sido aprovado em curso específico. (BRASIL, 2007, p. 1)

Os profissionais autônomos prestam serviços para empresas, ou outros TACs, mas não possuem vínculo empregatício, bem como não dispõe dos benefícios e seguranças da CLT e no geral assumem as responsabilidades para com o veículo de transporte (BRASIL, 2007; SILVA, 2015).

Silva (2015) ressalta que uma parcela pequena dos transportadores autônomos contribui com o INSS por conta própria, não possuindo respaldo da previdência em caso de problemas de saúde.

2.2.3 Questões financeiras

O caminhoneiro que trabalha regido pela CLT é um funcionário assalariado, mas segundo Silva (2015) seu salário também depende de comissão sob o valor do frete ou por quilômetros percorridos.

Os caminhoneiros autônomos adquirem sua renda a partir do valor do frete, quanto mais viagens realizadas, maior a sua renda, porém todas as despesas, sejam elas relacionadas à viagem ou ao caminhão, terão que ser deduzidas do valor recebido. Estas despesas incluem: gastos com higiene pessoal, combustível, pedágio, hospedagem, impostos, alimentação, financiamento e manutenção do veículo. No caso do TAC auxiliar, a renda provém do pagamento do TAC, que varia entre 10 e 15% do valor bruto do frete. (SILVA, 2015).

Uma pesquisa utilizando o método etnográfico realizada por Silva (2015) foi capaz de registrar diversos relatos de caminhoneiros sobre as condições de trabalho, rotina, remuneração e problemas com a profissão.

Silva (2015) descreve, conforme os relatos de motoristas, que os empregados CLT ou TACs auxiliares, com muito esforço, chegam a ganhar de três a quatro salários mínimos e uma parcela significativa é gasta durante a viagem. E que no caso dos autônomos, as

empresas repassam apenas um terço ou metade do valor do frete, e desse valor 70% são para cobrir as despesas da viagem.

Portanto, considerando as despesas, a situação em relação ao lucro dos proprietários de caminhão, comparados com os empregados CLT e TACs auxiliares é semelhante, dado os gastos no decorrer da viagem, alto custo de manutenção do caminhão e o valor defasado dos fretes (SILVA, 2015).

O autor ainda aponta que os caminhões quebram com frequência, pois transportam muito peso e funcionam muitas horas por dia com poucos intervalos, além do estado de conservação não ideal das estradas, que resulta no aumento dos custos financeiros. Em sua pesquisa, o autor também relata que muitos motoristas não realizam manutenções preventivas no caminhão, uma vez que isso incide em mais gastos e deixa o meio principal para obtenção de renda parado em uma oficina.

Analisando os custos de transporte no eixo São Paulo ao Rio de Janeiro, os autores Araújo, Bandeira e Campos (2014) perceberam que com os valores praticados pelos autônomos, seria impossível para o transportador arcar com todos os custos operacionais e de gerenciamento corretamente. Esta informação corrobora com os relatos descritos por Silva (2015) da discrepância do valor do frete.

2.2.4 Documento Eletrônico de Transporte

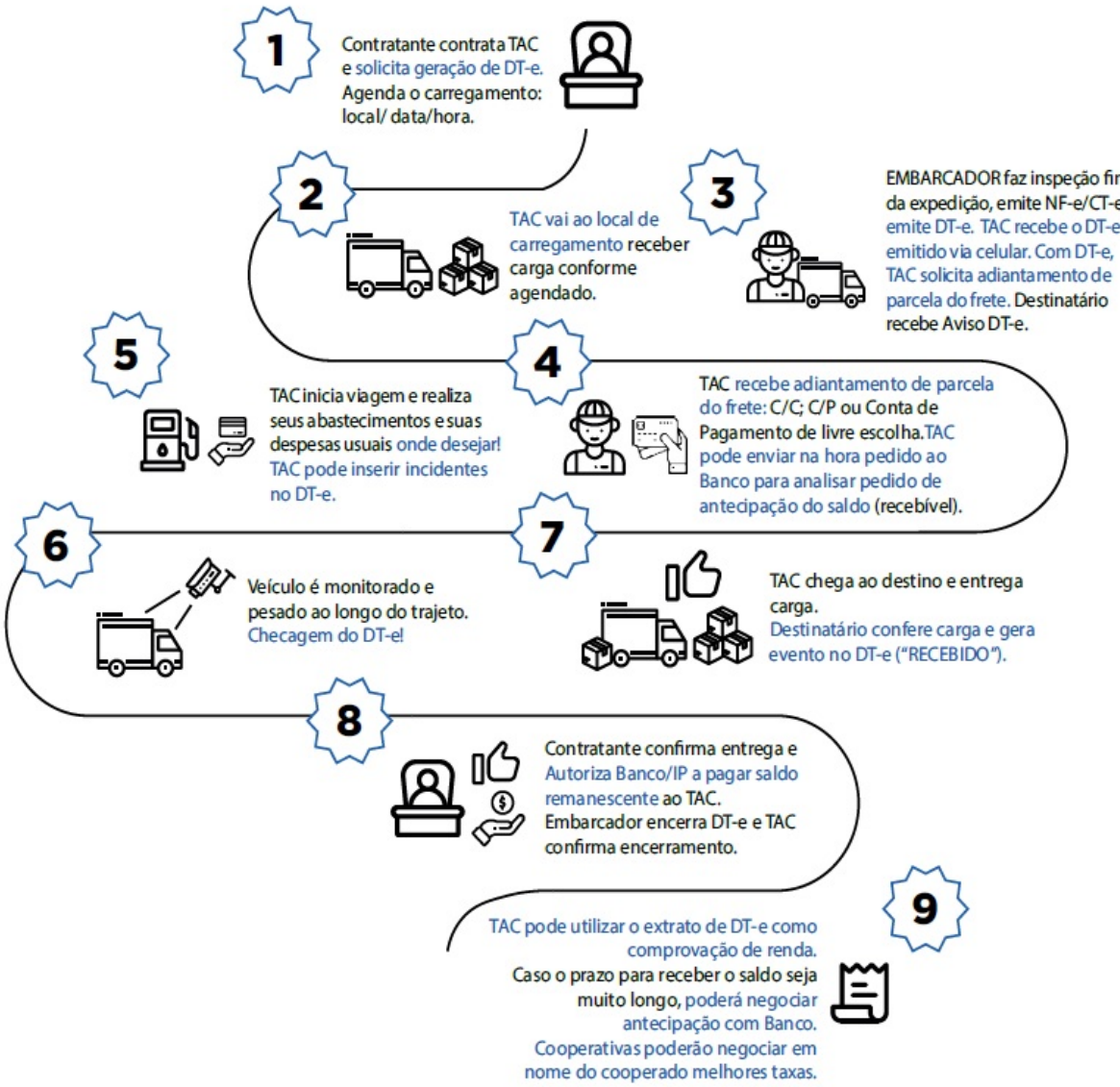
Instituído pela Lei n.º 14.206, de 27 de setembro de 2021, o Documento Eletrônico de Transporte (DT-e) pretende unificar e simplificar dados relacionados ao transporte de cargas, em todos os modos de transporte. O DT-e é uma plataforma tecnológica digital que visa reunir as informações que estão atualmente dispersas em mais de 90 documentos exigidos pelas 77 autoridades federais (GOV.BR, 2021; GOV.BR, 2022a).

A proposta do DT-e apresenta alguns benefícios para o mercado de transporte de cargas (GOV.BR, 2022c):

- **Redução de custos com obrigações administrativas:** por da conta desburocratização, redução de custo regulatório e simplificação relacionada aos documentos;
- **Aumento da eficiência logística:** mais tempo para realizar o transporte e menos com burocracia e fiscalização;
- **Documento apenas em formato digital:** acabando com necessidade de levar documentos impressos no caminhão;
- **Extrato dos DT-e emitidos:** o DT-e irá servir como comprovantes de rendimentos e podem ser acessados via aplicativo em dispositivo móvel.

- **Pagamento do frete via PIX:** os caminhoneiros poderão receber o valor do frete parcelado, ou integralmente, via PIX;
- **DT-e usado como fatura:** poderá ser usado como fatura para lastrear operações de crédito, renegociação de dívida, comprovação de renda;
- **Fiscalização do cumprimento da política do piso mínimo de frete:** o valor do frete contratado constará no DT-e, podendo ser fiscalizado pelas agências reguladoras;
- **Antecipação do vale-pedágio obrigatório:** o valor do pedágio fornecido ao caminhoneiro ficará destacado no DT-e;

Figura 3 – Resumo do fluxo operacional do DT-e.



Fonte: Retirado de gov.br (2022c).

As mudanças propostas pela plataforma, instituídas por lei, estão sendo aplicadas em ondas. Em cada onda, novos documentos são integrados ao DT-e, tipos diferentes de cargas são agregados e o uso do DT-e passa a ser obrigatório por tipo de carga. O projeto propõe 3 ondas, divididas de ano em ano, de 2021 até 2024, com a última onda sendo de 2024 em diante (ANTT, 2021). A Figura 3 mostra de forma resumida o fluxo operacional do DT-e.

A plataforma possui um aplicativo móvel que serve como ferramenta para o caminhoneiro controlar sua jornada, com as seguintes funcionalidades: calculadora de frete, consumo de diesel, jornada de trabalho, gerenciamento de rotas e agendamento de fretes (GOV.BR, 2022b). O aplicativo foi publicado no Google Play em janeiro de 2020 e possui mais de 10 mil *downloads* (GOOGLE PLAY, 2019).

2.2.5 Perfil do caminhoneiro

A CNT (2019) realizou entrevistas com mais de mil caminhoneiros no ano de 2018, sendo 714 autônomos e 352 empregados de frota, coletando algumas informações sobre o perfil dos caminhoneiros brasileiros. Segundo esta pesquisa, o mercado de trabalho é predominantemente masculino, com 99,5% dos profissionais homens, com a média de idade de 44,8 anos, ganhando cerca de R$ 4.600,00 por mês, trabalhando em média há 18,8 anos na profissão e que 87,7% deles utilizam a Internet.

Os caminhões utilizados têm em média 15,2 anos, 47% dos motoristas autônomos adquiriram o veículo através de financiamento e os caminhoneiros chegam a rodar mais de 9 mil quilômetros por mês e a trabalhar 11,5 horas por dia. Os principais pontos negativos da profissão foram: perigo, insegurança, o desgaste causado e convívio familiar comprometido. E as maiores dificuldades encontradas foram: assaltos, roubos e custo do combustível (CNT, 2019).

Acerca da escolaridade dos caminhoneiros no Brasil, Silva (2015) afirma que o setor de transportes é composto em sua maioria por pessoas de baixa escolaridade.

Conforme o relatório anual da ANTT (2015) existem 1.106.611 transportadores inscritos no Registro Nacional de Transportadores Rodoviários de Cargas (RNTRC), com uma frota total de 2.339.703 veículos. A Tabela 2 apresenta o número de registros para cada tipo de transportador.

Examinando a Tabela 2 é possível observar que a maioria dos transportadores são autônomos, constituindo mais de 82% dos registros, seguidos das empresas de transporte rodoviário de cargas (ETC) que representam mais de 16% das inscrições.

Tabela 2 – Transportadores Registrados no RNTRC.

Tipo de transportador	**Número de registros**	%
Profissionais autônomos	918.391	82,99%
Empresas	187.784	16,97%
Cooperativas	436	0,04%
Total	1.106.611	100%

Fonte: Tabela elaborada pelos autores com base em ANTT (2015).

2.3 Finanças

Esta seção tem como objetivo abordar aspectos relacionados às finanças, no que tange o planejamento e controle financeiro, no quesito pessoal e de pequenas empresas envolvidas com este tipo de atividade abordada neste trabalho.

2.3.1 Finanças Pessoais

De acordo com Santos (2014), em meio ao consumo excessivo, muitas pessoas se endividam, comprometem parte significativa de sua renda e acabam ficando inadimplentes. Para evitar este problema, o autor diz ser indispensável o registro e controle, de forma contínua, de todas as receitas obtidas e gastos realizados em um dado período, ou seja, formalizar um planejamento financeiro.

Um planejamento financeiro possibilita análises sobre como estão as finanças em um determinado momento e também projetar como elas podem estar no futuro. Dessa forma é possível remover gastos desnecessários, planejar compras, realizar objetivos de vida e encarar com menos dificuldades eventuais problemas (SANTOS, 2014).

O autor ainda afirma que o planejamento financeiro é visualizado na planilha ou formulário do orçamento, que apresenta a comparação entre a renda total e a despesa total em um dado período. Esse autor enfatiza que o valor mais importante no orçamento é o saldo líquido, que consiste na diferença entre renda total e despesa total, em que um valor resultante positivo indica dinheiro sobrando e um valor negativo indica dinheiro faltando.

As contas do orçamento, sejam receitas ou despesas, podem ser classificadas em duas categorias (SANTOS, 2014):

- **Fixas:** contas em que o valor não se altera em um determinado período. Exemplos: salário, aluguel e financiamento.
- **Variáveis:** contas que o valor varia em função da frequência de utilização. Exemplos: pagamentos de prestação de serviços, oficina mecânica, combustível.

Santos (2014) fornece um modelo de planilha de orçamento financeiro familiar ou pessoal, que contém receitas e despesas distribuídas mensalmente e agrupadas em modalidades comuns, sendo as apresentadas pelo autor: de moradia, alimentação, transporte, saúde e cuidados; educação e especialização; lazer, vestuário e calçados; e obrigações financeiras.

Um exemplo da planilha de orçamento financeiro proposta por Santos (2014) pode ser observado na Tabela 3 e Tabela 4, no caso de despesas cada modalidade seria uma tabela, todavia é necessário colocar no lugar de "Receitas" a modalidade de despesa.

Tabela 3 – Modelo Planilha de Orçamento Financeiro.

Receitas	Janeiro	Fevereiro	...	Dezembro
Salário líquido	...	...	...	...
Férias	...	...	...	...
13º salário	...	...	...	...
Pensão	...	...	...	...
Dividendos	...	...	...	...
Outras receitas	...	...	...	...

Fonte: Adaptado de Santos (2014).

Tabela 4 – Modelo Planilha de Orçamento Financeiro Totais.

	Janeiro	Fevereiro	...	Dezembro
Despesa total	...	...	...	...
Receita total	...	...	...	...
Resultado líquido mensal	...	...	...	...

Fonte: Adaptado de Santos (2014).

Santos (2014) ressalta que para uma vida saudável é preciso não gastar mais do que ganha, ou seja, ter um saldo líquido positivo. Esta economia deve ser destinada para a formação ou reforço de uma reserva financeira que lhe apoiará ao longo da vida. Para o autor, é necessário ter conhecimento detalhado de sua situação financeira. Logo, é de suma importância a elaboração, utilização e monitoramento assíduo do orçamento financeiro.

Para Santos (2014) é essencial que parte dos valores poupados sejam destinados para: aposentadoria financeiramente saudável, seguro de bens patrimoniais, cuidado com a saúde, entre outros.

Algumas ações essenciais para evitar o desequilíbrio financeiro propostas por Santos (2014, p. 260) são:

- **Diariamente:**

  - Guardar comprovantes de gastos e registrá-los para o controle;

- **Mensalmente:**

  - Verificar se os rendimentos e gastos correspondem com o previsto;
  - Analisar as variações e refletir sobre as causas;
  - Ajustar o orçamento, se for preciso;

- **Anualmente:**

  - Elaborar orçamento para o próximo ano, incluindo rendas e despesas já conhecidas;
  - Definir uma margem líquida para aplicar na caderneta de poupança e/ou planos de previdência privada;
  - Definir uma margem líquida para eventualidades.

2.3.2 Finanças para pequenas empresas

Um transportador autônomo de cargas pode ser entendido como uma empresa, geralmente pequena, pois a relação de trabalho com as empresas de transporte de cargas é de natureza comercial, por meio de prestação de serviços (BRASIL, 2007).

Gazzoni (2003) aponta a importância do fluxo de caixa como uma ferramenta que possibilita o planejamento e controle financeiro da empresa, que deve ser elaborado conforme o tamanho da empresa e necessidades dos gestores. A autora afirma que esta ferramenta permite que a empresa conheça qual capital necessário para lidar com seus compromissos, a liquidez da empresa e necessidades futuras.

Para Sanvicente e Santos (2000 apud GAZZONI, 2003, p. 40), o fluxo de caixa tem como objetivo básico: "a projeção das entradas e saídas de recursos financeiros para determinado período". Logo, em empresas de pequeno porte, o controle financeiro se assemelha bastante com o proposto por Santos (2014) para finanças pessoais.

Gazzoni (2003) conclui que o fluxo de caixa é de grande valia para o pequeno negócio, por ser facilmente entendido e realizado pelo gestor, além de auxiliar a empresa a planejar seu futuro, evitando discrepâncias entre suas receitas e despesas.

2.4 *Design* de interface

Nesta seção alguns aspectos relevantes da área de *design* para este trabalho serão abordados, sendo eles: usabilidade, experiência de usuário e processos de *design*.

2.4.1 Usabilidade e Experiência de usuário

Para Sharp, Rogers e Preece (2019) a usabilidade consiste em garantir que produtos interativos sejam fáceis de aprender, eficazes no uso e agradáveis na perspectiva dos usuários. De acordo com Barbosa et al. (2021), na área de Interação Humano-computador (IHC) a interpretação de quê os aspectos mais subjetivos relacionados aos usuários estão inclusos no critério de usabilidade é minoritária, estes aspectos vão além da satisfação do usuário com o sistema, como, por exemplo, seus sentimentos e estado de espírito, e também são importantes para a interação humano-computador.

Consequentemente, o termo experiência do usuário, do inglês *User Experience* (UX), é utilizado para englobar estas características, sendo definido por Nielsen e Norman (2023, tradução nossa) como algo que engloba todos os aspectos da interação dos usuários finais com a empresa, seus serviços e produtos. Uma propriedade importante que faz parte da usabilidade e experiência de usuário é a acessibilidade que será abordada na Seção 2.4.1.1.

2.4.1.1 Acessibilidade

A acessibilidade é uma característica que indica a capacidade que os diferentes tipos de usuários têm ao interagirem com algo, seja algo físico ou um software. Portanto, uma interface acessível não pode apresentar empecilhos que dificultem a interação e acesso à informação (BARBOSA et al., 2021; SHARP; ROGERS; PREECE, 2019).

Barbosa et al. (2021) afirmam que os usuários podem possuir limitações físicas, mentais ou de aprendizado que podem impedí-los de acessarem um sistema, sejam elas temporárias ou permanentes. Os autores ressaltam também a influência da idade dos usuários na capacidade de interação com um sistema, por exemplo, problemas associados à visão causados pelo envelhecimento.

De acordo com o gov.br (2023), a acessibilidade digital consiste na eliminação de barreiras na Internet, os sítios na *web* devem ser construídos de forma que as pessoas possam entender, navegar e interagir efetivamente com as páginas.

2.4.2 *Design* Dirigido por Objetivos

O *design* é um processo que envolve diversas atividades. As mais básicas são: entendimento do problema ou questão, proposta de uma solução e avaliação da solução proposta. Dessa forma, um processo de *design* em IHC define com mais detalhes como estas atividades devem ser executadas, a ordem de execução e os artefatos utilizados e criados por elas.

O *design* dirigido por objetivos é um processo de *design* que guia o *designer* a projetar a interface criativamente, auxiliando os usuários a atingirem seus objetivos. Di-

ferentemente de um sistema projetado e analisando as tarefas realizadas pelos usuários, este processo incentiva o *designer* a utilizar as tecnologias existentes para que os objetivos sejam realizados, independentemente das tarefas envolvidas (BARBOSA et al., 2021).

Para alcançar este *design* criativo é importante diferenciar tarefas e ações dos usuários de objetivos. Cooper et al. (2014 apud BARBOSA et al., 2021, p. 91) definem um objetivo como: "uma expectativa de uma condição final, em que ações e tarefas são passos intermediários (em diferentes níveis de organização) que ajudam alguém a atingir um objetivo ou conjunto de objetivos". Em suma, as tarefas são os meios no qual um usuário pode cumprir com os objetivos.

Além de atender às necessidades dos usuários, o processo sistemático do *design* dirigido por objetivos também é capaz de satisfazer os requisitos técnicos, de negócio e de organização, sendo dividido em seis fases: pesquisar, modelar, definir requisitos, projetar, refinar e manter (BARBOSA et al., 2021).

O objetivo da fase de pesquisa é conhecer os usuários, o domínio do sistema e o contexto de uso. Assim, torna-se possível definir as necessidades e objetivos das partes interessadas.

Na fase de modelagem os usuários, domínio do sistema e contextos de uso são modelados de modo a organizar as informações adquiridas na etapa de pesquisa, que representam os conceitos reais das partes interessadas. A partir dos artefatos produzidos é possível investigar e discutir sobre o conhecimento adquirido para alcançar uma compreensão suficiente do problema ou questão (BARBOSA et al., 2021).

Na fase de definição dos requisitos, deve-se analisar as informações coletadas e modeladas nas etapas anteriores para elicitar e conciliar, caso haja conflitos, os requisitos do usuário, de negócio e técnicos (BARBOSA et al., 2021).

Na fase de projeto, o *designer* tem que projetar uma interface com poucos detalhes (baixa fidelidade), focando na criação da estrutura e comportamento da interface, e posteriormente na etapa de refinamento o foco é no detalhamento da interface projetada (alta fidelidade) e verificação da coerência das tarefas, sendo importante envolver os usuários na avaliação (BARBOSA et al., 2021).

Barbosa et al. (2021) evidencia que por mais que não exista uma etapa de avaliação explícita no processo de *design* dirigido por objetivos, as avaliações devem ser realizadas em todas as fases, especialmente nas fases de projeto e refinamento.

3 Metodologia

A finalidade deste capítulo é apresentar as metodologias utilizadas para o desenvolvimento do software, discorrendo sobre a metodologia de pesquisa, metodologia de desenvolvimento, *design*, requisitos, arquitetura, ferramentas e o cronograma de atividades.

3.1 Metodologia de Pesquisa

Como abordado na Seção 1.5, este trabalho pode ser classificado a partir de sua abordagem, natureza, objetivos e procedimentos. Os termos supracitados serão detalhados no decorrer desta seção.

3.1.1 Abordagem da pesquisa

Quanto à abordagem, a presente pesquisa é definida como qualitativa. Gerhardt e Silveira (2009) afirmam que em pesquisas qualitativas, a representatividade numérica não é impactante, e sim o entendimento profundo de um determinado grupo social. Os autores complementam dizendo que os métodos qualitativos tentam explicar as razões dos fatos, mas não se submetem à prová-los, visto que nesse tipo de abordagem, os dados analisados são não-métricos.

Como descrito na Seção 1.4.1, o principal objetivo do trabalho era o desenvolvimento de um software (aplicativo) que permitisse um controle sistematizado do trabalho, principalmente nos aspectos financeiros dos caminhoneiros brasileiros. Dessa forma, para que fosse possível atingi-lo, era imprescindível conhecer e entender o contexto ao qual estes profissionais se encontram e, então, propor uma solução baseada em parâmetros qualitativos levantados a partir desse entendimento.

3.1.2 Natureza da pesquisa

Por se tratar do real desenvolvimento de um software, abrangendo inclusive a implementação e alguns testes iniciais ao seu uso, que auxiliaria os caminhoneiros com uma ferramenta simples de gestão financeira, o presente trabalho seria classificado como de natureza aplicada, que objetivaria gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específico e envolvendo verdades e interesses locais. (GERHARDT; SILVEIRA, 2009).

3.1.3 Objetivos da pesquisa

Gerhardt e Silveira (2009) afirmam que a pesquisa exploratória tem o objetivo de dispor uma maior familiaridade com o problema, de maneira a deixá-lo mais em evidência ou construir hipóteses se utilizando, por exemplo, de levantamentos bibliográficos.

Dessa forma, este trabalho foi assim classificado, visto que visava propor uma possível solução para um problema ou dificuldade recorrente enfrentado no cotidiano dos caminhoneiros. Um volume significativo de dados e informações foram obtidas por meio do levantamento bibliográfico apresentado no Capítulo 2, sendo integrado a este levantamento a interação com alguns desses profissionais que ampliaram o entendimento de tal realidade.

3.1.4 Procedimentos da pesquisa

Os tipos de pesquisa não são mutuamente exclusivos. Por exemplo: uma pesquisa pode ser, ao mesmo tempo, bibliográfica, documental, de campo e estudo de caso (MORESI, 2003).

Dessa forma, quanto aos procedimentos, este trabalho foi realizado envolvendo pesquisa bibliográfica e produção tecnológica com a elaboração de um aplicativo.

3.1.4.1 Pesquisa Bibliográfica

A pesquisa bibliográfica foi feita a partir do levantamento de referências teóricas já analisadas e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de *websites* (GERHARDT; SILVEIRA, 2009). Moresi (2003) acrescenta que esse tipo de pesquisa fornece instrumental analítico para outras pesquisas. A pesquisa bibliográfica deste trabalho foi explorada no Capítulo 2, no qual os principais aspectos e fundamentações necessárias para a melhor compreensão do contexto do trabalho foram apresentadas.

3.1.4.2 Produção Tecnológica

Serzedello e Tomaél (2011) definem a produção tecnológica como a geração de produtos e de processos tecnológicos, com o objetivo de auxiliar na resolução de problemas práticos. De acordo com os autores, essa produção tem como objetivo assistir à sociedade por meio do desenvolvimento de inovações, impactando os meios tecnológicos, econômicos e sociais.

3.2 Metodologia de Desenvolvimento de Software

As metodologias de desenvolvimento de software adotadas para a elaboração deste trabalho se complementam e possuem diversas semelhanças em seus valores e práticas, sendo cada uma delas influenciadas pelo movimento ágil que começou no início dos anos 2000 (FOWLER, 2019). De acordo com Kim et al. (2021) muitos enxergam o *DevOps* como a continuação lógica do movimento de desenvolvimento de software ágil que iniciou em sequência (anos 2001).

As metodologias *Extreme Programming* (XP) e *DevOps* foram escolhidas por proporcionarem uma forma ágil de desenvolvimento de software, baseada em ciclos curtos, que depende da comunicação, uso de testes automatizados, capacidade de reagir a requisitos que mudam rapidamente e por proporcionarem práticas para criar softwares de qualidade velozmente. Portanto, no contexto deste trabalho, todos os valores, princípios e práticas abordados na Seção 3.2.1 e Seção 3.2.2 foram utilizados no processo de desenvolvimento do software.

3.2.1 *Extreme Programming* (XP)

A metodologia de desenvolvimento XP é um estilo de desenvolvimento de software focada na aplicação excelente de técnicas de programação, comunicação efetiva e trabalho em equipe. O XP possui uma filosofia de desenvolvimento de software baseada nos valores de comunicação, *feedback*, simplicidade, coragem e respeito (BECK; ANDRES, 2004).

Beck e Andres (2004) dizem que o XP se diferencia de outras metodologias por conta de seus ciclos curtos de desenvolvimento, capacidade de agendar com flexibilidade a implementação de funcionalidades, dependência de testes automatizados que asseguram a evolução do sistema, dependência de um processo de *design* evolutivo, entre outros.

O XP é uma metodologia leve, de forma a trazer apenas o que é necessário para gerar valor para o cliente. Segundo Beck e Andres (2004, n.p, tradução nossa): não é possível carregar um monte de bagagem e se mover rapidamente. Assim, os autores ressaltam que não existe um processo de software "engessado", em que cada equipe de XP irá utilizá-lo de forma diferente, com diferentes níveis de sucesso. O XP pode funcionar bem com equipes de qualquer tamanho e se adapta a requisitos vagos ou que mudam rapidamente.

Beck e Andres (2004) dividem a explicação do XP em três partes: valores, princípios e práticas. Na Seção 1.5.2, está listado o que foi utilizado com mais ênfase neste trabalho, considerando o que é definido por estes autores:

- Valores

  - **Comunicação:** a comunicação pode ajudar a resolver um problema. É possível ouvir pessoas que tiveram problemas semelhantes no passado, como uma equipe é possível decidir o que fazer para resolver um problema. A comunicação é importante para criar o sentimento de equipe e colaboração;
  - **Simplicidade:** fazer o sistema ser o mais simples possível para resolver o problema atual. A simplicidade depende do contexto, se uma equipe é especialista em algum tópico, provavelmente, utilizar recursos mais avançados deste tópico será simples;

- Princípios
  - **Qualidade:** projetos não ficam devagar por exigirem alta qualidade, a equipe deve se orgulhar do trabalho que faz. O trabalho deve ser feito da melhor maneira possível, ou seja, da melhor forma que a equipe conseguir;
  - ***Baby steps*:** realizar a menor atividade possível que se saiba estar na direção correta, pois a sobrecarga de pequenos passos é menor do que a sobrecarga de um grande passo errado;

- Práticas
  - **Ciclo semanal:** planejar o trabalho semanalmente, em uma reunião no início da semana: revisar o progresso até então, selecionar histórias suficientes para uma semana, fragmentar as histórias em tarefas e estimá-las;
  - **Integração contínua:** integrar e testar mudanças após não mais de algumas horas, quanto maior o tempo para integrar, maior e mais imprevisível é o custo;
  - **Código compartilhado:** qualquer membro da equipe pode melhorar qualquer parte do sistema em qualquer momento.
  - **Código e testes:** manter apenas o código e os testes como artefatos permanentes, outros documentos podem ser gerados a partir do código e dos testes.

Para Beck e Andres (2004), a intenção é que os valores se apoiem. Melhorar a comunicação auxilia na aquisição da simplicidade, eliminando requisitos desnecessários das preocupações atuais, por sua vez atingindo a simplicidade faz com que se precise comunicar menos coisas. Ademais, a prática de integração contínua expressa o princípio de *baby steps*, por integrar e testar poucas horas de trabalho de cada vez.

3.2.2 *DevOps*

O *DevOps* representa uma convergência de diversos movimentos filosóficos e gerenciais, resultando em práticas técnicas e culturais, que aumentam a capacidade de uma

equipe em desenvolver aplicações rapidamente. Esta metodologia remove a separação entre a equipe de desenvolvimento e a equipe de operações, que eram tradicionalmente separadas. Os engenheiros trabalham em todo o ciclo de vida do software, do desenvolvimento à implantação, não limitados a uma única função (AWS, 2023; KIM et al., 2021).

Na Seção 1.5.2 foram listadas as práticas utilizadas com mais ênfase neste trabalho, sendo definidas por Kim et al. (2021) como:

- **Integração contínua:** esta prática é basicamente igual à prática de integração contínua apresentada na Seção 3.2.1;
- **Infraestrutura como código:** o trabalho de operações é automatizado e tratado como código da aplicação, empregando práticas modernas de desenvolvimento que podem ser aplicadas em todo o fluxo, permitindo um fluxo de implantação mais rápido, incluindo a integração contínua, entrega contínua e implantação contínua;
- **Testes automatizados rápidos e confiáveis:** visando desenvolver um produto com qualidade, mesmo em estágios iniciais, é necessário fazer com que os desenvolvedores escrevam testes automatizados como parte do dia a dia. O fluxo para a verificação do software possui as seguintes etapas: *build* (processo o qual artefatos executáveis são criados) e empacotamento do software, execução dos testes automatizados, análise estática de código, análise de duplicação e cobertura de testes. Em caso de sucesso, o pacote criado é implantado em um ambiente similar ao de produção.

3.3 *Design*

O processo *design* utilizado para o projeto da interface do software é o *Design* Dirigido por Objetivos. Este processo define seis etapas, até este ponto do trabalho, apenas as três primeiras foram realizadas: pesquisar, modelar e definir requisitos, sendo as demais etapas realizadas nos momentos seguintes.

Na Seção 2.2 foi relatado o resultado da pesquisa bibliográfica acerca dos caminhoneiros no Brasil. Com esta pesquisa, foi possível conhecer a rotina, relações de trabalho, dificuldades, questões financeiras e o perfil dos profissionais no país. Esta seção resume a etapa de pesquisa do processo de *design*.

A partir das informações coletadas foi possível observar a importância dos pontos levantados por Lara (2012), citado na Seção 1.3, acerca da acessibilidade do software para pessoas de meia-idade, que coincide com o perfil dos usuários levantados. Devido à agenda exaustiva dos caminhoneiros retratada na Seção 2.2.1 as tarefas do aplicativo na interface

devem ser o mais simples possível, assim o usuário irá perder menos tempo na realização da atividade e poderá focar no trabalho.

Baseado nas informações coletadas na etapa de pesquisa foi criado um elenco de personas, composto de duas personas primárias e uma *antipersona*, que representam a modelagem dos usuários do sistema e seus objetivos.

Uma persona é um personagem fictício que descreve um usuário típico do sistema, sendo úteis para representar usuários em discussões e manter todos focados no mesmo alvo. Já uma *antipersona* é utilizada para deixar claro que o sistema não está sendo projetado para este público. Uma persona pode ser definida pelos seguintes elementos: identidade, situação (ou *status*), objetivos, habilidades, tarefas, relacionamentos, requisitos e expectativas (BARBOSA et al., 2021). No Apêndice A deste trabalho podem ser observadas tais características como exemplo de personas criadas.

Com a representação em personas foi possível definir os objetivos dos perfis diferentes de usuários do aplicativo, podendo ser ressaltada a relevância do controle financeiro levantado como foco principal deste trabalho, além da importância destas atividades para todos os indivíduos envolvidos, conforme abordado na Seção 2.3.

3.4 Requisitos

Sommerviller e Sawyer (1997) definem os requisitos de software como descrições do comportamento do sistema, suas propriedades ou atributos, ou também como as limitações que podem ser enfrentadas no processo de desenvolvimento.

3.4.1 Elicitação dos Requisitos

Wiegers e Beatty (2013) afirmam que o coração do desenvolvimento de requisitos é a elicitação, que pode ser definida como as necessidades e limitações de várias partes interessadas no sistema de software. Os autores definem esse processo como sendo analítico e colaborativo, que inclui atividades para coletar, descobrir, extrair e definir requisitos.

Muitas são as técnicas para se elicitar requisitos. Wiegers e Beatty (2013) explicam que tais técnicas podem envolver tanto atividades facilitadoras, em que há uma maior interação com as partes interessadas para descobrir necessidades do sistema, como também podem envolver atividades independentes, no qual se trabalha sozinho para descobrir informações. Os autores argumentam que cada técnica oferece uma exploração diferente dos requisitos e podem revelar requisitos completamente diferentes.

Para a elicitação dos requisitos deste trabalho foram utilizadas as técnicas de introspecção, *brainstorming* e uma técnica proposta por Wiegers e Beatty (2013) para definição de requisitos de qualidade (detalhada na Seção 3.4.1.3).

3.4.1.1 Introspecção

De acordo com a Moraes (2010), a técnica de introspecção é baseada em se colocar no lugar do usuário enquanto o mesmo se encontra executando uma tarefa, utilizando equipamentos e outros recursos, ou seja, elucidar quais são as características que o sistema deve possuir para obter sucesso.

Por fornecer requisitos partindo da perspectiva dos usuários e por ter um perfil de usuário modelado a partir do uso de personas, explicada na Seção 3.3, a técnica de introspecção foi utilizada. Com isso, foram elicitados requisitos de acordo com as necessidades averiguadas.

3.4.1.2 *Brainstorming*

Barbosa et al. (2021) afirmam que uma sessão de *brainstorming* tem o objetivo de elicitar uma grande quantidade de opiniões dos integrantes em torno de um assunto central de forma descontraída e aberta. Além disso, a atividade pode ser realizada para todo tipo de produto ou serviço, sendo seu resultado uma lista de necessidades e desejos dos usuários. Com base nisso, para as sessões de *brainstorm*, foram criadas as seguintes perguntas guias:

- O que o aplicativo deve fornecer para os caminhoneiros?
- O que um usuário do aplicativo deve fazer?
- Quais são as restrições do aplicativo?
- Quando você pensa no aplicativo, o que vem na sua mente?
- Há alguma coisa que há em outros aplicativos que possa existir nesse?
- Se não houvesse limitação de recursos, quais tipos de funcionalidades seriam interessantes no aplicativo?

Essa técnica foi aplicada por ser uma técnica simples de se realizar, por focar na colaboração dos participantes para a definição dos requisitos. A partir das perguntas guias citadas e com o conhecimento adquirido acerca dos caminhoneiros no Brasil relatado na Seção 2.2, foi possível propor vários requisitos, que podem ser encontrados na Seção 3.4.3.

3.4.1.3 Atributos de Qualidade

Wiegers e Beatty (2013) trazem também uma perspectiva além das funcionalidades do software, afirmando que usuários também possuem expectativas, muitas vezes não declaradas, sobre quão bem o produto desenvolvido irá funcionar. Os autores exemplificam

dizendo que tais expectativas incluem o quão fácil o software é de usar, o quão rápido ele executa, o quanto ele raramente falha, como ele se comporta em condições inesperadas, e até o quão barulhento ele é.

Atributos de qualidade servem como origem de vários requisitos. Eles também conduzem muitas decisões de *design* e de arquitetura do software (WIEGERS; BEATTY, 2013). Por isso, foi considerado relevante para este trabalho priorizar os atributos de qualidade mais importantes e também definir requisitos para cada um deles, visando proporcionar uma boa experiência de uso do software para os usuários.

Por se tratar de um aspecto tão importante na concepção de um software, foi utilizada uma técnica apresentada por Wiegers e Beatty (2013), dividida em cinco passos: começar com um escopo abrangente, reduzir o escopo, priorizar a lista, elicitar expectativas por atributo e definir requisitos de qualidade.

3.4.1.3.1 Começar com um escopo abrangente

Wiegers e Beatty (2013) sugerem começar com uma grande lista de atributos de qualidade a serem considerados. Este passo reduz as chances de deixar uma característica de qualidade importante para trás. Os atributos definidos como escopo abrangente para este trabalho são:

- **Qualidade externa:**
  - **Disponibilidade:** capacidade do sistema de ter seus serviços disponíveis quando e aonde são necessários;
  - **Instalabilidade:** o quão fácil é instalar, desinstalar e reinstalar corretamente a aplicação;
  - **Integridade:** capacidade do sistema de se proteger contra perda e não acurácia de dados;
  - **Interoperabilidade:** o quão fácil o sistema consegue se conectar e trocar dados com outro sistemas ou componentes;
  - **Performance:** o quão rápido e previsível o sistema reage às ações dos usuários e outros eventos;
  - **Confiabilidade:** por quanto tempo o sistema consegue executar sem acontecer uma falha;
  - **Robustez:** o quão bem o sistema reage a condições de operação inesperadas;
  - **Proteção:** o quão bem o sistema se protege contra dano ou injúria;
  - **Segurança:** o quão bem o sistema se protege contra acesso não autorizado à aplicação e aos seus dados;

  - **Usabilidade:** o quão fácil é para as pessoas aprenderem, lembrarem e utilizar o sistema;

- **Qualidade interna:**

  - **Eficiência:** o quão eficiente o sistema é ao utilizar os recursos computacionais;
  - **Modificabilidade:** o quão fácil é manter, alterar, melhorar e reestruturar o sistema;
  - **Portabilidade:** facilidade de fazer o sistema funcionar em outros ambientes operacionais;
  - **Reusabilidade:** capacidade de utilizar componentes do sistema em outros sistemas;
  - **Escalabilidade:** facilidade do sistema em crescer para lidar com mais usuários, transações, servidores ou outras extensões;
  - **Verificabilidade:** o quão rapidamente desenvolvedores e testadores podem confirmar se o sistema foi implementado corretamente.

3.4.1.3.2 Reduzir o escopo

Neste passo deve-se discutir com as partes interessadas qual dos atributos são mais prováveis de serem importantes para o projeto. Na prática, alguns atributos serão claramente importantes, outros claramente descartáveis, porém é necessária uma discussão para aqueles atributos que gerem algum tipo de dúvida quanto à inclusão (WIEGERS; BEATTY, 2013).

Os atributos de qualidade julgados como mais importantes para este trabalho foram os seguintes:

- Usabilidade;
- Portabilidade;
- Confiabilidade;
- Modificabilidade;
- Integridade;
- Segurança.

3.4.1.3.3 Priorizar a lista

Wiegers e Beatty (2013) afirmam que priorizar os atributos pertinentes ajuda no foco para futuras discussões de elicitação. Eles explicam que essa priorização é feita através de uma tabela, em que se comparam todos os atributos fornecendo o símbolo de "menor que" (<) caso um atributo seja menos importante que o outro, e o acento circunflexo (^) caso o atributo seja mais importante que o outro. O resultado deste passo pode ser encontrado na Tabela 5.

Tabela 5 – Priorização de atributos de qualidade.

Atributo	**Pontuação**	**Usabilidade**	**Integridade**	**Portabilidade**	**Modificabilidade**	**Segurança**	**Confiabilidade**	**Eficiência**
Usabilidade	3		^	<	<	^	^	<
Integridade	6			<	<	<	<	<
Portabilidade	2				<	<	^	^
Modificabilidade	1					^	^	<
Segurança	2						^	^
Confiabilidade	5							<
Eficiência	2							

Fonte: Autoria própria.

A Tabela 5 define os *trade-offs* (conflitos entre os atributos de qualidade), ou seja, qual atributo será escolhido como prioridade caso seja necessário escolher entre um dos dois.

3.4.1.3.4 Elicitar expectativas por atributo

Para elicitar as expectativas de acordo com os atributos priorizados, o objetivo foi interpretar o que os usuários querem dizer ao falar que o software deve se amigável, rápido, confiável ou robusto. Perguntas que exploram as expectativas dos usuários podem levar a requisitos de qualidade específicos que ajudam os desenvolvedores a criarem um excelente produto (WIEGERS; BEATTY, 2013). As expectativas dos usuários elicitadas para este trabalho foram:

- **Usabilidade:**
  - A estrutura das tarefas do sistema devem estar bem definidas e simples;
  - Promover a eficiência do usuário em primeiro lugar, em relação a do computador;
  - Deve haver correspondência entre o sistema e as expectativas dos usuários;
  - O sistema deve ajudar o usuário a se recuperar de um erro, informando-lhe sobre o que ocorreu;
  - O sistema deve fornecer um equilíbrio entre o controle e liberdade do usuário;
  - O modelo conceitual da interface deve prezar por consistência, para facilitar o aprendizado do sistema;
  - O sistema deve manter um padrão e consistência em relação a esse padrão definido;
  - O sistema deve tornar visível para os usuários o que é possível realizar e como as ações devem ser feitas;
  - O sistema deve apresentar apenas o que for necessário para realização de uma tarefa;
  - O sistema deve possibilitar cadastro de gastos e receitas de forma *offline*.
- **Integridade:**
  - O sistema deve prover durabilidade dos dados cadastrados;
  - Os dados salvos no sistema devem ser consistentes;
  - Os dados informados pelos usuários devem ser os mesmos salvos no sistema;
  - Os dados salvos de forma *offline* devem ser atualizados com o servidor o mais rápido possível.

- **Modificabilidade:**
  - O código deve ser escrito com padrões bem definidos;
  - Os componentes do sistema sempre que possível devem ser construídos voltado ao reúso.
- **Portabilidade:**
  - O sistema deve funcionar tanto no Android quanto no iOS.
- **Eficiência:**
  - As colunas mais utilizadas em buscas devem possuir índices.
- **Segurança:**
  - Os dados sensíveis serão sempre criptografados;
  - Exclusão de conta deve ser confirmada por *e-mail*;
  - O banco de dados do sistema deve estar protegido com usuário e senha fortes;
  - Mudanças de senha devem primeiramente confirmar a senha atual do usuário;
  - Desativação de conta devem primeiramente confirmar a senha atual do usuário;
  - O usuário deve conseguir realizar conexão *login* pelo método de autenticação *passwordless*.
- **Confiabilidade:**
  - O sistema deve se recuperar de 90% das falhas;
  - Os códigos do sistema devem possuir testes automatizados com uma boa cobertura lógica;
  - O sistema deve se recuperar de uma falha catastrófica em menos de 2 dias.

3.4.1.3.5 Definir requisitos de qualidade

Este passo consiste em definir requisitos de qualidade para cada um dos atributos priorizados a partir das expectativas de usuários definidas no passo anterior (WIEGERS; BEATTY, 2013). O resultado da aplicação da técnica para esse passo, pode ser observado na Seção 3.4.3.2.

3.4.2 Priorização dos Requisitos

Como afirmam Wiegers e Beatty (2013), todo projeto com limitações de recursos precisa definir as prioridades das funcionalidades definidas para o software. A técnica de priorização ajuda a revelar objetivos concorrentes, resolver conflitos, planejar entregas incrementais, fazer o controle de escopo e fazer as decisões necessárias em relações a possíveis trocas de acordo com a necessidade de cada funcionalidade. Os autores complementam dizendo que quando as expectativas das partes interessadas é alta e os prazos são curtos, sendo preciso ter certeza que o software entrega as funcionalidades mais críticas e que mais agregam valor o mais rápido possível.

Com base nisso, foi identificada a necessidade da priorização dos requisitos elicitados, com o objetivo de trazer as funcionalidades mais críticas e que agregariam mais valor para os usuários e auxiliaria na decisão em relação à ordem de implementação dos requisitos.

3.4.2.1 MoSCoW

Wiegers e Beatty (2013) explicam que as quatro letras principais na técnica de priorização conhecida como MoSCoW representam quatro classificações possíveis para os requisitos da seguinte maneira:

- ***M**ust* (Deve): O requisito deve ser satisfeito para a solução ser considerada um sucesso;
- ***S**hould* (Deveria): O requisito é importante e deveria ser incluído na solução se possível, mas não é obrigatório para se obter sucesso;
- ***C**ould* (Poderia): É uma funcionalidade desejável, mas uma que poderia ser deferida ou eliminada. Sua implementação depende do tempo e recursos;
- ***W**on't* (Não será): Indica que o requisito não será implementado em um dado momento, mas poderia ser incluso em uma próxima versão.

Por se tratar de uma técnica que é direta em representar quais são os requisitos primordiais para o bom funcionamento da aplicação, delimitando funcionalidades que poderiam não trazer tanto valor ao usuário, ou que poderiam ser restringidas em detrimento do tempo, essa foi a técnica de priorização utilizada neste trabalho. Os resultados podem ser encontrados na Seção 3.4.3.1.

3.4.3 Resultados dos Requisitos

A partir da utilização das técnicas de elicitação abordadas, os requisitos elicitados são apresentados na Seção 3.4.3.1 e Seção 3.4.3.2.

Como padrão de identificação dos requisitos, foi utilizada uma abordagem sequencial, na qual cada requisito possuia uma indicação de seu tipo seguido de um número sequencial (WIEGERS; BEATTY, 2013). Para requisitos funcionais, o tipo correspondeu ao acrônimo RF (Requisito Funcional) e para os requisitos de qualidade o acrônimo adotado foi RNF (Requisito Não Funcional).

3.4.3.1 Requisitos Funcionais

Requisitos funcionais descrevem os comportamentos observáveis que o sistema irá exibir em condições específicas e as ações que o sistema fornece aos usuários (WIEGERS; BEATTY, 2013). Os requisitos funcionais elicitados para este trabalho foram priorizados utilizando a técnica MoSCoW, abordada na Seção 3.4.2.1, da seguinte forma:

- ***Must*:**
  - **RF01** - Gestão de cadastro de usuários;
  - **RF02** - Fornecer categorias de gastos e receitas;
  - **RF03** - Fornecer grupos de categorias;
  - **RF04** - Gestão de fretes;
  - **RF05** - Gestão de gastos de um frete;
  - **RF06** - Fornecer indicadores de fretes em um período;
  - **RF07** - Fornecer a possibilidade de gerir gastos e receitas para fretes já finalizados;
  - **RF08** - Fornecer indicadores de gastos e receitas em um período;
  - **RF09** - Filtrar fretes por características;
  - **RF10** - Filtrar gastos e receitas por características;
  - **RF11** - O sistema deve atualizar os dados armazenados remotamente com os dados cadastrados de forma *offline* pelo usuário.
- ***Should*:**
  - **RF12** - Gestão de gastos e receitas (fixas e variáveis);
  - **RF13** - Fornecer resumo ao encerrar um frete;
  - **RF14** - Prover agenda de compromissos;
  - **RF15** - Gestão de informações dos veículos;
  - **RF16** - Estabelecer um teto de gastos por frete;
  - **RF17** - Fornecer a marcação do trajeto do frete baseado na origem e destino;

  - **RF18** - Fornecer indicadores personalizados.

- ***Could*:**

  - **RF19** - Estimativa de valor de frete;
  - **RF20** - Fornecer distância aproximada de um frete;
  - **RF21** - Fornecer previsão do tempo de acordo com o local atual e de destino;
  - **RF22** - Seleção de veículo padrão;
  - **RF23** - Alertar em caso de variação de gastos de uma categoria.

- ***Won't*:**

  - **RF24** - Gamificação para o cadastro dos gastos e receitas, possibilitando a realização dessas atividades de uma forma lúdica e menos onerosa;
  - **RF25** - Estimativa de valor de frete baseada em dados históricos;
  - **RF26** - Fornecer situação atual de estradas;
  - **RF27** - Prover agenda de compromissos de um frete;
  - **RF28** - Fornecer previsão para manutenção de peças e revisões.

Os requisitos estão descritos em um alto nível de abstração, não possuindo muitos detalhes acerca da implementação, pois este detalhamento ocorrerá nas iterações ágeis do projeto, conforme explicado na Seção 3.2.1. A palavra "gestão", quando utilizada nos requisitos, se refere às ações de criação, edição, remoção e exibição dos itens descritos no requisito.

3.4.3.2 Requisitos de Qualidade

Requisitos de qualidade especificam como, e o quão bem, o sistema desempenha suas funções. Por exemplo: ser rápido, confiável, fácil de utilizar, entre outros aspectos abordados por (WIEGERS; BEATTY, 2013). Os requisitos de qualidade elicitados para este trabalho forams:

- **Usabilidade:**

  - **RNF01** - O sistema deve reutilizar os componentes já criados que se encaixem na tarefa necessária;
  - **RNF02** - O sistema deve utilizar um conjunto de ícones e *assets* padronizados, sempre com o mesmo propósito;
  - **RNF03** - O sistema deve informar erros em campos de formulários;

  - **RNF04** - O sistema deve submeter requisições apenas se os campos forem válidos e estiverem corretos;
  - **RNF05** - O sistema deve possibilitar o uso de alguns recursos do sistema de maneira *offline*.

- **Integridade:**
  - **RNF06** - O sistema deve prover uma forma de armazenamento persistente que mantenha os dados dos usuários consistentes e íntegros.

- **Modificabilidade:**
  - **RNF07** - Os repositórios do sistema deverão estar configurados com ferramentas de análise estática de código, que verificam a conformidade com padrões de escritas definidos de acordo com padrões da comunidade da linguagem de programação utilizada.

- **Portabilidade:**
  - **RNF08** - O sistema deve funcionar tanto no sistema operacional Android quanto no iOS.

- **Segurança:**
  - **RNF09** - O sistema deve criptografar senha e outros dados sensíveis dos usuários;
  - **RNF10** - O sistema deve solicitar a senha atual do usuário antes de desativar ou excluir a conta;
  - **RNF11** - O sistema deve solicitar confirmação do usuário por *e-mail* antes de excluir a conta.

- **Eficiência:**
  - **RNF04** - O sistema deve submeter requisições apenas se os campos forem válidos e estiverem corretos;
  - **RNF12** - As colunas do banco de dados que forem utilizadas com frequência em consultas devem possuir índices.

- **Confiabilidade:**
  - **RNF13** - O código fonte do sistema deve possuir testes automatizados, com uma cobertura lógica de pelo menos 60%;
  - **RNF14** - O sistema deve executar os testes de forma automática integrada ao ciclo de desenvolvimento, executando toda a suíte de testes antes de integrar novas funcionalidades e correções de *bugs* ao código principal.

3.4.3.3 Verificação dos Requisitos

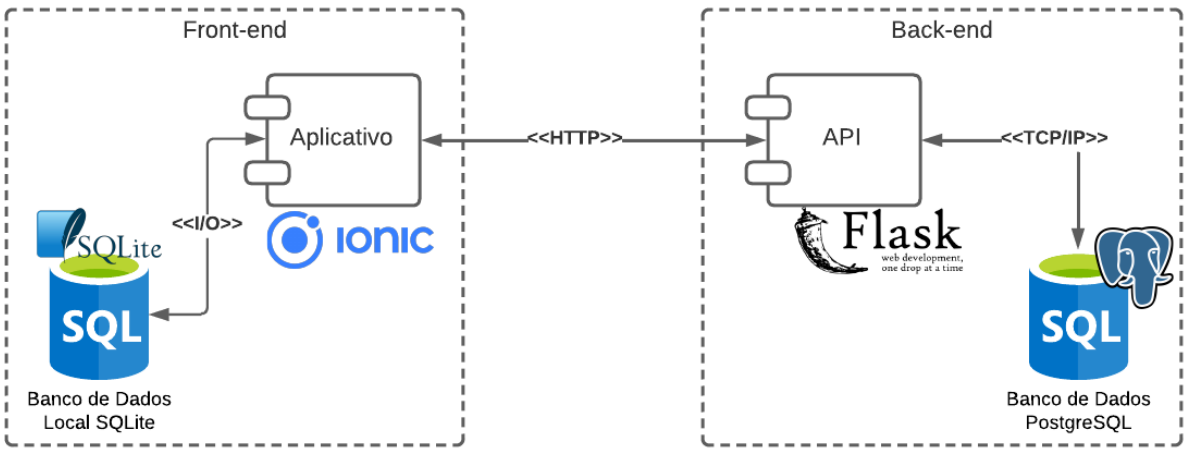
Wiegers e Beatty (2013) afirmam que se um requisito não é verificável, decidir se a implementação do mesmo foi correta se torna uma questão de opinião e não uma análise objetiva. Com base nisso, os requisitos definidos para este trabalho podem ser verificados a partir de testes de software, sejam eles automatizados ou testes de usabilidade, e também com o processo de inspeção, utilizando, por exemplo, a técnica de *checklist* de defeitos.

3.5 Arquitetura

A arquitetura geral do sistema pode ser visualizada na Figura 4, que contém os componentes básicos do sistema. A arquitetura obedece o modelo Cliente-Servidor, em que no caso o cliente é chamado de Aplicativo, enquanto o servidor é chamado de API (*Application Programming Interface*). A comunicação entre o Aplicativo e a API utilizarou o protocolo *Hypertext Transfer Protocol* (HTTP) e o padrão arquitetural *Representational State Transfer* (REST).

Os componentes no escopo "*Back-end*" são serviços disponíveis na Internet e estão implantados em nuvem, utilizando a ferramenta Heroku. Enquanto os componentes no escopo "*Front-end*" são serviços disponíveis no dispositivo móvel do usuário do sistema (cliente).

Figura 4 – Representação arquitetural.



Fonte: Autoria própria.

Ambos, *front-end* e *back-end*, utilizam o padrão arquitetural Active Record. Este padrão consiste em encapsular o acesso ao banco de dados em um objeto, lidando com os dados e comportamento. A estrutura de dados do Active Record deve corresponder a estrutura do banco de dados, um atributo em uma classe para cada coluna do banco de

dados (FOWLER, 2003). Para a implementação deste padrão no projeto foi estabelecido o uso da biblioteca SQLAlchemy para o *back-end* e a biblioteca TypeORM para o *front-end*.

3.5.1 Aplicativo

O componente "Aplicativo" é um software que executa no dispositivo móvel do usuário. É a interface gráfica pela qual os usuários irão interagir com o sistema. Foi desenvolvido utilizando a abordagem de aplicação híbrida, utilizando o *framework* Ionic.

3.5.2 SQLite

O banco de dados SQLite foi utilizado para permitir o armazenamento de dados quando o dispositivo móvel do usuário não estiver conectado à Internet. Estes dados são sincronizados com o banco de dados remoto (descrito na Seção 3.5.4) para satisfazer o requisito RF29 e RNF05.

O SQLite é um Sistema Gerenciador de Bancos de Dados Relacional (SGBDR) leve e rápido, porém implementa transações serializáveis que são atômicas, consistentes, isoladas e duráveis, reconhecidas como as propriedades ACID (*Atomicidade, Consistência, Isolamento, Durabilidade*). Mesmo que as transações deste SGBDR sejam interrompidas por uma falha no aplicativo, falha no sistema operacional ou falta de energia no dispositivo, sua integridade será preservada (SQLITE, 2021). Estas características do SQLite são suficientes para garantir os requisitos de integridade necessários para este trabalho.

3.5.3 API (*Application Programming Interface*)

O componente API é um servidor que recebe requisições HTTP do Aplicativo e as interpreta para executar as ações solicitadas. Ela foi desenvolvida utilizando o *framework* Flask. A arquitetura proposta para esta API se baseia nos padrões arquiteturais REST, *Model-View-Controller* (MVC) e Active Record.

3.5.3.1 MVC (*Model-View-Controller*)

O padrão arquitetural MVC consiste em três camadas de responsabilidades. Dadas as restrições definidas para o software, este padrão arquitetural apresenta a vantagem de ser simples e fácil de se utilizar (FOWLER, 2003).

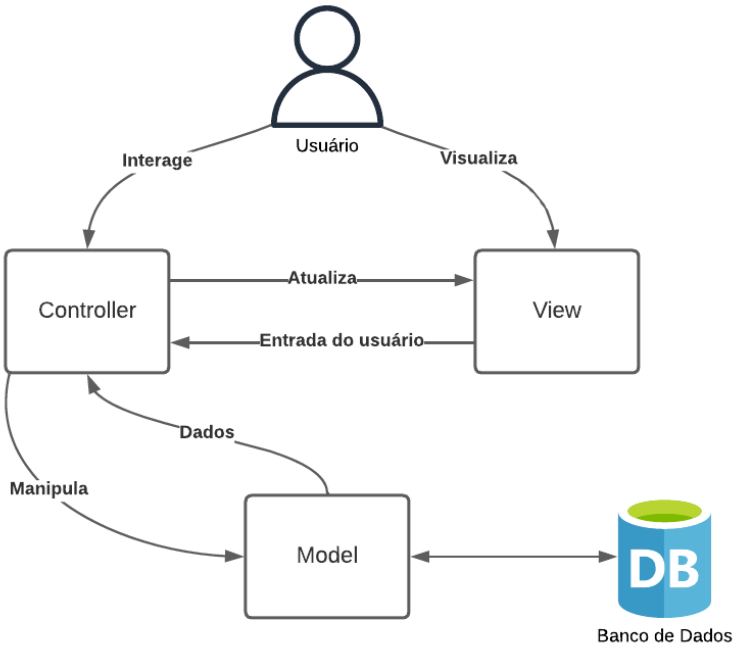
A *model* é um recurso que representa informações do domínio, não sendo visual e que contém todos os dados e comportamentos que não são utilizados pela interface gráfica (FOWLER, 2003).

A camada *view* representa a apresentação do modelo na interface, podendo ser uma tela com diversos *widgets*, uma página HTML ou qualquer outra forma de apresentação. A

view apenas apresenta as informações, as mudanças são tratadas pela camada *controller*. A *controller* trata as entradas dos usuários, manipula a *model* e faz com que a *view* atualize de acordo com as mudanças (FOWLER, 2003).

No contexto deste trabalho, o resultado da *view* é um JSON (*JavaScript Object Notation*) com as informações necessárias para cada solicitação feita pelo Aplicativo, enquanto as ações da *controller* serão iniciadas a partir das requisições HTTP recebidas pelo servidor.

Figura 5 – Representação do padrão MVC.



Fonte: Baseado em Fowler (2003).

Na Figura 5 é possível observar as relações entre o usuário e as camadas do padrão MVC.

3.5.4 PostgreSQL

Para o armazenamento remoto dos dados foi utilizado o Sistema Gerenciador de Banco de Dados Objeto-Relacional PostgreSQL. Este sistema gerenciador é de código aberto e também implementa transações ACID como SQLite (POSTGRESQL, 2023). Este SGBD (Sistema Gerenciador de Banco de Dados) é utilizado para o armazenamento remoto, garantindo persistência dos dados mesmo em caso de perda do dispositivo móvel, visando satisfazer o requisito RF29, enquanto o SQLite apenas armazena os dados no dispositivo do usuário, conforme apresentado na Seção 3.5.2.

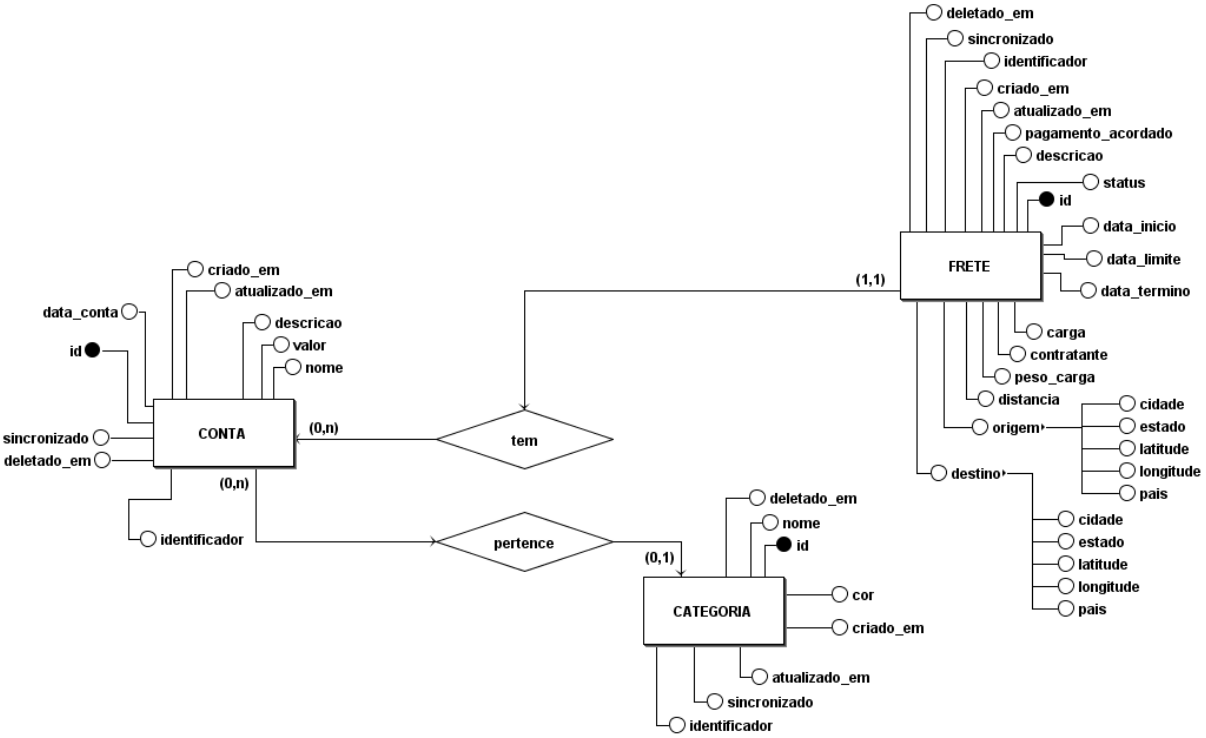
3.5.5 Modelagem dos Bancos de Dados

Por se tratar de dois bancos de dados, um para armazenamento local dos arquivos como especificado na Seção 3.5.2, e outro para o armazenamento remoto, como especificado na Seção 3.5.4, se fazem necessárias duas modelagens para contemplar cada um dos respectivos bancos de dados utilizados, pois mesmo que exista uma grande semelhança entre ambos, existem particularidades específicas para cada um deles em suas diferentes situações.

O Diagrama Entidade-Relacionamento (DE-R) representa o projeto no nível conceitual, demonstrando as entidades, seus atributos e relacionamentos, enquanto o Diagrama Lógico de Dados (DLD) representa o banco de dados em seu nível lógico e estabelece a estrutura dos dados que será implementado por meio de tabelas, seus relacionamentos, além dos atributos e seus tipos de dados, que constituem a camada de persistência do aplicativo.

3.5.5.1 Banco de Dados Local

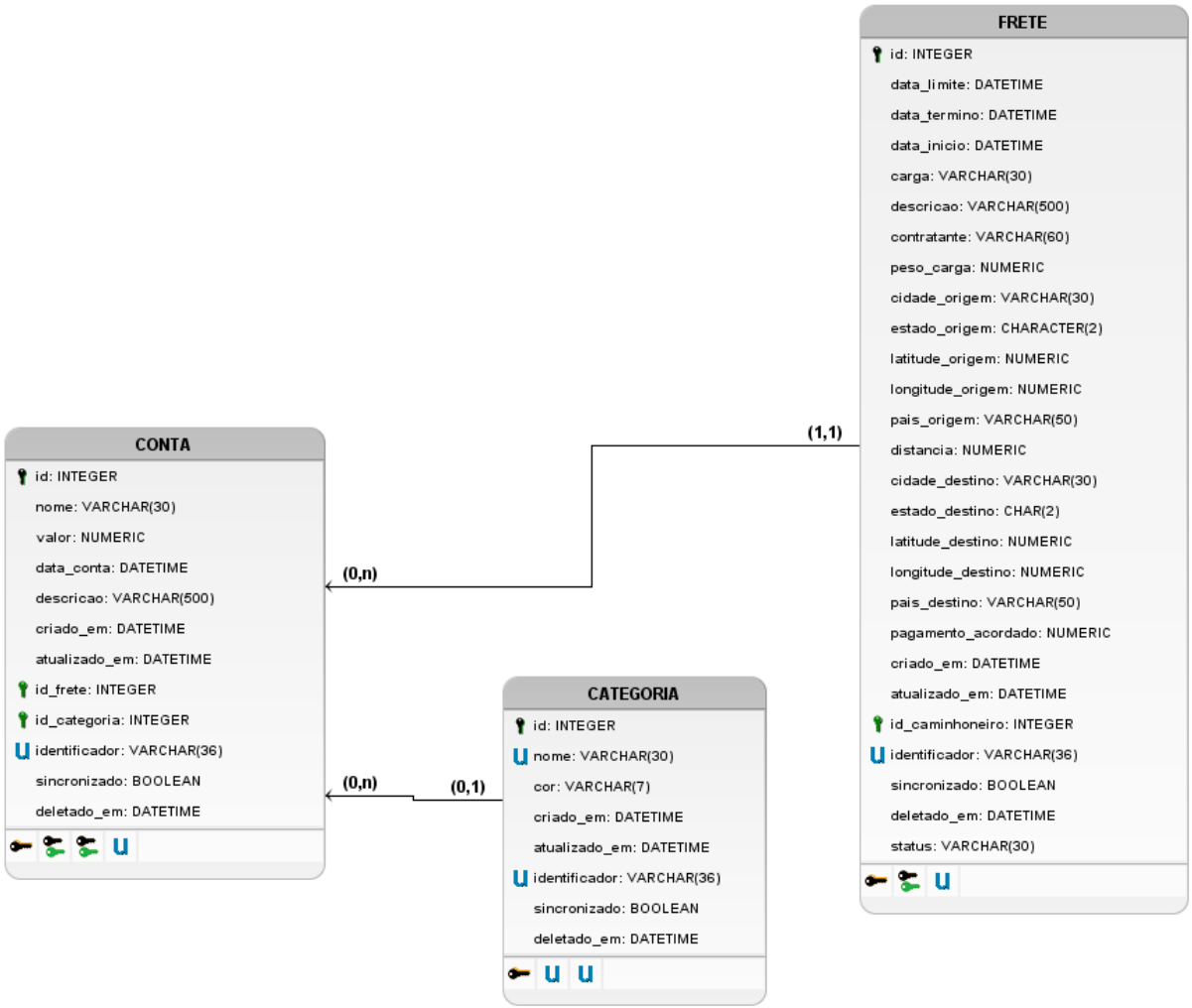
Figura 6 – Diagrama Entidade-Relacionamento local no dispositivo móvel.



Fonte: Autoria própria.

O banco de dados local é representado no nível conceitual na Figura 6 e em seu nível lógico na Figura 7. Por executar no dispositivo móvel do caminhoneiro, não é necessária uma relação para o usuário.

Figura 7 – Diagrama Lógico de Dados local no dispositivo móvel.

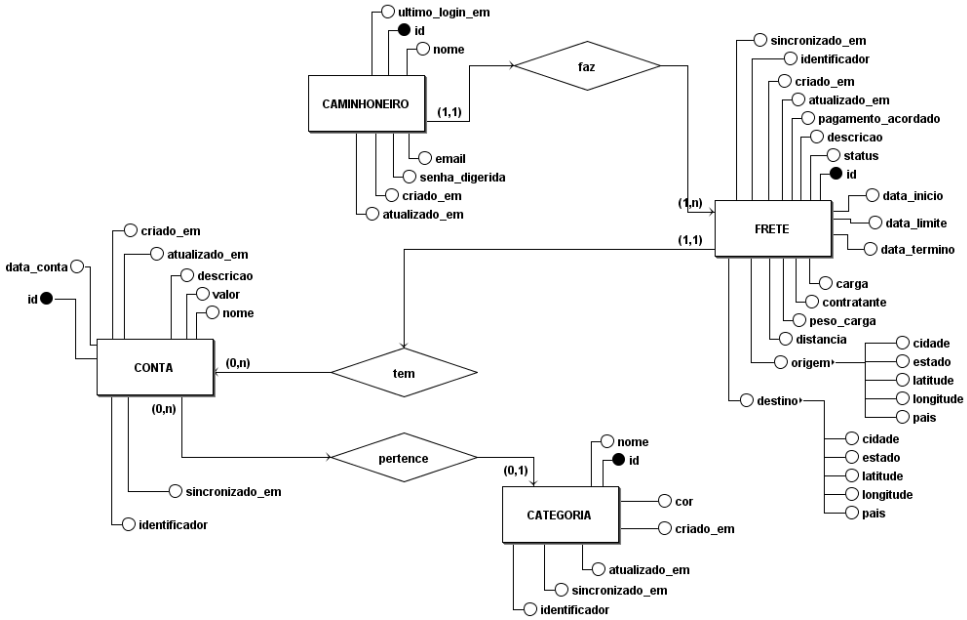


Fonte: Autoria própria.

3.5.5.2 Banco de Dados Remoto

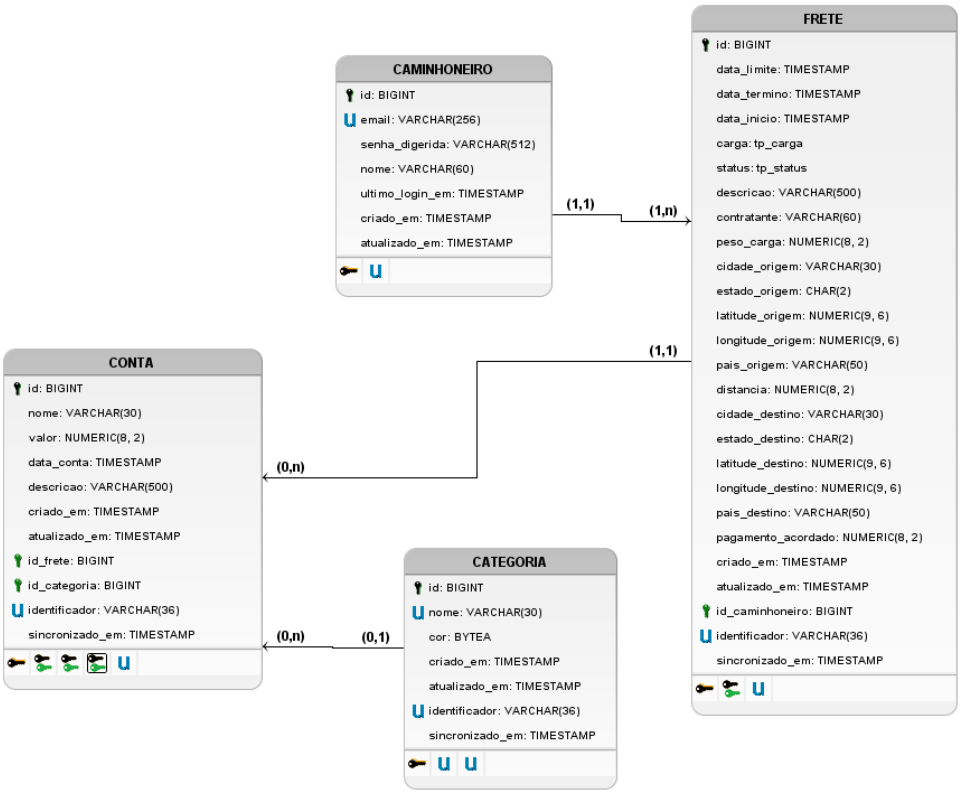
O banco de dados remoto armazena em nuvem os dados dos caminhoneiros, incluindo as contas de usuários, fretes, gastos, receitas e categorias. O projeto ao nível conceitual é apresentado na Figura 8 e em seu nível lógico na Figura 9.

Figura 8 – Diagrama Entidade-Relacionamento remoto da aplicação.



Fonte: Autoria própria.

Figura 9 – Diagrama Lógico de Dados remoto da aplicação.



Fonte: Autoria própria.

3.6 Suporte Tecnológico

3.6.1 Git e GitHub

O Git é um sistema de controle de versão distribuído gratuito e de código aberto capaz de lidar com projetos de todos os tamanhos tendo velocidade e eficiência (GIT, 2023).

O GitHub é uma plataforma de hospedagem de código para controle de versões e colaboração. O Git é o coração do GitHub, sendo o responsável pelo controle do GitHub (GITHUB, 2023). Para o controle de versão, hospedagem do projeto na nuvem e controle das histórias e tarefas de desenvolvimento, o GitHub foi utilizado.

3.6.2 Codecov

O Codecov é uma ferramenta para relatório de cobertura para qualquer suíte de testes, que possibilita aos desenvolvedores dados para implantar códigos confiáveis. a ferramenta é capaz de se integrar com GitHub e GitHub Actions para automação da coleta de dados de cobertura (CODECOV, 2023).

3.6.3 Visual Studio Code e Visual Studio Live Share

O Visual Studio Code (VSCode) é um editor de código-fonte que combina leveza e poder em uma solução de *desktop*, disponível para Windows, MacOS e Linux. Oferecendo suporte nativo para JavaScript e um ecossistema robusto de extensões para várias outras linguagens de programação, como C++, Java e Python, o VSCode é uma ferramenta versátil para desenvolvedores de software (VISUAL STUDIO CODE, 2023).

Além disso, o Visual Studio Live Share é uma extensão do VSCode que permite o desenvolvimento colaborativo em tempo real entre dois ou mais indivíduos. De acordo com a Visual Studio Code (2023), não importa a linguagem de programação, o sistema operacional ou o tipo de aplicativo sendo desenvolvido, essa extensão permite que os usuários compartilhem projetos com outros sem a necessidade de clonar repositórios ou configurar ambientes. Com sua ampla seleção de extensões para múltiplas linguagens de programação, suporte multiplataforma e integração com o GitHub, o VSCode se mostra como uma escolha ideal para o desenvolvimento de aplicativos colaborativos simultâneos.

3.6.4 brModelo

O brModelo é uma ferramenta CASE (*Computer-Aided Software Engineering*) auxilia no processo modelagem de banco de dados (CâNDIDO, 2020). Ele foi utilizado para a criação dos diagramas relacionados à modelagem dos dados como o Diagrama de Entidade e Relacionamento e o Diagrama Lógico de dados.

3.6.5 Lucidchart

Lucidchart ajuda seus usuários a criar e compartilhar fluxogramas profissionais (MCKINNON, 2023). Ele foi utilizado para a criação de diagramas e fluxogramas do projeto de maneira conjunta.

3.6.6 Heroku

Heroku é uma *Platform as a Service* (PaaS) que permite o *build*, execução e operação de aplicações inteiramente na nuvem. Foi utilizado para a implantação do componente API e do banco de dados PostgreSQL. (HEROKU, 2023)

3.6.7 Android Studio

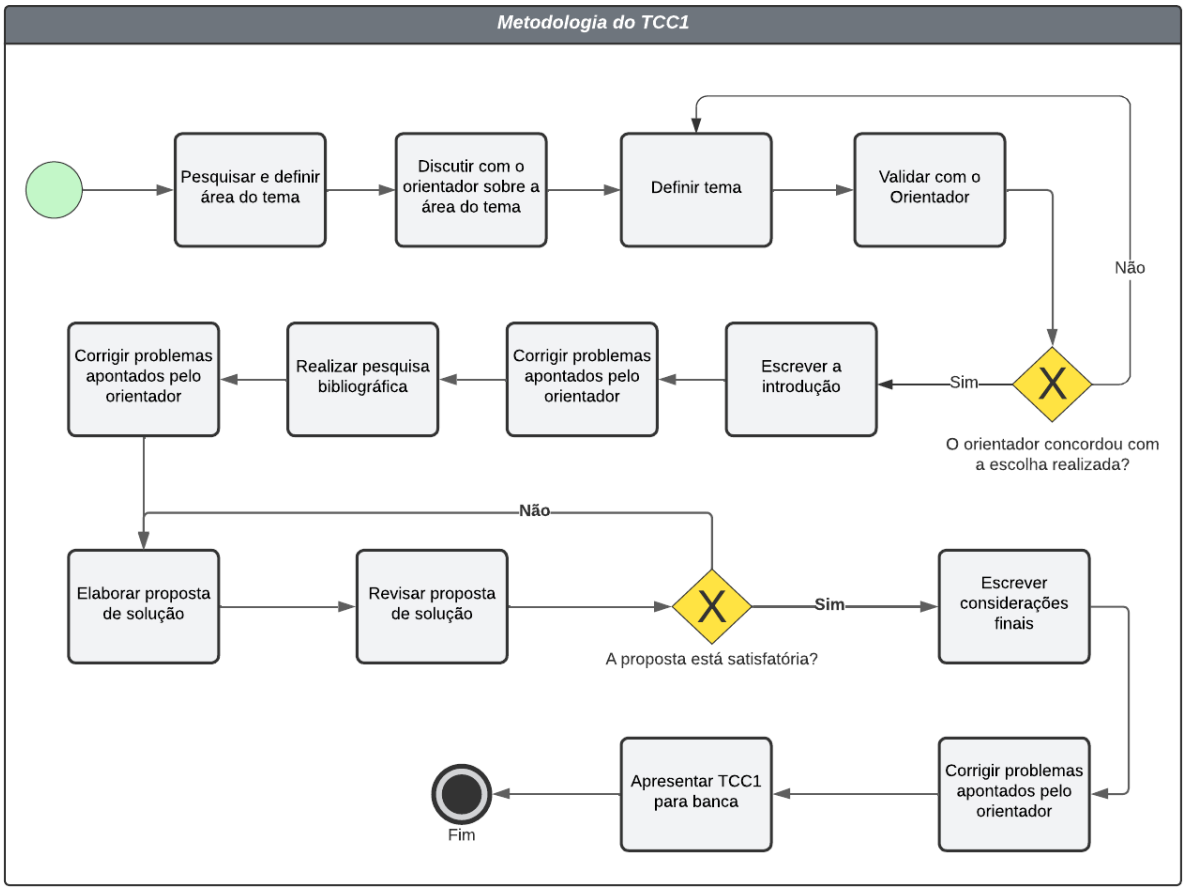
Android Studio é a IDE (*Integrated Development Environment*) oficial do sistema operacional Android. O objetivo desta plataforma é acelerar o desenvolvimento e ajudar os desenvolvedores a construir aplicativos de alta qualidade para dispositivos Android. Esta ferramenta foi utilizada para simular a execução em um dispositivo e gerar o instalador para o sistema Android (ANDROID STUDIO, 2022).

3.7 Processo Metodológico

Este trabalho (TCC) foi elaborado em duas etapas, denominadas TCC1 e TCC2. Na primeira etapa (TCC1) foi elaborada a proposta do trabalho enquanto na segunda (TCC2) foi desenvolvido o projeto proposto. A Seção 3.7.1 e a Seção 3.7.2 apresentam, respectivamente, cada uma dessas etapas deste trabalho.

3.7.1 TCC1

Figura 10 – Metodologia para elaboração do TCC1.



Fonte: Autoria própria.

As atividades presentes na Figura 10 são descritas a seguir:

- **Definir área do tema:** pesquisar e discutir acerca da área de pesquisa do trabalho;
- **Discutir com o orientador sobre a área do tema:** discutir com orientador sobre as possíveis áreas de pesquisa do trabalho;
- **Definir tema:** discutir ideias de possíveis temas de trabalho e escolher uma;
- **Validar com o Orientador:** validar com o orientador se o tema do trabalho é coerente com o esperado pela universidade;
- **Escrever a introdução:** descrever resumidamente o contexto do projeto proposto;
- **Realizar pesquisa bibliográfica:** detalhar as fundamentações teóricas necessárias para um melhor entendimento do contexto do trabalho;
- **Elaborar proposta de solução:** apresentar a proposta do software a ser desenvolvido;

- **Revisar proposta de solução:** revisar a proposta em busca de problemas que devam ser corrigidos;
- **Corrigir problemas apontados pelo orientador:** realizar correções apontadas pelo orientador do trabalho;
- **Escrever considerações finais:** descrever de maneira sucinta e objetiva o que foi alcançado com a proposta de trabalho (TCC1);
- **Apresentar TCC1 para banca:** realizar apresentação da proposta para a banca examinadora;

3.7.1.1 Cronograma TCC1

Tabela 6 – Cronograma de Atividades TCC1.

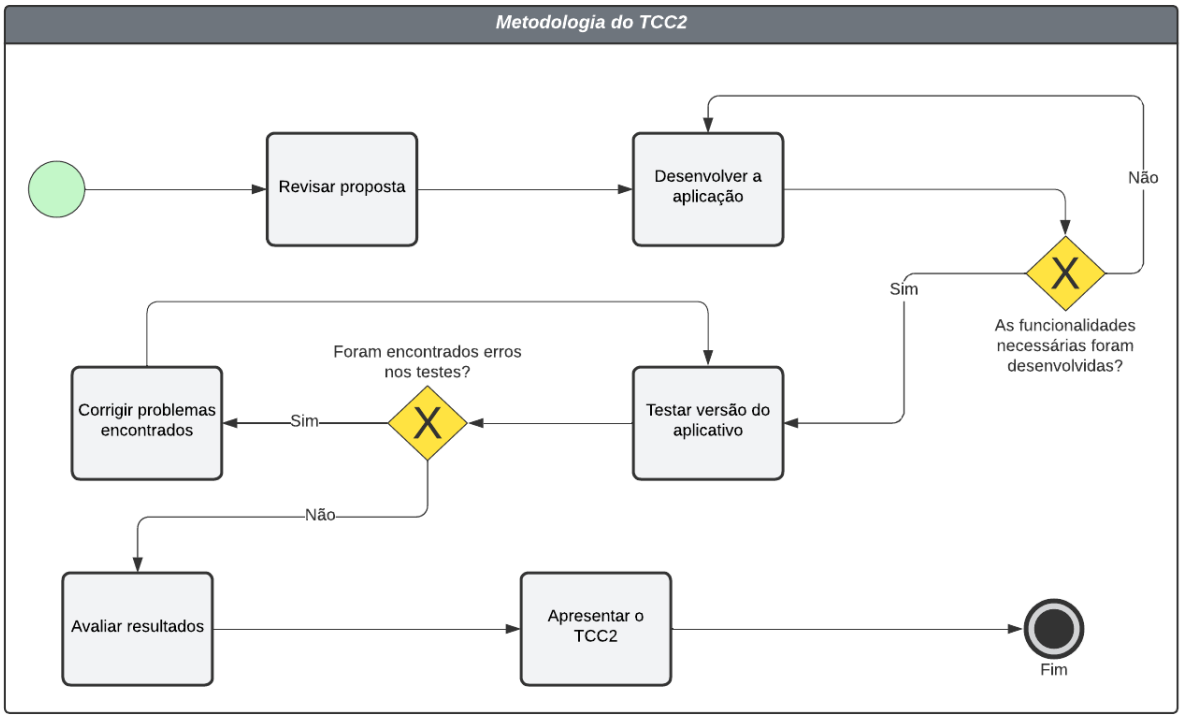
Atividade	Out	Nov	Dez	Jan	Fev
Definir área do tema	X	X			
Discutir com o orientador sobre a área do tema	X	X			
Definir tema	X	X			
Validar com o Orientador		X			
Escrever a introdução		X			
Realizar pesquisa bibliográfica		X	X	X	
Elaborar proposta de solução				X	
Revisar proposta de solução				X	
Escrever considerações finais				X	
Corrigir problemas apontados pelo orientador				X	X
Apresentar TCC1 para banca					X

Fonte: Autoria própria.

A primeira etapa foi concluída com a apresentação da proposta para a banca avaliadora em fevereiro de 2023, conforme cronograma do TCC1 disponível na Tabela 6. A aprovação aconteceu com a implementação da simulação arquitetural de alguns casos de uso previstos no projeto, como pode ser observado no Apêndice B deste trabalho.

3.7.2 TCC2

Figura 11 – Metodologia para elaboração do TCC2.



Fonte: Autoria própria.

As atividades presentes na Figura 11 esclarecem os processos da segunda etapa do trabalho a seguir:

- **Revisar proposta:** corrigir possíveis apontamentos e solicitações da banca após a apresentação do TCC1 e gerar uma nova versão que esteja de acordo com a proposta aprovada pela mesma;
- **Desenvolver aplicativo:** realização das iterações ágeis de desenvolvimento, com implementação de funcionalidades com testes automatizados. A metodologia é melhor abordada na Seção 3.2.1;
- **Gerar versão do aplicativo:** gerar a primeira versão da aplicação, que possa ser testada e evoluída em um dispositivo móvel;
- **Testar versão do aplicativo:** realizar os testes necessários para ver se a aplicação se comporta da maneira esperada;
- **Ajustar e incrementar a partir dos resultados do teste:** corrigir e evoluir a solução de acordo com o resultado dos testes da versão;
- **Avaliar resultados:** avaliar o software desenvolvido e descrever os resultados alcançados;

- **Apresentar TCC2:** realizar apresentação dos resultados para a banca examinadora;

3.7.2.1 Cronograma TCC2

Tabela 7 – Cronograma de Atividades TCC2.

Atividade	Mar	Abr	Maio	Jun	Jul
Revisar a proposta	X	X			
Desenvolver a aplicação		X	X	X	
Gerar versão do aplicativo			X	X	X
Testar versão do aplicativo			X	X	X
Ajustar e incrementar a partir dos resultados do teste				X	X
Avaliar resultados				X	X
Apresentar o TCC2					X

Fonte: Autoria própria.

A segunda etapa deste trabalho está sendo concluída também conforme o cronograma apresentado na Tabela 7, ao qual estará sendo apresentado para nova banca avaliadora. Os esclarecimentos mais detalhados ao processo de desenvolvimento efetivado no período previsto na Tabela 7, além de algumas evoluções e ajustes necessários à implementação proposta na primeira etapa (TCC1) são abordados no Capítulo 4.

4 Desenvolvimento

Este capítulo descreve os aspectos mais relevantes no processo de desenvolvimento do software utilizando as metodologias definidas no Capítulo 3. Como descrito na Seção 3.6.1 o GitHub foi utilizado para hospedar o código-fonte e controlar as histórias e tarefas de desenvolvimento, podendo os repositórios serem acessados nos seguintes endereços virtuais:

- Tarefas e documentação: https://github.com/tcc-fga-igor-paiva-thiago-lopes/docs
- Repositório *front-end*: https://github.com/tcc-fga-igor-paiva-thiago-lopes/frontend
- Repositório *back-end*: https://github.com/tcc-fga-igor-paiva-thiago-lopes/backend

4.1 Etapas do desenvolvimento

Como citado na Seção 3.2, uma das metodologias de desenvolvimento adotadas para a execução deste trabalho foi o XP, com uma estratégia de iterações de desenvolvimento semanais, em que cada integrante ficaria responsável por desenvolver tarefas que contemplam os requisitos definidos na Seção 3.4. Ao final de cada iteração, uma reunião entre os integrantes foi realizada, de forma a dar prosseguimento com a execução do trabalho, atribuindo novas tarefas, fazendo modificações pontuais, ou continuando com as tarefas atuais, caso não tenham sido finalizadas.

Para que um requisito fosse cumprido, ele deveria ser revisado pelo outro integrante da dupla e devidamente testado, de forma a cumprir com o requisito de qualidade RNF13 descrito na Seção 3.2 e com a própria metodologia de desenvolvimento indicada.

4.1.1 Alterações nos requisitos

Durante o desenvolvimento, notou-se que alguns dos requisitos priorizados inicialmente na Seção 3.4.3.1 como *Must* não agregariam valor ao usuário neste momento, sendo eles:

- RF03: este requisito, no escopo definido para execução durante este trabalho, não agregaria valor ao usuário, visto que todos os gastos inseridos pelos usuários serão referentes a uma viagem. Dessa forma, categorizar o gasto por uma categoria já satisfaria a necessidade do usuário;
- RF08 e RF10: estes requisitos foram movidos para a categoria *Should* por dependerem do requisito RF12, que foi inicialmente priorizado nessa categoria.

4.1.2 Iterações ágeis

Os requisitos implementados foram desenvolvidos nas seguintes iterações ágeis do projeto:

- 1ª iteração:
  - RF01 - Gestão de cadastro de usuários;
  - RF04 - Gestão de fretes;
  - RF07 - Fornecer a possibilidade de gerir gastos e receitas para fretes já finalizados.
- 2ª iteração:
  - RF01 - Gestão de cadastro de usuários;
  - RF04 - Gestão de fretes.
- 3ª iteração:
  - RF01 - Gestão de cadastro de usuários;
  - RF02 - Fornecer categorias de gastos e receitas.
- 4ª iteração:
  - RF02 - Fornecer categorias de gastos e receitas;
  - RF04 - Gestão de fretes.
- 5ª iteração:
  - RF02 - Fornecer categorias de gastos e receitas;
  - RF04 - Gestão de fretes.
- 6ª iteração:
  - RF02 - Fornecer categorias de gastos e receitas;
  - RF04 - Gestão de fretes.
- 7ª iteração:
  - RF11 - O sistema deve atualizar os dados armazenados remotamente com os dados cadastrados de forma *offline* pelo usuário;
  - RF02 - Fornecer categorias de gastos e receitas.
- 8ª iteração:

  - RF11 - O sistema deve atualizar os dados armazenados remotamente com os dados cadastrados de forma *offline* pelo usuário;
  - RF02 - Fornecer categorias de gastos e receitas.

- 9ª iteração:
  - RF09 - Filtrar fretes por características;
  - RF11 - O sistema deve atualizar os dados armazenados remotamente com os dados cadastrados de forma *offline* pelo usuário;

- 10ª iteração:
  - RF05 - Gestão de gastos de um frete;
  - RF07 - Fornecer a possibilidade de gerir gastos e receitas para fretes já finalizados.

- 11ª iteração:
  - RF05 - Gestão de gastos de um frete;
  - RF07 - Fornecer a possibilidade de gerir gastos e receitas para fretes já finalizados;

- 12ª iteração:
  - RF06 - Fornecer indicadores de fretes em um período.

4.2 Processo de testes

Conforme descrito na Seção 3.2 as metodologias de desenvolvimento utilizadas neste projeto se baseiam no desenvolvimento contínuo de testes automatizados, dessa forma o requisito RNF14, apresentado na Seção 3.4.3.2, define a execução de toda a suíte de testes, bem como análise a estática de código, em cada nova integração na *branch* principal do produto. Para isso foi utilizado o GitHub e sua ferramenta interna GitHub Actions, que permite a execução de ações automáticas ligadas a eventos da ferramenta, toda vez que uma solicitação de integração de alguma melhoria ou *bug* são executados:

- **Testes automatizados** - todos os testes automatizados são executados e indicam uma falha quando qualquer um deles falhar. Para a execução dos testes, o código-fonte é copiado, as dependências são instaladas, os testes são executados e os dados enviados para o Codecov;

- **Relatório de cobertura de testes** - A ferramenta Codecov traz um relatório com a cobertura de cada arquivo modificado e cobertura geral;

- **Análise estática** - utilizando ferramentas comuns da comunidade, Flake8 para o *back-end* e o ESLint para o *front-end*, estas ferramentas são utilizadas para impor padrões de escrita de código, identificar variáveis e funções não utilizadas, possíveis *bugs* e erros de programação.

Durante o desenvolvimento da aplicação foram desenvolvidos vários tipos de testes para tentar cobrir da melhor forma possível de todos os cenários e requisitos definidos: testes unitários, testes de integração e testes ponta a ponta.

De acordo com Myers, Badgett e Sandler (2012), um teste de módulo ou teste unitário é um processo de testar individualmente subprogramas, sub-rotinas, classes e procedimentos em um programa, ao invés de testar o programa por inteiro, sendo o teste focado nas pequenas partes do programa.

Testes de integração focam em achar erros quando módulos desenvolvidos de forma independente são conectados uns com os outros, em que o objetivo é testar se o máximo de módulos desenvolvidos separadamente funcionam conforme o esperado (FOWLER, 2018).

Testes ponta a ponta também são conhecidos como *end-to-end test*, *full-stack test* e *broad-stack test*, sendo testes que exercitam a maioria das partes de uma aplicação grande. Esses testes geralmente manipulam a aplicação por meio de uma interface, tal como ferramentas de aplicações web, como no Selenium. Um teste que exercita uma aplicação via uma interface de serviço também pode ser considerado um teste ponta a ponta do servidor (MYERS; BADGETT; SANDLER, 2012).

Testes automatizados unitários e de integração foram desenvolvidos tanto para o *front-end* quanto para o *back-end*, testando os módulos da aplicação, utilizando o Jest e o Pytest, respectivamente. Testes ponta a ponta foram realizados apenas para o servidor, simulando requisições e verificando a resposta e efeitos da chamada no banco de dados. Devido à complexidade de configuração do banco de dados PostgreSQL e pelo fato de utilizar o SQLite no cliente, no ambiente de testes do servidor o SQLite foi utilizado.

No contexto do *front-end* foram realizados testes automatizados dos componentes da interface utilizando a ferramenta Vue Test Utils do *framework* Vue.js. Essa ferramenta permite que os componentes sejam montados da forma semelhante a que ocorre na execução normal da aplicação. Com esse tipo de teste, é possível simular cliques, verificar a presença e conteúdo dos elementos da página, possibilitando testes mais fidedignos com a realidade.

4.3 Processo de *design*

Conforme o processo de *design* na Seção 3.3, para o projeto das interfaces gráficas, primeiramente as interfaces eram desenhadas com poucos detalhes (baixa fidelidade) e

posteriormente refinada para incluir mais detalhes (alta fidelidade). Para as telas mais simples do sistema o protótipo de baixa fidelidade era desenvolvido em código no *front-end*, porém com menos detalhes.

Para avaliar as interfaces projetadas foi utilizado um método de inspeção, mais especificamente uma avaliação heurística, que é um método de avaliação formativa, ou seja, que pode ser aplicado durante o processo de design e antes de uma solução concluída (BARBOSA et al., 2021). Esta técnica consiste em incorporar o papel do usuário final e avaliar se a interface está de acordo com as heurísticas de Nielsen, Maciel et al. (2004, p. 8) que contém dez heurísticas:

- ***Status* do sistema** - o usuário deve ser informado pelo sistema em tempo razoável sobre o que está acontecendo;
- **Compatibilidade do sistema com o mundo real** - o modelo lógico do sistema deve ser compatível com o modelo lógico do usuário;
- **Controle do usuário e liberdade** - o sistema deve tornar disponíveis funções que possibilitem saídas de funções indesejadas;
- **Consistência e padrões** - o sistema deve ser consistente quanto à utilização de sua simbologia e à sua plataforma de hardware e software;
- **Prevenção de erros** - o sistema deve ter um design que se preocupe com as possibilidades de erro;
- **Reconhecimento ao invés de relembrança** - as instruções para o bom funcionamento do sistema devem estar visíveis no contexto em que o usuário se encontra;
- **Flexibilidade e eficiência de uso** - o sistema deve prever o nível de proficiência do usuário em relação ao próprio sistema;
- **Estética e design minimalista** - os diálogos do sistema devem conter somente informações relevantes ao funcionamento;
- **Ajuda aos usuários no reconhecimento, diagnóstico e correção de erros** - as mensagens devem ser expressas em linguagem clara, indicando as possíveis soluções;
- **Ajuda e documentação** - a informação desejada deve ser facilmente encontrada, de preferência deve ser contextualizada e não muito extensa.

Para garantir o requisito RNF02, apresentado na Seção 3.4.3.2, bem como as heurísticas de “Consistência e padrões” e “Reconhecimento ao invés de relembrança” para a construção de todas as interfaces foi utilizado um conjunto padronizado de ícones e

componentes do *framework* utilizado (Ionic). Dessa forma, todas as páginas possuem o mesmo estilo e não causam estranheza aos usuários.

As telas do aplicativo se adéquam em parte às heurísticas de "Prevenção de erros" e "Controle do usuário e liberdade" por prevenir ações destrutivas com uma janela de confirmação. Os exemplos são o processo de *logout*, remoção de algum registro ou ações que desfaçam algo trabalhoso realizado anteriormente. Também a heurística "Ajuda aos usuários no reconhecimento, diagnóstico e correção de erros" foi respeitada no contexto de exibir nos formulários os erros de validação, colocando cada erro imediatamente abaixo do campo com uma descrição clara do problema(s).

A Seção 4.3.1 compõe alguns casos de aplicação do processo de *design* são detalhados.

4.3.1 Telas do aplicativo

4.3.1.1 Tela de gestão

No contexto do projeto existem várias entidades definidas na abstração da realidade para o software, cada uma dessas entidades precisam de ações de criação, listagem, edição e remoção por parte do usuário. Para isso foi criado um componente reutilizável para realizar essas ações para qualquer entidade, em que o protótipo desta tela pode ser observado na Figura 12.

Avaliando o protótipo apresentado na Figura 12, foram encontrados alguns problemas:

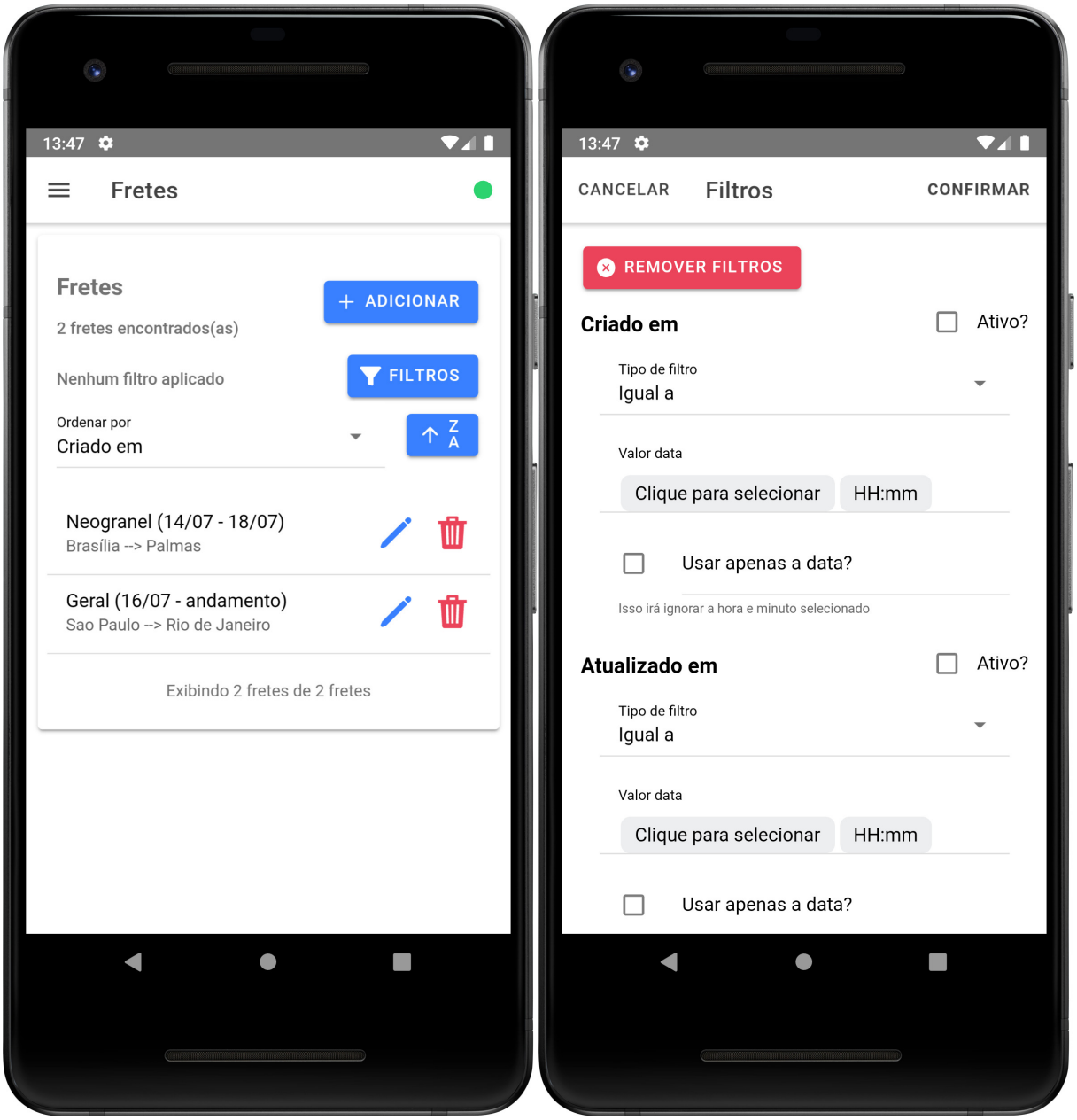
- Não são exibidos quantos registros estão sendo listados, relacionado a heurística de "*Status* do sistema". Este problema pode ser resolvido adicionando um texto com quantos registros foram encontrados e quantos estão sendo exibidos;
- Apenas um trecho de texto para descrever o registro pode não ser suficiente, como, por exemplo, nos fretes. Este problema está relacionado a heurística de "*Status* do sistema" e pode ser resolvido adicionando um outro texto, porém menor na linha do registro;
- É importante que, caso existam muitos registros, nem todos sejam carregados ao abrir a tela por questões de desempenho, que podem causar erros, relacionados à heurística de "Prevenção de erros".

Figura 12 – Protótipo de telas de gestão.

Fonte: Autoria própria.

Na Figura 13 é possível visualizar a tela de gestão de fretes, que utiliza o componente citado anteriormente. Nesta figura também é possível visualizar uma janela com as opções de filtros que podem ser aplicados na consulta, bem como os dados de filtro e ordenação fora da janela. Também é possível observar que na ordenação dos registros listados existe uma seta e duas letras indicando a ordem. As letras estão presentes para indicar se a ordem é crescente ou decrescente. Os filtros e ordenação não foram citados como problemas na interface, porque foram desenvolvidos em iterações posteriores ao protótipo inicial da tela.

Figura 13 – Telas de gestão de fretes à esquerda e de janela de filtros à direita.

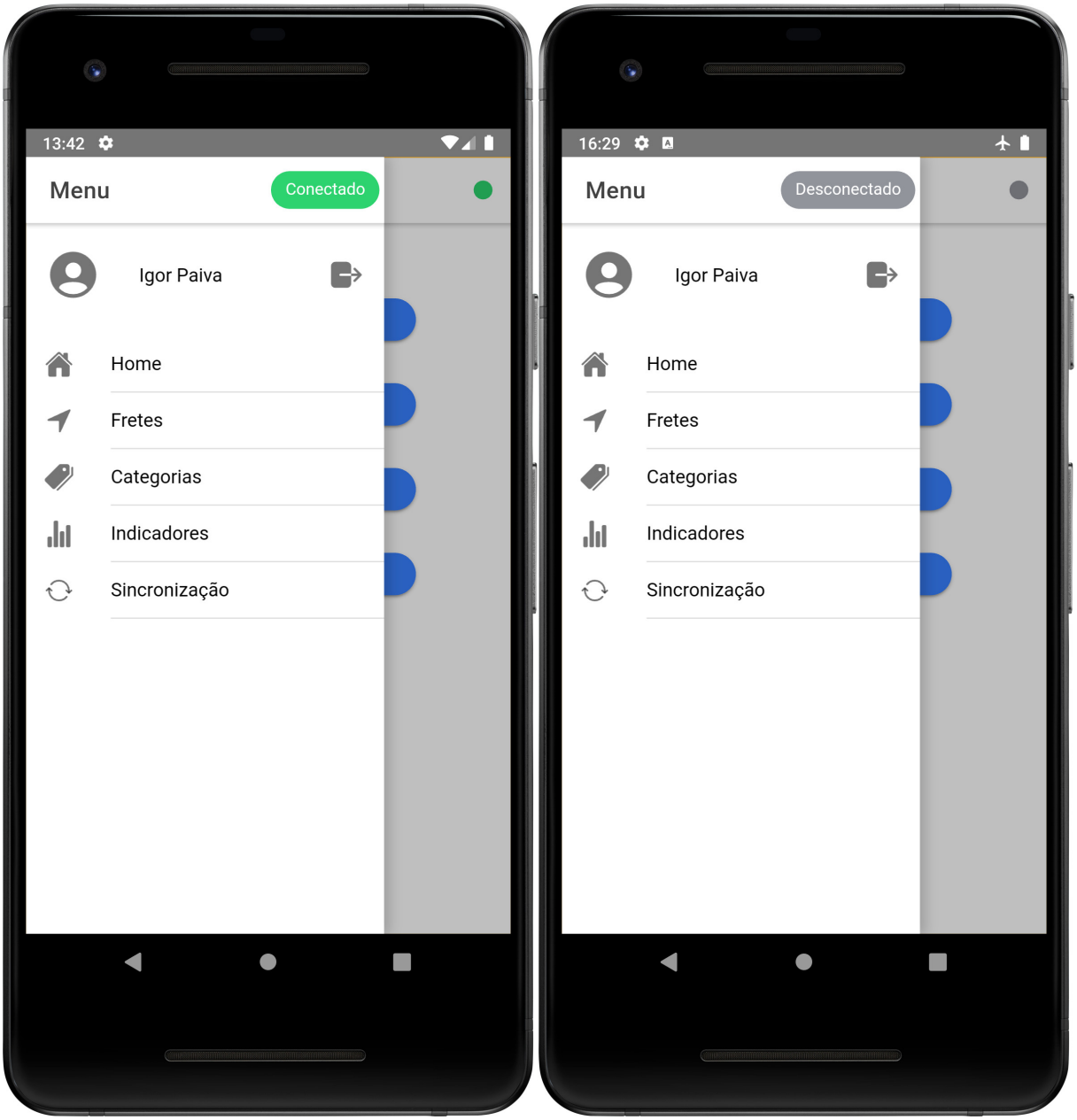


Fonte: Autoria própria.

4.3.2 *Status* de conexão com a Internet

Para fornecer ao caminhoneiro um indicador de sua conexão ao utilizar o aplicativo, foi criado um símbolo que indica quando o dispositivo está conectado ou não à Internet. Na Figura 14 é exibido o indicador na parte mais à direita do *menu* e nas demais telas do aplicativo, em que este símbolo foi adicionado avaliando a interface com a heurística de "*Status* do sistema".

Figura 14 – Telas indicando o *status* ou situação da conexão.

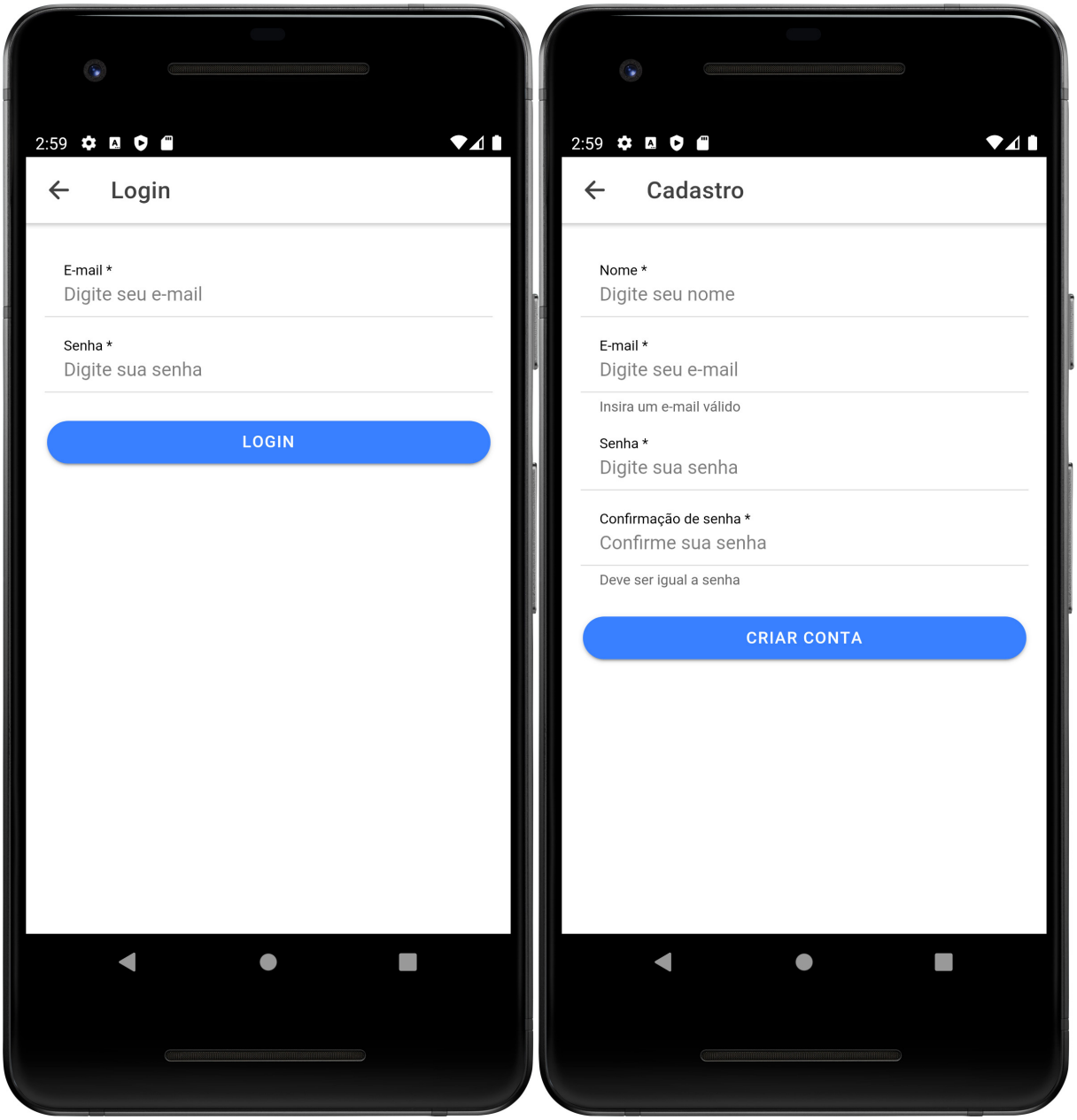


Fonte: Autoria própria.

4.3.2.1 Tela de *login* e cadastro

Para acessar o restante do aplicativo, o usuário necessita primeiramente criar uma conta e fazer login. Ambas as telas podem ser encontradas na Figura 15, a funcionalidade de cadastro foi desenvolvida na simulação apresentada no Apêndice B.

Figura 15 – Telas de login à esquerda e cadastro à direita.



Fonte: Autoria própria.

4.3.2.2 Formulário de fretes

Os fretes do caminhoneiro possuem diversos campos que podem ser preenchidos, portanto, ao criar o protótipo da tela foi preciso organizar todos os campos para que

o *design* fosse minimalista. Primeiramente, o protótipo começou com todos os campos, estando um abaixo do outro na tela. Porém, para não ficar confuso, foi dividido em dois passos: dados gerais do frete e dados de localização. Na parte de localização foi utilizado um *menu* suspenso para separar os dados da origem e destino. A Figura 16 exibe o protótipo com esses dois passos.

Figura 16 – Protótipo do formulário de fretes.

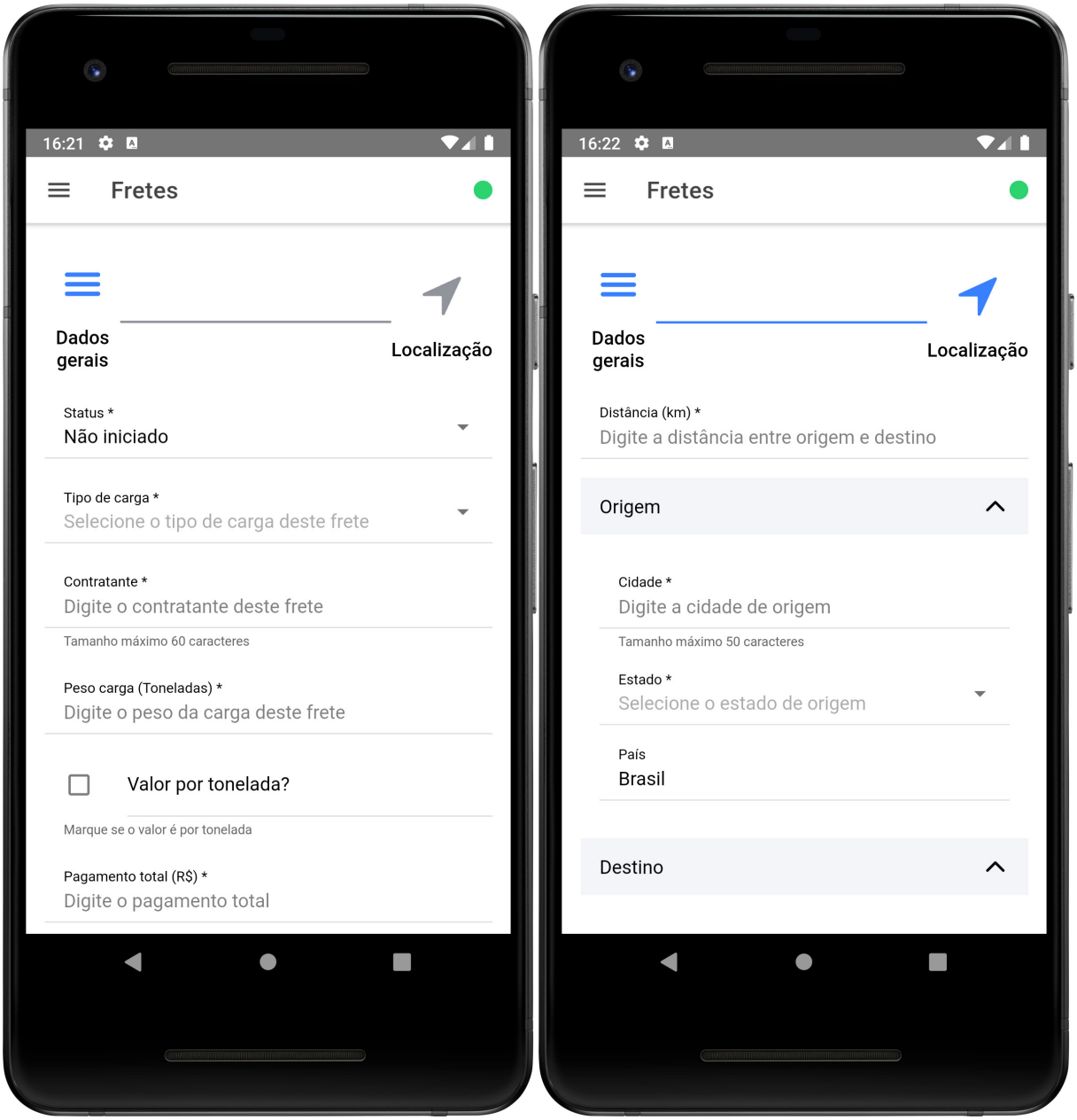
Fonte: Autoria própria.

Ao avaliar a interface projetada e apresentada na Figura 16 foram encontrados alguns problemas, listados abaixo. A versão após realizada todas as correções citadas pode ser visualizada na Figura 17.

- Apenas marcar o frete como finalizado, não era suficiente para organizar os possíveis estados do frete. Dessa forma, esse campo foi alterado para utilizar uma lista com valores possíveis: não iniciado, em progresso, aguardando descarga e finalizado;
- O tipo de carga como texto livre poderia gerar erros que dificultariam a análise dos fretes. Para prevenir esses erros, o campo foi mudado para uma lista com valores possíveis, utilizando como base os tipos de carga do aplicativo InfraBR citado na Seção 2.2.4;

- É comum que os caminhoneiros sejam pagos ou negociem os valores por tonelada de carga. Sendo assim, foi adicionado uma caixa de seleção na interface para ser possível marcar se o valor informado é o valor total ou por tonelada.

Figura 17 – Formulário de fretes.

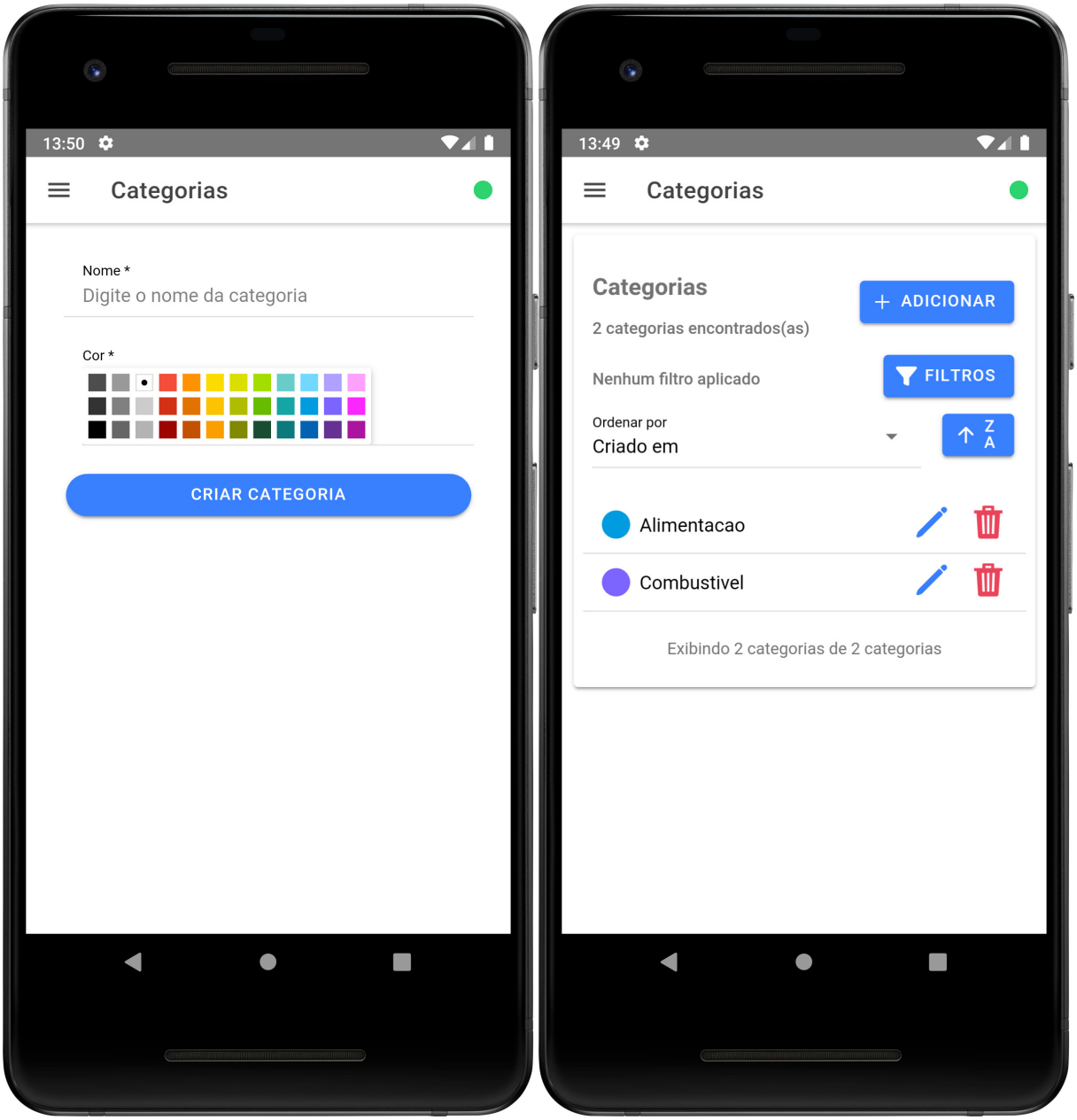


Fonte: Autoria própria.

4.3.2.3 Telas de categorias

Para criação e gerência das categorias, foram criadas duas telas, seguindo o padrão já apresentado, uma tela para cadastro e edição e uma para gestão. A Figura 18 exibe como essas telas foram implementadas no aplicativo.

Figura 18 – Tela de cadastro de categoria à esquerda e tela de gestão das categorias à direita.



Fonte: Autoria própria.

4.3.2.4 Telas de Indicadores

Nas Figuras 19, 20 e 21 é possível observar as telas de indicadores presentes no aplicativo. Visando fornecer liberdade ao usuário para definir o período o qual será analisado em ambas as telas foram adicionados filtros utilizando a data de finalização dos fretes. Ademais também foi adicionada duas formas de apresentação por lista ou gráfico, indicadas graficamente com ícones, utilizados em todo o aplicativo, pretendendo se adequar à heurística de "Consistência e padrões".

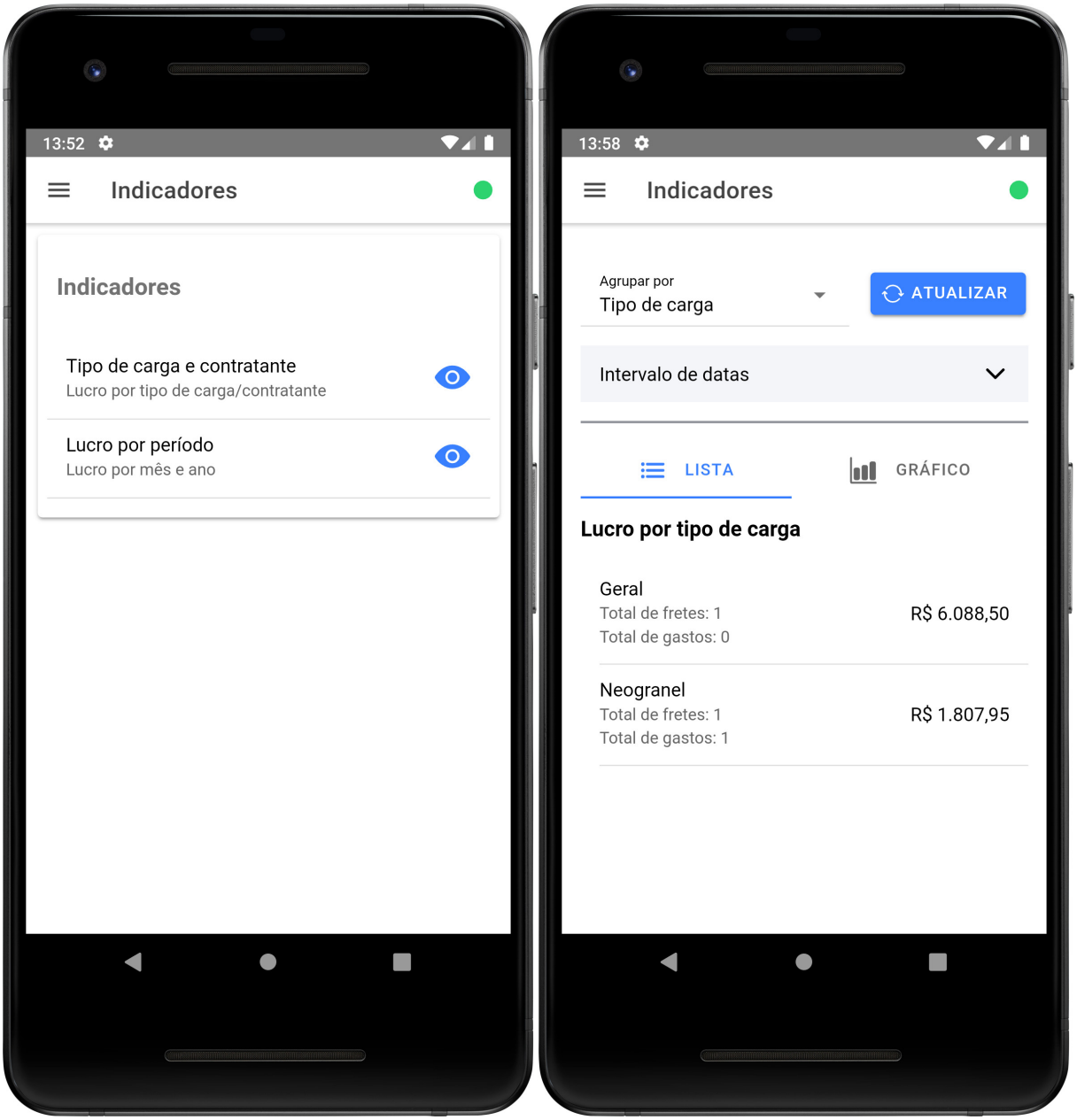
Figura 19 – Tela de indicadores por receita, despesa e lucro acima e tela por lucro com o passar do tempo, abaixo.

Fonte: Autoria própria.

4.3.2.4.1 Lucro por tipo de carga, contratante e trajeto

Na esquerda da Figura 20 é exibido a lista com todos os indicadores fornecidos, e na direita é apresentado a tela de lucro por tipo de carga, que exibe para cada tipo de carga dos fretes o número total de fretes, número total de gastos e o somatório do lucro para este tipo.

Figura 20 – Tela de indicadores à esquerda e tela de indicador de lucro por tipo de carga à direita.

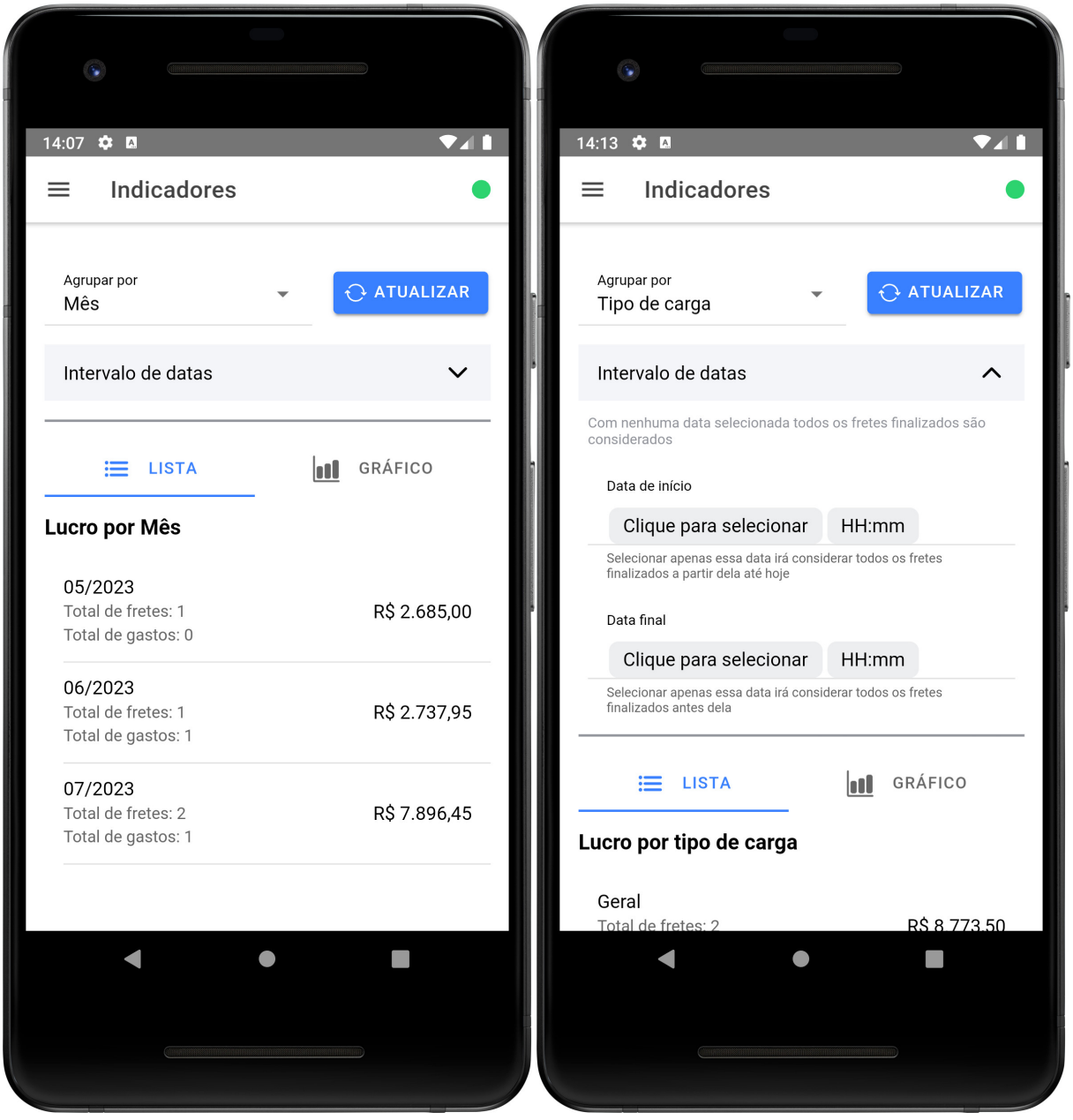


Fonte: Autoria própria.

4.3.2.4.2 Lucro por mês e ano

Na esquerda da Figura 21 é apresentado a tela de lucro para cada mês e ano, que exibe para cada mês e ano dos fretes cadastrados o número total de fretes, número total de gastos e o somatório do lucro para este período.

Figura 21 – Tela de indicador de lucro por mês à esquerda e tela de indicador de lucro por mês com filtros.



Fonte: Autoria própria.

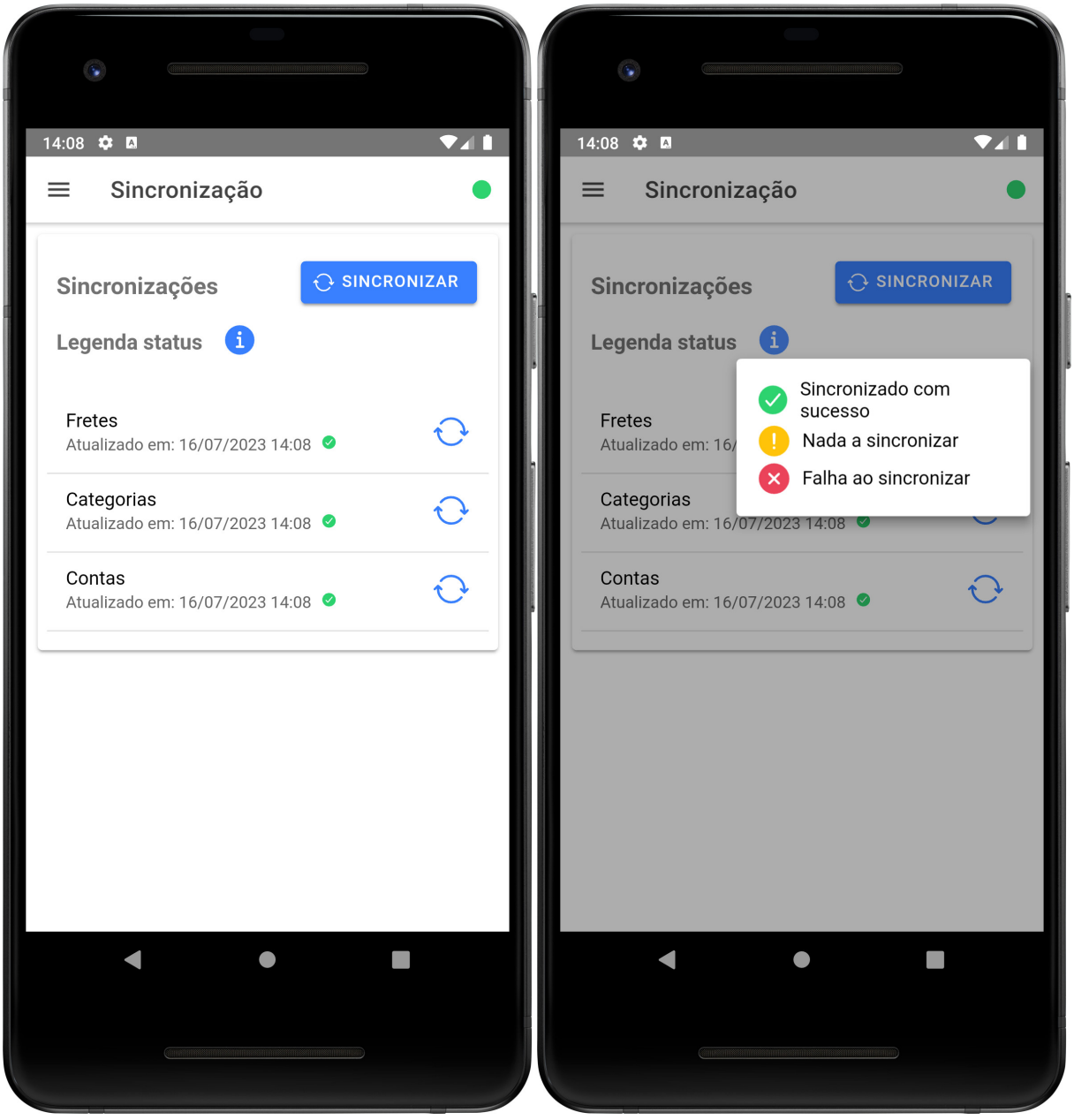
4.3.2.5 Tela de sincronização

Para indicar o *status* da última sincronização dos dados cadastrados localmente, foi criada uma tela que exibe para cada entidade o *status*, horário da sincronização e

permite ações como sincronizar todos os registros ou apenas os registros de uma entidade, esta tela pode ser encontrada na Figura 22.

Esta interface foi projetada para atender a heurística de "*Status* do sistema". Inicialmente esta tela não possuía uma legenda indicando o ícone e respectivo *status* da sincronização, avaliando a interface utilizando a heurística de "Reconhecimento ao invés de relembrança", a legenda foi adicionada à tela. A legenda pode ser observada na Figura 22.

Figura 22 – Tela de sincronização à esquerda e legenda de *status* à direita.



Fonte: Autoria própria.

4.4 Verificação de requisitos

Para verificar os requisitos elicitados na Seção 3.4 foi utilizada a técnica de inspeção, sendo ela um processo de revisão formal de software e corresponde a uma atividade importante da garantia de qualidade do software. É um dos métodos mais eficientes e efetivos na busca por um produto de melhor qualidade (WIEGERS; BEATTY, 2013).

4.4.1 Requisitos Funcionais

Os requisitos funcionais desenvolvidos foram priorizados de acordo com a técnica de priorização MoSCoW, abordada na Seção 3.4.2.1. Para este trabalho, os requisitos classificados como *Must* foram desenvolvidos, com pequenas alterações, como previsto pelo XP.

Para que fossem considerados finalizados, todos os requisitos desenvolvidos necessitaram de revisão do código e implementação de testes automatizados, como abordado na Seção 3.4.3.3. Portanto, esses foram os requisitos finalizados:

- **RF01** - Gestão de cadastro de usuários;
- **RF02** - Fornecer categorias de gastos e receitas;
- **RF04** - Gestão de fretes;
- **RF05** - Gestão de gastos de um frete;
- **RF06** - Fornecer indicadores de fretes em um período;
- **RF07** - Fornecer a possibilidade de gerir gastos e receitas para fretes já finalizados;
- **RF09** - Filtrar fretes por características;
- **RF11** - O sistema deve atualizar os dados armazenados remotamente com os dados cadastrados de forma *offline* pelo usuário.

4.4.2 Requisitos de Qualidade

Para verificar se os requisitos de qualidade foram satisfeitos, foi utilizado uma lista de verificação com os critérios dos requisitos que devem ser verificados em todas as partes do software, podendo esta lista ser observada a seguir.

- **Usabilidade:**
  - **RNF01** - o sistema possui um conjunto de componentes próprios utilizados com frequência? Sim, existem componentes com o *status* de conexão, nota de

erro de formulário, botão de data e hora, componente de gestão de registros, componente de passos;

- **RNF02** - o sistema utiliza um conjunto de ícones e *assets* padronizados, sempre com o mesmo propósito? Sim, como apresentado na Seção 4.3 as telas são criadas utilizando os componentes e ícones do *framework* Ionic;
- **RNF03** - o sistema informa erros nos campos do formulário? Sim, em todos os formulários os campos são validados e mensagens de erros são apresentadas em cada campo;
- **RNF04** - o sistema envia requisições apenas quando os campos são válidos? Sim, o sistema apenas envia as requisições após a validação dos campos dos formulários;
- **RNF05** - o sistema possibilita o uso de alguns recursos do sistema de forma *offline*? Sim, as funcionalidades mais relevantes para o contexto de um caminhoneiro estão disponíveis mesmo sem conexão com a Internet.

- **Integridade:**
  - **RNF06** - o sistema utiliza uma forma de armazenamento persistente, que garante transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade)? Sim, tanto no *back-end* quanto no *front-end*, que utilizam o PostgreSQL e o SQLite, respectivamente.
- **Modificabilidade:**
  - **RNF07** - os repositórios do sistema estão configurados com ferramentas de análise estática de código? Sim, conforme apresentado na Seção 4.2 ambos os repositórios estão configurados para executar a análise estática automaticamente.
- **Portabilidade:**
  - **RNF08** - o sistema funciona para os sistemas operacionais Android e iOS? Foi verificado que o aplicativo é compatível com o sistema operacional Android. No entanto, visto que nenhum integrante do grupo de desenvolvimento possuía acesso a um dispositivo iOS ou MacOS, além da ausência mínima dos registros necessários para disponibilização da aplicação neste sistema operacional e o prazo limitado para elaboração desse trabalho, não foi possível averiguar o funcionamento nessa plataforma. Apesar disso, Ionic (2011) afirma que apenas com uma base de código, é possível executar o aplicativo em ambas as plataformas utilizando o *framework*.
- **Segurança:**

  - **RNF09** - o sistema criptografa senhas e outros dados sensíveis dos usuários? Sim, nenhum dado sensível do usuário é salvo sem que seja criptografado.
  - **RNF10** - o sistema solicita a senha atual do usuário antes de desativar ou excluir a conta? Não se aplica, visto que para a primeira versão do projeto, não existe a opção de exclusão de conta.
  - **RNF11** - o sistema solicita confirmação por email antes de desativar ou excluir a conta? Não se aplica, visto que para a primeira versão do projeto, não existe a opção de exclusão de conta.

- **Eficiência:**
  - **RNF04** - o sistema valida os campos antes de submeter requisições? Todos os formulários do *front-end* são devidamente validados e mensagens de erro amigáveis são mostradas ao usuário em caso de falha ao submeter a requisição.
  - **RNF12** - as colunas do banco de dados utilizadas com frequência possuem índices? Foram criados índices nas colunas que tiram bom proveito desse recurso, como atributos identificadores, chaves candidatas (*unique*) ou que são usados com frequência em consultas ao banco de dados da aplicação.

- **Confiabilidade:**
  - **RNF13** - o código-fonte do sistema possui testes automatizados, com uma cobertura lógica de pelo menos 60%? Tanto o *front-end* quanto o *back-end* possuem cobertura lógica de testes acima dos 60%.
  - **RNF14** - O sistema executa testes de forma automática integrada ao ciclo de desenvolvimento? O projeto conta com uma suíte de testes automatizada utilizando de recursos do GitHub. Integrações para o código-fonte são bloqueadas até que todos os testes sejam executados sem falhas.

5 Considerações Finais

Este capítulo tem como objetivo apresentar os resultados obtidos, analisar os objetivos alcançados, identificar limitações e desafios enfrentados durante a execução do projeto e possíveis trabalhos futuros que podem ser elaborados.

5.1 Conclusões

Conforme descrito na Seção 1.4.1, o objetivo deste trabalho foi desenvolver uma aplicação móvel que forneça aos caminhoneiros do Brasil uma forma simples para a gestão financeira das viagens, gastos e receitas.

A partir da pesquisa bibliográfica foi possível conhecer mais sobre as características dos caminhoneiros no Brasil, formas de implementação de um aplicativo móvel, finanças pessoais e de pequenas empresas. Analisando as informações relatadas no referencial teórico desse trabalho, uma metodologia de pesquisa e outra de desenvolvimento foi criada, delimitando o escopo do software e as necessidades dos usuários, além das tecnologias que seriam utilizadas durante o desenvolvimento. A partir desse contexto, a aplicação foi desenvolvida com o intuito de satisfazer os objetivos definidos para o trabalho, em sintonia com sua questão de pesquisa.

Ao que tange o desenvolvimento do aplicativo, o objetivo foi implementar todos os requisitos funcionais elicitados que estivessem classificados como *Must*, seguindo a técnica de priorização MoSCoW descritos na Seção 3.4.3.1. Além disso, os requisitos de qualidade deveriam ser cumpridos para que as expectativas dos usuários fossem devidamente atingidas.

O projeto de uma interface intuitiva, porém limitada ao tamanho de tela do dispositivo móvel, foi um dos principais desafios do projeto da interface, mas o processo de *design* utilizado contribuiu para um nível de acesso mais adequado ao perfil de usuário de um caminhoneiro.

Com o desenvolvimento das funcionalidades definidas foi possível prover aos caminhoneiros uma forma de controle das viagens, gastos e receitas, por meio de um dispositivo móvel, possibilitando o uso do aplicativo de forma *offline*, adequado à rotina dos caminhoneiros. Além disso, os indicadores fornecidos podem ajudar os profissionais a analisarem suas finanças, contribuindo para uma possível melhoria em sua saúde financeira.

A aplicação desenvolvida contempla um conjunto de práticas e técnicas para garantir uma solução eficaz para a gestão financeira dos caminhoneiros. A abordagem *DevOps*, em conjunto com o XP, promove a colaboração, automação, entrega e integração contínua,

resultando em um processo mais eficiente e em uma aplicação de maior qualidade. Combinando metodologias ágeis de desenvolvimento, padrões arquiteturais bem estabelecidos e estratégias de bancos de dados adequadas, a aplicação atende os objetivos estipulados para este trabalho.

Dessa forma, com os objetivos concluídos, restam possíveis trabalhos futuros que serão abordados na Seção 5.2. O instalador do aplicativo pode ser encontrado no endereço virtual https://github.com/tcc-fga-igor-paiva-thiago-lopes/docs

5.2 Trabalhos Futuros

Apesar do desenvolvimento dos requisitos considerados essenciais para o sucesso do trabalho, vários requisitos elicitados listados na Seção 3.4.3.1 que não foram desenvolvidos podem agregar muito valor ao usuário, como a gestão de gastos e receitas fixas e variáveis gerais, estabelecimento de um teto de gastos por frete, distância aproximada de um frete, marcar o trajeto de um frete baseado no ponto de origem e destino, fornecer mais indicadores, entre outros.

Outro fator importante para ser explorado seria a verificação da compatibilidade com a plataforma iOS, visto que durante a execução deste trabalho não foi possível se certificar do funcionamento para esse outro sistema operacional.

A execução de um teste de usabilidade com possíveis usuários reais do sistema seria de grande valia para a validação da interface e dos requisitos desenvolvidos durante o trabalho. Com ele, seria possível indicar possíveis pontos de melhoria na experiência do usuário, além da identificação de possíveis novas funcionalidades que agreguem valor para os caminhoneiros.

Uma documentação formalizada do aplicativo, que forneça um ambiente interativo e virtual de suporte aos seus usuários, além da possibilidade do envio de *feedback* e sugestões pode ser ainda explorada com o intuito de fazer com que estes importantes profissionais do transporte nacional possam contribuir relatando possíveis *bugs* e sugestões de melhorias e/ou evoluções, além de poderem tirar dúvidas com relação ao uso do aplicativo, contendo também dicas e sugestões que colaborem com a saúde financeira de seus usuários.

Referências

ALESSI, A.; ALVES, M. K. Hábitos de vida e condições de saúde dos caminhoneiros do brasil: uma revisão da literatura. *Revistas Eletrônicas PUCRS*, Rio Grande do Sul Brasil, v. 8, n. 3, p. 129–136, set.-dez 2015. Disponível em: <http://dx.doi.org/10.15448/1983-652X.2015.3.18184>. Acesso em: 8 jan. 2023. Citado na página 33.

ANDRADE, A. W.; AGRA, R.; MALHEIROS, V. Estudos de caso de aplicativos móveis no governo brasileiro. Brasilia, DF, Brasil, 2013. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/5740/5637> Acesso em: 8 jan. 2023. Citado na página 28.

ANDROID STUDIO. *Android Studio*. 2022. Disponível em: <https://developer.android.com/studio/features>. Acesso em: 09 fev. 2023. Citado na página 69.

ANTT. *Relatório Anual 2015*. 2015. Disponível em: <http://anuario.antt.gov.br/index.php/content/view/4880/Relatorios_Anuais.html>. Acesso em: 14 jan. 2023. Citado 2 vezes nas páginas 38 e 39.

ANTT. *Documento Eletrônico de Transporte: Webinário "Avanços e Entraves na Multimodalidade no Brasil"*. 2021. Disponível em: <https://portal.antt.gov.br/documents/359159/0/Webina%CC%81rio+Avanc%CC%A7os+e+Entraves+da+Multimodalidade+no+Brasil-Documento+Eletro%CC%82nico+de+Transporte+-+DT-e+-GABRIEL+VALDERRAMA.pdf/77a02752-f3d1-1929-b118-5bfe76a85199?t=1638899752287>. Acesso em: 16 jan. 2023. Citado na página 38.

ARAúJO, M. d. P. S.; BANDEIRA, R. A. d. M.; CAMPOS, V. B. G. Custos e fretes praticados no transporte rodoviário de cargas: uma análise comparativa entre autônomos e empresas. *Journal of Transport Literature*, Rio Grande do Sul Brasil, v. 8, n. 4, p. 187–226, oct. 2014. ISSN 2238-1031. Disponível em: <https://doi.org/10.1590/2238-1031.jtl.v8n4a8>. Acesso em: 14 jan. 2023. Citado na página 36.

AWS. *O que é o DevOps?* 2023. Disponível em: <https://aws.amazon.com/pt/devops/what-is-devops/>. Acesso em: 20 jan. 2023. Citado na página 49.

BARBOSA, S. D. J. et al. *Interacão Humano-Computador e Experiência do Usuário*. [S.l.]: Autopublicacão, 2021. Citado 5 vezes nas páginas 42, 43, 50, 51 e 79.

BECK, K.; ANDRES, C. *Extreme Programming Explained: embrace change*. USA: Addison Wesley Professional, 2004. Citado 3 vezes nas páginas 25, 47 e 48.

BRASIL. Lei nº 11.442, de 5 de janeiro de 2007. *Diário Oficial da República Federativa do Brasil*, Brasília, DF, 2007. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2007-2010/2007/lei/l11442.htm>. Acesso em: 13 jan. 2023. Citado 3 vezes nas páginas 34, 35 e 41.

BUENO, L. A educação financeira e o processo de desenvolvimento econômico do país. São Paulo Brasil, 2010. Disponível em: <https://www.monografias.

com/pt/trabalhos-pdf/educacao-financeira-processo-desenvolvimento-economico/educacao-financeira-processo-desenvolvimento-economico.pdf> Acesso em: 4 dez. 2022. Citado na página 21.

BUETTNER, K.; SIMMONS, A. M. Mobile web and native apps: How one team found a happy medium. 2011. Disponível em: <https://link.springer.com/content/pdf/10.1007/978-3-642-21675-6_63.pdf>, Acesso em: 15 jan. 2023. Citado 3 vezes nas páginas 27, 28 e 31.

CBO. *Motoristas de veículos de cargas em geral*. 2022. Disponível em: <http://www.mtecbo.gov.br/cbosite>. Acesso em: 04 dez. 2022. Citado na página 33.

CNT. *Conheça o perfil dos caminhoneiros do Brasil*. 2019. Disponível em: <https://www.cnt.org.br/agencia-cnt/pesquisa-cnt-perfil-caminhoneiros-brasil-2019>. Acesso em: 04 dez. 2022. Citado 2 vezes nas páginas 22 e 38.

CODECOV. *Code Coverage: More Than a Metric*. 2023. Disponível em: <https://about.codecov.io/>. Acesso em: 01 jul. 2023. Citado na página 68.

COOPER, A. et al. *About Face: The Essentials of Interaction Design*. Indianapolis, IN, USA: Wiley, 2014. Citado na página 43.

CâNDIDO, C. H. *Projeto brModelo 3.0 (atual v3.31)*. 2020. Disponível em: <http://www.sis4.com/brModelo/>. Acesso em: 09 fev. 2023. Citado na página 68.

DEHLINGER, J.; DIXON, J. Mobile application software engineering: Challenges and research directions. United States, 2011. Citado na página 28.

EISENMAN, B. *Learning React Native*. CA, Estados Unidos: Meg Foley, 2015. Citado na página 32.

EXAME. *Brasil é o 74º em ranking global de educação financeira*. 2015. Disponível em: <https://exame.com/invest/minhas-financas/brasil-e-o-74o-em-ranking-global-de-educacao-financeira/>. Acesso em: 4 dez. 2022. Citado na página 21.

FOWLER, M. *Patterns of Enterprise Application Architecture*. USA: Addison-Wesley, 2003. Citado 2 vezes nas páginas 63 e 64.

FOWLER, M. *IntegrationTest*. 2018. Disponível em: <https://martinfowler.com/bliki/IntegrationTest.html>. Acesso em: 01 jul. 2023. Citado na página 78.

FOWLER, M. *Agile Software Guide*. 2019. Disponível em: <https://www.martinfowler.com/agile.html>. Acesso em: 20 jan. 2023. Citado na página 47.

GAMMA, E. et al. *Padrões de Projeto*. Porto Alegre: Bookman, 2007. Citado na página 31.

GAZZONI, E. I. *Fluxo de caixa: ferramenta de controle financeiro para a pequena empresa*. Dissertação (Mestrado) — Engenharia de Produção. Universidade Federal de Santa Catarina, Santa Catarina, Brasil, 2003. Disponível em: <https://repositorio.ufsc.br/handle/123456789/85831>. Acesso em: 14 jan. 2023. Citado na página 41.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. Rio Grande do Sul: UFRGS Editora, 2009. Citado 3 vezes nas páginas 24, 45 e 46.

GIT. *[Página inicial do Git]*. 2023. Disponível em: <https://git-scm.com/>. Acesso em: 25 jan. 2023. Citado na página 68.

GITHUB. *Quickstart*. 2023. Disponível em: <https://docs.github.com/en/get-started/quickstart>. Acesso em: 25 jan. 2023. Citado na página 68.

GOOGLE PLAY. *InfraBR*. 2019. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.infrabr>. Acesso em: 16 jan. 2023. Citado na página 38.

GOV.BR. *Documento Eletrônico de Transporte - DT-e*. 2021. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/dt-e>. Acesso em: 16 jan. 2023. Citado na página 36.

GOV.BR. *Conheça os Conceitos e a Plataforma*. 2022. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/dt-e/tire-suas-duvidas-sobre-o-dt-e-1>. Acesso em: 16 jan. 2023. Citado na página 36.

GOV.BR. *InfraBR*. 2022. Disponível em: <https://www.gov.br/pt-br/apps/infrabr>. Acesso em: 16 jan. 2023. Citado na página 38.

GOV.BR. *Tire Suas Dúvidas Sobre o DT-e*. 2022. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/dt-e/tire-suas-duvidas-sobre-o-dt-e2>. Acesso em: 16 jan. 2023. Citado 2 vezes nas páginas 36 e 37.

GOV.BR. *Acessibilidade Digital*. 2023. Disponível em: <https://www.gov.br/governodigital/pt-br/acessibilidade-digital>. Acesso em: 03 mar. 2023. Citado na página 42.

HEROKU. *[Página inicial do Heroku]*. 2023. Disponível em: <https://www.heroku.com/home>. Acesso em: 11 jan. 2023. Citado na página 69.

HUYNH, M.; GHIMIRE, P.; TRUONG, D. Hybrid app approach: Could it mark the end of native app domination? 2017. Disponível em: <http://iisit.org/Vol14/IISITv14p049-065Huynh3472.pdf> Acesso em: 12 jan. 2023. Citado 5 vezes nas páginas 28, 29, 30, 31 e 32.

IBGE. *Logística dos Transportes no Brasil*. 2014. Disponível em: <https://www.ibge.gov.br/geociencias/organizacao-do-territorio/redes-e-fluxos-geograficos/15793-logistica-dos-transportes.html?=&t=acesso-ao-produto>. Acesso em: 04 dez. 2022. Citado na página 21.

IBGE. *Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal 2021 / IBGE, Coordenação de Pesquisas por Amostra de Domicílios*. 2022. Disponível em: <https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101963>. Acesso em: 04 dez. 2022. Citado 2 vezes nas páginas 21 e 22.

IONIC. *Introduction to Ionic*. 2011. Disponível em: <https://ionicframework.com/docs/> Acesso em: 16 jan. 2023. Citado 2 vezes nas páginas 32 e 93.

IPEA. Carga horária de trabalho: evolução e principais mudanças no brasil. 2009. Disponível em: <https://repositorio.ipea.gov.br/handle/11058/5304> Acesso em: 8 jan. 2023. Citado na página 33.

JOBE, W. Native apps vs. mobile web apps. 2013. Disponível em: <https://www.researchgate.net/profile/William-Jobe/publication/268153001_Native_Apps_Vs_Mobile_Web_Apps/links/546346b30cf2cb7e9da765c3/Native-Apps-Vs-Mobile-Web-Apps.pdf>, Acesso em: jan. 2023. Citado 3 vezes nas páginas 29, 30 e 31.

KIM, G. et al. *The DevOps Handbook*. Portland, OR, USA: IT Revolution Press, 2021. Citado 3 vezes nas páginas 25, 47 e 49.

LARA, S. M. A. *Mecanismos de apoio para usabilidade e acessibilidade na interação de adultos mais velhos na Web*. Tese (Doutorado) — Ciências de Computação e Matemática Computacional, Universidade de São Paulo, São Paulo, Brasil, dez 2012. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55134/tde-14022013-163940/publico/TeseRevisadaSilvana.pdf>. Acesso em: 06 dez. 2022. Citado 2 vezes nas páginas 22 e 49.

LIU, C. Z.; AU, Y. A.; CHOI, H. S. An empirical study of the freemium strategy for mobile apps: Evidence from the google play market. 2012. Disponível em: <https://web.archive.org/web/20170920045336id_/http://aisel.aisnet.org:80/cgi/viewcontent.cgi?article=1050&context=icis2012> Acesso em: 9 jan. 2023. Citado na página 28.

LOURENçO, G. Os efeitos da greve dos caminhoneiros. *Vitrine da Conjuntura*, Curitiba Brasil, v. 11, n. 6, p. 1–2, ago 2018. Disponível em: <https://img.fae.edu/galeria/getImage/351/13224112528955730.pdf>. Acesso em: 04 dez. 2022. Citado na página 21.

MACIEL, C. et al. *Avaliação Heurística de Sítios na Web*. 2004. Disponível em: <https://www.researchgate.net/profile/Cristiano-Maciel-3/publication/271272684_Avaliacao_Heuristica_de_Sitios_na_Web/links/54c419860cf2911c7a4dab08/Avaliacao-Heuristica-de-Sitios-na-Web.pdf>. Acesso em: 15 jan. 2023. Citado na página 79.

MACMAGAZINE. *Usuários gastaram US$33,6 bilhões com apps no 3º trimestre*. 2021. Disponível em: <https://macmagazine.com.br/post/2021/10/01/usuarios-gastaram-us336-bilhoes-com-apps-no-3o-trimestre>. Acesso em: 09 jan. 2023. Citado na página 28.

MCKINNON, T. *[Página inicial do Lucidchart]*. 2023. Disponível em: <https://www.lucidchart.com/pages/pt>. Acesso em: 25 jan. 2023. Citado na página 69.

MORAES, E. A. d. *Utilização de uma estratégia para identificação de fontes de informação na fase de elicitação*. Dissertação (Mestrado) — Informática. Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2010. Disponível em: <https://doi.org/10.17771/PUCRio.acad.15760>. Acesso em: 26 jan. 2022. Citado na página 51.

MORESI, E. Metodologia de pesquisa. 2003. Disponível em: <http://www.inf.ufes.br/~pdcosta/ensino/2010-2-metodologia-de-pesquisa/MetodologiaPesquisa-Moresi2003.pdf> Acesso em: 24 jan. 2023. Citado na página 46.

MYERS, G.; BADGETT, T.; SANDLER, C. *The Art of Sofware Testing*. USA: John Wiley & Sons, 2012. Citado na página 78.

NIELSEN, J.; NORMAN, D. *The Definition of User Experience (UX)*. 2023. Disponível em: <https://www.nngroup.com/articles/definition-user-experience/>. Acesso em: 11 jan. 2023. Citado na página 42.

OLIVEIRA, S. A educaÇÃo financeira em gestÃo de pessoas. Rio Grande do Sul Brasil, 2022. Disponível em: <http://repositorio.laboro.edu.br:8080/xmlui/bitstream/handle/123456789/332/Educa%c3%a7%c3%a3o%20financeira%20em%20gest%c3%a3o%20de%20pessoas.pdf?sequence=1&isAllowed=y>. Acesso em: 4 dez. 2022. Citado na página 21.

POSTGRESQL. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. 2023. Disponível em: <https://www.postgresql.org/>. Acesso em: 22 jan. 2023. Citado na página 64.

ROMA, P.; RAGAGLIA, D. Revenue models, in-app purchase, and the app performance: Evidence from apple's app store and google play. Itália, 2016. Disponível em: <https://core.ac.uk/download/pdf/53304885.pdf> Acesso em: 10 jan. 2023. Citado na página 27.

SANTOS, J. O. d. *Finanças pessoais para todas as idades: um guia prático*. São Paulo: Atlas, 2014. Citado 3 vezes nas páginas 39, 40 e 41.

SANVICENTE, A. Z.; SANTOS, C. d. C. e. *Orçamento na Administração de Empresas*. São Paulo: Atlas, 2000. Citado na página 41.

SERZEDELLO, T.; TOMAéL, M. Produção tecnológica da universidade estadual de londrina: mapeamento da área de ciências agrárias pela plataforma lattes. *AtoZ*, Curitiba, v. 1, n. 1, p. 23–37, jan 2011. Disponível em: <https://revistas.ufpr.br/atoz/article/view/41281/25200>. Acesso em: 07 dez. 2022. Citado 2 vezes nas páginas 24 e 46.

SHARP, H.; ROGERS, Y.; PREECE, J. *Interaction Design: beyond human-computer interaction*. Indianapolis, IN, USA: John Wiley & Sons, 2019. Citado na página 42.

SILVA, R. A. *Vida de Caminhoneiro: sofrimento e paixão*. Dissertação (Mestrado) — Psicologia. Pontifícia Universidade Católica de Campinas, São Paulo, Brasil, 2015. Disponível em: <https://repositorio.sis.puc-campinas.edu.br/bitstream/handle/123456789/15582/ccv_ppgpsico_me_Ramon_AS.pdf>. Acesso em: 8 jan. 2023. Citado 5 vezes nas páginas 33, 34, 35, 36 e 38.

SOMMERVILLER, I.; SAWYER, P. *REQUIREMENTS ENGINEERING*. EN: Microsoft Press, 1997. Citado na página 50.

SQLITE. *SQLite is Transactional*. 2021. Disponível em: <https://www.sqlite.org/transactional.html>. Acesso em: 22 jan. 2023. Citado na página 63.

STATCOUNTER. *Mobile Operating System Market Share Worldwide*. 2022. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide>. Acesso em: 04 dez. 2022. Citado na página 27.

VISUAL STUDIO CODE. *Getting Started*. 2023. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 25 jan. 2023. Citado na página 68.

WANG, H.-Y.; LIAO, C.; YANG, L.-H. What affects mobile application use? the roles of consumption values. Canada, 2013. Disponível em: <https://pdfs.semanticscholar.org/213f/bf5121b1872c0d98565890c63d63851d3d85.pdf> Acesso em: 8 jan. 2023. Citado na página 27.

WIEGERS, K.; BEATTY, J. *Software Requirements*. USA: Microsoft Press, 2013. Citado 12 vezes nas páginas 50, 51, 52, 53, 54, 56, 57, 58, 59, 60, 62 e 92.

Apêndices

APÊNDICE A – Personas

A.1 Persona 1

Figura A-1 – Foto ilustrativa de Antônio Pereira da Silva[1].

Fonte: Retirado de <https://thispersondoesnotexist.com/>. Acesso em: 26 jan. 2023

Antônio Pereira da Silva, de 50 anos, é transportador autônomo de cargas independente, possui 30 anos de experiência na área e possui uma bitrem.

- **Status:** Persona Primária
- **Objetivos:**
  - Se sentir no controle de todos os gastos de suas viagens;
  - Se aposentar com 70 anos;
  - Juntar dinheiro para comprar uma casa;
  - Saber o quanto gastou com combustível no mês;
  - Viajar a passeio com a família uma vez ao ano;
  - Aumentar o lucro em suas viagens;
- **Habilidades:**
  - Bom de negócio;
  - Comunicativo;
  - Dirigir veículos pesados;

[1] Imagem meramente ilustrativa gerada por inteligência artificial.

  - Possui ensino fundamental completo;

- **Tarefas:**
  - Realiza em média duas viagens por semana;
  - Leva o bitrem para o lava-jato a cada mês durante o fim de semana;
  - Leva o bitrem para revisão uma vez por ano, deixando seu veículo parado por 1 semana;
  - Escolhe a carga mais rentável;
  - Prioriza destinos mais curtos;

- **Relacionamentos:**
  - Seus familiares;
  - Colegas de profissão;
  - Mecânicos;
  - Empresas de transporte de cargas;
  - Fabricantes e comerciantes que necessitam de transportadores;

- **Expectativas:**
  - Receber um resumo de viagens no final do mês;
  - Poder criar categorias para seus gastos;
  - Poder registrar gastos sem acesso à Internet;
  - Receber um relatório ao final de uma viagem;
  - Ser avisado em caso de aumento com gasto de combustível com o passar do tempo;

A.2 Persona 2

Figura A-2 – Foto ilustrativa de Patrick Barbosa de Araújo[1].

Fonte: Retirado de <https://thispersondoesnotexist.com/>. Acesso em: 26 jan. 2023

Patrick Barbosa de Araújo, de 40 anos, é empregado da empresa Jefferson Caminhões, possui 15 anos de experiência na área e não tem caminhão próprio.

- **Status:** Persona Primária
- **Objetivos:**
  - Comprar um caminhão em até 10 anos;
  - Economizar em suas despesas durante viagens;
  - Juntar dinheiro para construir um apartamento para alugar;
  - Registrar rapidamente suas despesas;
  - Saber quantas viagens e quanto recebeu de comissão durante o ano
- **Habilidades:**
  - Comprometido;
  - Focado;
  - Dirigir veículos pesados;
  - Possui ensino médio incompleto;
- **Tarefas:**
  - Espera o descarregamento da carga;
  - Realiza quantas viagens conseguir na semana;

[1] Imagem meramente ilustrativa gerada por inteligência artificial.

  - Leva o caminhão para ajustes pontuais na oficina;
  - Comunicar a empresa quando precisa fazer algum ajuste de rota;

- **Relacionamentos:**
  - Seus familiares;
  - Colegas de profissão;
  - Funcionários da Jefferson Caminhões;

- **Expectativas:**
  - Receber um relatório mensal de despesas;
  - Poder registrar gastos e receitas genéricos, como contas de água, luz, telefone, etc.;
  - Estabelecer um teto de gastos por viagem;
  - Receber um relatório ao final de cada viagem;

A.3 *Antipersona*

Figura A-3 – Foto ilustrativa de Jefferson Peixoto Sousa[1].

Fonte: Retirado de <https://thispersondoesnotexist.com/>. Acesso em: 26 jan. 2023

Jefferson Peixoto Sousa, de 38 anos, é fundador e sócio majoritário da empresa Jefferson Caminhões, uma empresa de transporte de cargas.

- **Status:** *Antipersona*
- **Objetivos:**
  - Trocar de carro todo ano;
  - Fazer a empresa crescer cada vez mais;
  - Obter maior previsibilidade das entregas de sua empresa;
  - Acompanhar a localização de todos os transportadores de sua empresa;
- **Habilidades:**
  - Persuasivo;
  - Focado;
  - Estressado;
  - Possui ensino superior completo;
- **Tarefas:**
  - Participar das reuniões semanais, mensais e anuais da empresa;
  - Assinar contratos da empresa;
  - Levar a filha no shopping;

[1] Imagem meramente ilustrativa gerada por inteligência artificial.

- **Relacionamentos:**
  - Seus familiares;
  - Conselho da empresa;
  - Funcionários da Jefferson Caminhões;
- **Expectativas:**
  - Acompanhar a localização dos transportadores da empresa;
  - Receber relatórios de desempenho da empresa;
  - Obter mais dados dos transportares da empresa;

APÊNDICE B – Simulação da Arquitetura

A fim de validar, de forma inicial, a arquitetura proposta, foi realizada uma simulação do software. Nesta simulação, foram realizadas as conexões entre o componente Aplicativo e o componente API, além do componente API e o banco de dados PostgreSQL. Foi gerado o instalador do aplicativo para o sistema operacional Android, que se conecta com a implantação do componente API e o banco de dados para validar a capacidade de implantação utilizando a arquitetura definida.

A simulação consiste em satisfazer, em parte, os requisitos "RF01 - Gestão de cadastro de usuários", "RNF03 - O sistema deve informar erros em campos de formulários", "RNF04 - O sistema deve submeter requisições apenas se os campos forem válidos e estiverem corretos" e "RNF09 - O sistema deve criptografar senha e outros dados sensíveis dos usuários". As tecnologias utilizadas para o desenvolvimento foram as definidas respeitando a arquitetura proposta.

No desenvolvimento da simulação, foram criados, seguindo os padrões arquiteturais definidos, os seguintes componentes da arquitetura proposta:

- Uma *model* que representa a relação CAMINHONEIRO e todos os seus dados, conforme a modelagem;
- Uma *controller* que trata as ações solicitadas pelo Aplicativo, cria e retorna o JSON com as informações (*view*);
- Duas páginas no *front-end*, criadas como componentes do Vue.js, utilizando o *framework* Ionic. As páginas iterativas elaboradas foram a de cadastro de usuário (caminhoneiro) e a tela inicial do aplicativo (exibida ao abrir o aplicativo);

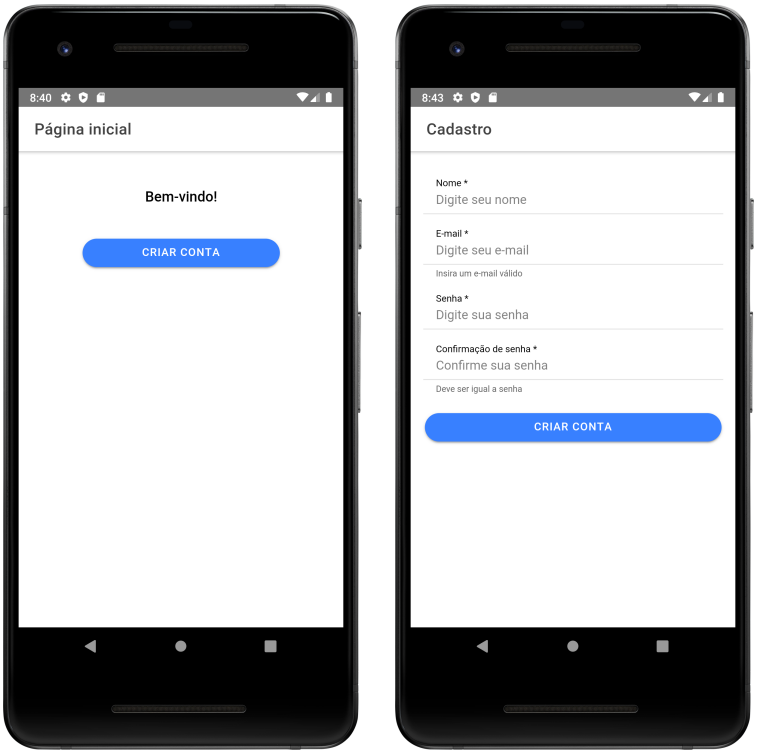
O banco de dados PostgreSQL e a API foram implantados em nuvem, utilizando a ferramenta Heroku. Já o instalador do aplicativo para o sistema operacional Android, foi criado na ferramenta Android Studio.

O repositório com código do *front-end* e *back-end* pode ser encontrado no seguinte endereço virtual:

https://github.com/tcc-fga-igor-paiva-thiago-lopes/prova-conceito

O arquivo instalador para o sistema Android pode ser acessado pela ligação virtual disponível também no GitHub: instalador aplicativo Android. Por motivos de padronização do código consoante à linguagem de programação utilizada, a tabela criada no banco de dados e suas colunas estão identificadas com seus nomes em inglês.

Figura B-4 – Tela página inicial e cadastro da simulação de arquitetura.



Fonte: Autoria própria.

O resultado da simulação pode ser visualizado com uma imagem das telas do aplicativo, conforme a Figura B-4. Com esta simulação foi possível analisar que os principais aspectos da arquitetura proposta para solução coerente de software ao problema apresentado atendem a demanda averiguada. Também foi possível demonstrar que a implantação da API, banco de dados e disponibilização do aplicativo para o sistema Android é possível. Por fim, foram cumpridos, de maneira parcial, os requisitos supracitados, de forma a validar as metodologias apresentadas.