INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DE RONDÔNIA – CAMPUS JI-PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DESENVOLVIMENTO DE APLICATIVO DE COMPARTILHAMENTO DE CARONAS COM REACT NATIVE

MARIA JÚLIA SOUZA DE ALBUQUERQUE LINS

Ji-Paraná – RO
2025

MARIA JÚLIA SOUZA DE ALBUQUERQUE LINS

DESENVOLVIMENTO DE APLICATIVO DE COMPARTILHAMENTO DE CARONAS COM REACT NATIVE

Monografia apresentada ao Instituto Federal de Rondônia - Campus Ji-Paraná, como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Área de Concentração: Ciências Exatas e da Terra.

Orientador (a): Prof. Me. Reinaldo Lima Pereira

Ji-Paraná – RO

2025

FICHA CATALOGRÁFICA

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Lins, Maria Júlia Souza de Albuquerque.
Carona Solidária - Sistema de transporte colaborativo entre os alunos do IFRO / Maria Júlia Souza de Albuquerque Lins. - Ji-Paraná, 2025.
93 f.

Orientador(a): Prof. Reinaldo Lima Pereira.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ji-Paraná, 2025.

1. Aplicativo Mobile. 2. Compartilhamento de Caronas. 3. Mobilidade Urbana. I. Pereira, Reinaldo Lima (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Cleuza Diogo Antunes, CRB-11/864

BANCA EXAMINADORA

(Titulação (Dr./Me./Esp./) e Nome Completo Professor(a) Avaliador(a))
Avaliador nº 1

(Titulação (Dr./Me./Esp./) e Nome Completo Professor(a) Avaliador(a))
Avaliador nº 2

(Titulação (Dr./Me./Esp./) e Nome Completo Professor(a) Avaliador(a))
Presidente da Banca Examinadora

Processo SEI nº: _____

Data da Defesa: 20 de agosto de 2025

Nota Final: _____

AGRADECIMENTOS

À minha família, que não deixou de acreditar em mim e em meu potencial por um dia sequer, sempre me incentivando a alcançar meus objetivos; Aos meus amigos e aos meus colegas de curso, por todos os bons momentos juntos; Aos professores e servidores do IFRO, por seu compromisso ímpar com a educação e com os alunos; E à todos aqueles que permaneceram ao meu lado durante a elaboração deste trabalho, os meus mais sinceros agradecimentos. Seu apoio e motivação foram essenciais para a conclusão deste projeto e serei eternamente grata!

RESUMO

Este estudo propõe o desenvolvimento de um aplicativo mobile de caronas solidárias para a população de Rondônia, fazendo uso de tecnologias como React Native no frontend, MySQL para armazenamento dos dados e Node.js para construção de API, fundamentado nos princípios de economia compartilhada e tecnologia social. A plataforma inova ao permitir que usuários alternem dinamicamente entre as funções de motorista e passageiro no mesmo aplicativo, enquanto integra funcionalidades de rede social para possibilitar conexões entre conhecidos. Assim, a solução oferece uma alternativa mais econômica e viável aos desafios de mobilidade urbana no estado de Rondônia, com possibilidade de expansão para outras regiões de características similares.

Palavras-chave: Aplicativo Mobile; Compartilhamento de Caronas; Mobilidade Urbana.

ABSTRACT

This study proposes the development of a mobile ride-sharing application for the population of Rondônia, Brazil, utilizing a React Native frontend, Node.js API architecture, and MySQL database management. Grounded in the principles of the sharing economy and social technology, the platform innovates by enabling users to dynamically switch between driver and passenger roles within the same application. It integrates social networking features to facilitate connections among acquaintances, creating a trusted community-based system. The solution offers a cost-effective and viable alternative to urban mobility challenges in Rondônia, with potential for expansion to other regions with similar characteristics.

Keywords: Mobile Application; Ride-Sharing; Urban Mobility.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

IFRO – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia

UML – Unified Modeling Language

HTTP – Hypertext Transfer Protocol

REST – Representational State Transfer

ORM – Object-Relational Mapping

I/O – In/Out

TI – Tecnologia da Informação

UX – User Experience

UI – User Interface

IDE – Integrated Development Environment

SUMÁRIO

1. INTRODUÇÃO

A mobilidade urbana no estado de Rondônia representa um desafio estrutural recorrente, que afeta diretamente a qualidade de vida da população. A dependência de transportes individuais, aliada à escassez de alternativas acessíveis e integradas, pode vir a resultar em congestionamentos, aumento de custos com deslocamento ao utilizar alternativas como viagens por aplicativo e dificuldades no acesso a serviços essenciais. Essa realidade sugere uma necessidade de soluções inovadoras que promovam deslocamentos mais eficientes e sustentáveis.

Nesse cenário, a tecnologia emerge como uma aliada estratégica, oferecendo recursos capazes de transformar a maneira como as pessoas se locomovem, especialmente por meio de plataformas digitais colaborativas. A economia compartilhada tem se consolidado como uma tendência global, trazendo consigo práticas como o carpooling, que é um sistema de caronas em que pessoas com trajetos semelhantes compartilham veículos, reduzindo custos e o impacto ambiental.

Com base nesse contexto, este trabalho propõe o desenvolvimento de um aplicativo mobile voltado à promoção de caronas solidárias para o público geral do estado de Rondônia. A proposta visa oferecer uma plataforma segura, escalável e de fácil acesso, que permita aos usuários tanto oferecer quanto solicitar caronas dentro do mesmo ambiente digital. O aplicativo contará ainda com funcionalidades que possibilitam a conexão entre usuários conhecidos, por meio da criação de redes de confiança denominadas comunidades, visando garantir maior segurança nas interações.

Ao explorar os princípios da prática de compartilhamento de viagens e aplicar conceitos modernos de desenvolvimento de software, este projeto busca contribuir de forma prática para a melhoria da mobilidade urbana local. Dessa forma, o projeto tem como objetivo principal o desenvolvimento de uma aplicação que sirva como uma opção válida para auxiliar cidadãos que vivenciam os problemas atuais da mobilidade urbana em Rondônia, apresentando-se como uma alternativa viável, econômica e segura para o deslocamento cotidiano da população.

1.1. Problemática

Como se dá, atualmente, a locomoção dos alunos do IFRO que dependem de transporte público ou de terceiros e que não possuem meio de transporte próprio, como carro ou moto, até o Campus? De que forma esse acesso limitado a meios de transporte afeta a frequência escolar e a pontualidade dos alunos? De que forma a tecnologia pode ser direcionada para criar uma alternativa viável, econômica e sustentável visando o auxílio à mobilidade desses alunos até o Instituto?

1.2. Justificativa

O uso recorrente de aplicativos de corrida como Uber e Urbano Norte muitas vezes pode acarretar custos elevados com base na frequência, o que pode não ser a melhor alternativa para os alunos que não possuem veículo próprio. Nesse cenário, o desenvolvimento de um aplicativo para compartilhamento de caronas solidárias entre os alunos do IFRO surge como uma opção mais econômica, favorável e segura.

1.3. Objetivos

1.3.1. Objetivo Geral

O objetivo geral deste projeto é desenvolver uma aplicação mobile voltada para o gerenciamento e acompanhamento de caronas solidárias compartilhadas.

1.3.2. Objetivos Específicos

I. Incentivar o ato de compartilhar caronas como uma alternativa sustentável.
II. Desenvolver uma API com Node.js.
III. Desenvolver a aplicação mobile utilizando o framework de código aberto React Native.

1.4. Estrutura do Trabalho

Este trabalho está estruturado em cinco seções e um apêndice. A presente seção é a primeira, que traz um contexto introdutório ao tema. A segunda seção é o referencial teórico, na qual são exploradas informações que giram em torno do tema do trabalho e apresentadas as ferramentas utilizadas no desenvolvimento. A terceira seção é destinada à metodologia que foi definida como base. Na quarta seção, são apresentados e discutidos os resultados obtidos. A quinta e última seção traz a conclusão do presente trabalho com algumas considerações finais acerca do desenvolvimento da aplicação. Por fim, o apêndice I é o projeto de desenvolvimento de software, contendo todas as informações pertinentes à análise e ao planejamento da aplicação.

2. REFERENCIAL TEÓRICO

Atualmente, aplicativos de mobilidade urbana são muito populares e presentes no dia a dia das pessoas. Alguns exemplos de necessidades que esses aplicativos atendem são a ausência de veículo próprio, motivos externos como dias chuvosos ou simplesmente situações incomuns e imprevistos nos quais faz-se necessário solicitar uma corrida por meio de aplicativo, como Uber, por exemplo. De acordo com Sarmento, o aplicativo Uber:

> [...] é uma plataforma tecnológica para smartphones lançada nos Estados Unidos em 2010, que permite estabelecer uma conexão entre motoristas profissionais e pessoas interessadas em contratá-los. Por seu intermédio, indivíduos previamente cadastrados no site/aplicativo da Consulente conseguem encontrar, de modo simples e ágil, motoristas parceiros da UBER para transportá-los com conforto e segurança. [...] Esses motoristas são empreendedores individuais, que utilizam a plataforma UBER em sistema de "economia compartilhada" (sharing economy), que otimiza o acesso e contato entre passageiros e condutores. (SARMENTO, 2015, p.1).

Além de empresas de aplicativos que oferecem o serviço de motoristas particulares, como a Uber e a Urbano Norte, existem também soluções voltadas para o compartilhamento de caronas, que surgem como uma ótima alternativa para pessoas que buscam fazer viagens e deslocamentos por um preço reduzido, além de também dividir o custo da viagem para a pessoa que está dirigindo. Um exemplo é o Blablacar, um aplicativo de carona que proporciona a locomoção por um preço menor que os demais aplicativos por meio de viagens comunitárias, distribuindo os custos da viagem entre os motoristas e passageiros.

O chefe do Departamento de Assuntos Econômicos e Sociais (DESA) da ONU, Liu Zhenmin, em seu relatório "Transporte Sustentável, Desenvolvimento Sustentável", diz:

> "O transporte é vital para promover a conectividade, o comércio, o crescimento econômico e o emprego. No entanto, também está implicado como uma fonte significativa de emissões de gases de efeito estufa. Resolver esses trade-offs é essencial para alcançar o transporte sustentável e, por meio dele, o desenvolvimento sustentável" (2021, p. iv-v).

O relatório de Transporte Sustentável afirma ainda que, quando apropriadamente aplicadas, tecnologias novas e emergentes são fundamentais para resolver muitos desafios urgentes (ONU, 2021). Aplicativos como o Blablacar seriam benéficos atuando na redução de veículos nas vias, mas mantendo a quantidade de viagens (PEREIRA *apud* LACERDA, 2023). Ainda segundo Lacerda, o aplicativo Blablacar mostra-se um marco no avanço tecnológico, a fim de solucionar parcialmente os problemas provenientes da sustentabilidade no deslocamento (2023). Nesse sentido, a proposta da aplicação carrega também esse viés ambientalista, uma vez que a prática de compartilhar caronas

reduz a quantidade de automóveis em circulação, de maneira que contribui com a redução da poluição do ar.

Esta questão, que é extremamente prejudicial à saúde e ao meio ambiente, é um grande desafio no Brasil pois as principais fontes de emissão de poluentes nas regiões metropolitanas são móveis, incluindo veículos como carros, motos, ônibus e caminhões (INSTITUTO DE ENERGIA E MEIO AMBIENTE, 2019). Dentre as principais fontes poluidoras nas áreas urbanas, a circulação de veículos que dependem da queima de combustíveis derivados do petróleo é a maior responsável pelos incômodos causados por fuligem, odores desagradáveis, entre outros poluentes lançados na atmosfera (SOUSA, 2024). Segundo o IBGE (2024), a frota de veículos no Brasil no ano de 2024 somava mais de 123 milhões, dos quais 1 milhão foram registrados em Rondônia.

Práticas como o compartilhamento de viagens, também conhecido como *carpooling*, podem ser consideradas alternativas de sustentabilidade viáveis e acessíveis, diminuindo a quantidade de carros nas ruas, reduzindo a emissão de gases poluentes e contribuindo para um ar mais limpo.

2.1. FERRAMENTAS DE DESENVOLVIMENTO

2.1.1. Astah UML

A linguagem visual UML (Linguagem de Modelagem Unificada) define um conjunto de diagramas, utilizados para modelar e documentar sistemas, permitindo uma visão abrangente do projeto e facilitando a detecção de possíveis erros ainda na fase da modelagem, evitando futuros conflitos no desenvolvimento do software.

> Os diagramas UML permitem mapear processos passo a passo e visualizar o fluxo de trabalho dos softwares. UML, que vem do inglês Unified Modeling Language, foi inicialmente usado como uma linguagem de modelagem em engenharia de software, sendo posteriormente adotado para a criação de estruturas de aplicativos, modelagem e documentação de software. Profissionais de vários setores usam o diagrama para modelar processos de negócios e fluxos de trabalho com o auxílio do diagrama UML. (MIRO, 2025, s.p.).

O diagrama UML oferece à organização um método padronizado para o mapeamento de processos, etapa a etapa. Eles permitem que sua equipe visualize facilmente a relação entre sistemas e tarefas (MELO, 2024). Apesar de existirem muitos tipos de diagramas UML, os mais utilizados são os chamados Diagrama de Classe e Diagrama de Caso de Uso. O Astah é uma ferramenta de modelagem e diagramação UML, que possibilita a criação de diagramas que oferecem uma visualização ampla da arquitetura de um software de forma simples.

O Astah conta com diferentes planos de serviços, como o Astah Professional, que é uma ferramenta completa de concepção de software com UML, ERD, fluxograma, DFD e outras funcionalidades; O Astah UML, uma leve ferramenta de diagramação UML e com capacidades de mapeamento mental; O Astah System Safety, que é uma ferramenta MBSE que inclui SysML, GSN, STPA/STAMP, SCDL apropriada para engenheiros de segurança, entre outros (ASTAH, 2025). Neste projeto, a ferramenta a ser utilizada será o Astah UML, pois oferece o suporte adequado para a modelagem de diagrama de Caso de Uso.

2.1.2. Figma

O Figma é uma ferramenta de design e prototipagem versátil que possibilita a criação de protótipos realistas e interativos, a colaboração de forma contínua e a otimização do desenvolvimento de produtos de forma ágil e prática. Ele oferece uma ampla gama de recursos para projetar e prototipar interfaces para web e mobile, o que o torna valioso para profissionais e equipes de design que buscam otimizar seu fluxo de trabalho (FIGMA, 2025). Ele oferece ferramentas avançadas para design de interfaces e a função de criação de fluxos de interações realistas com o design, sem a necessidade de qualquer tipo de código.

A criação de um protótipo com as principais telas e funcionalidades do aplicativo é essencial para o desenvolvimento, evitando possíveis erros, validando ideias, possibilitando testes de diferentes interações e encontrar soluções. Uma vez que a prototipação possua todo o fluxo, estrutura e composição do projeto, ela se torna uma base forte para o desenvolvimento do aplicativo (CUSTÓDIO, 2023). Para facilitar esse trabalho, o Figma disponibiliza uma ferramenta chamada Dev Mode, que possui uma área de inspeção que possibilita ver todos os detalhes de um elemento.

2.1.3. JavaScript e TypeScript

Segundo Flanagan (2013), o JavaScript é uma linguagem de programação de alto nível, orientada a objetos, dinâmica, interpretada e não tipada da Web, atualmente usada pela grande maioria dos sites e interpretada por todos os navegadores modernos. Tendo sido criada inicialmente com o propósito de ser uma linguagem de script, tornou-se uma linguagem de uso geral robusta e eficiente.

Flanagan (2013) introduz conceitos chave a respeito das características da linguagem JavaScript, como, por exemplo, o fato de ser uma linguagem que diferencia letras maiúsculas de minúsculas. Logo, palavras-chave, variáveis, nomes de função e outros identificadores devem ser

digitados conforme a composição compatível. Também menciona a possibilidade da omissão de um separador de instruções, que em muitas linguagens é o símbolo de ponto e vírgula, contanto que as instruções sejam escritas em linhas separadas (FLANAGAN, 2013).

O núcleo da linguagem JavaScript consiste em benefícios comuns da programação, permitindo fazer coisas como (MOZILLA, 2025):

- Armazenar conteúdo útil em variáveis
- Operações com pedaços de texto (strings)
- Executar código em resposta a determinados eventos em uma página da Web

Em resumo, JavaScript é uma linguagem de programação flexível e versátil que adiciona interatividade a uma página web e uma das tecnologias web mais populares, que conta com diversos frameworks que auxiliam a construir aplicações e sites com eficiência e segurança. No entanto, a linguagem é dinamicamente tipada, ou seja, ela não verifica se os dados foram atribuídos de forma consistente e não sabe o tipo de uma variável até que ela seja instanciada.

TypeScript, por sua vez, é uma linguagem de programação estática desenvolvida pela Microsoft que se baseia no JavaScript, oferecendo todos os seus recursos e adicionando um sistema de tipos estáticos, possibilitando a redução de erros comuns antes da execução. Possui o mesmo uso do JavaScript, porém tem o foco na segurança e escalabilidade dos projetos e, enquanto o código JavaScript é interpretado diretamente no navegador, o código TypeScript é compilado para JavaScript antes de rodar.

2.1.4. Sistema de Gerenciamento de Banco de Dados

SILBERSCHATZ (2006, p.4) define o conceito de um sistema de gerenciamento de banco de dados (SGBD) como uma coleção de dados inter-relacionados e um conjunto de programas que permitem aos usuários acessar e modificar esses dados. Para ELMASRI (2011), um SGBD é uma coleção de programas que permite aos usuários criar e manter um banco de dados. Em resumo, o SGBD é um software que facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados.

> Outras funções importantes fornecidas pelo SGBD incluem proteção do banco de dados e sua manutenção por um longo período. A proteção inclui proteção do sistema contra defeitos (ou falhas) de hardware ou software e proteção de segurança contra acesso não autorizado ou malicioso. Um banco de dados grande pode ter um ciclo de vida de muitos anos, de modo que o SGBD precisa ser capaz de manter o sistema, permitindo que ele evolua à medida que os requisitos mudam com o tempo (ELMASRI, 2011, p.3).

Usar um bom SGBD com as capacidades adequadas traz diversas vantagens ao desenvolvimento do projeto, como o controle de redundância, evitando desperdício de espaço de armazenamento e inconsistência de arquivos, restrição de acessos não autorizados e oferecimento de backup (ELMASRI, 2011).

Requere-se que um sistema seja capaz de recuperar dados de maneira eficiente para que seja funcional. Segundo Silberschatz (2006), os desenvolvedores ocultam a complexidade dos usuários sob níveis de abstração, pois muitos usuários de sistemas de banco de dados não possuem habilidades tecnológicas. Esses níveis de abstração são o nível de view, sendo o nível de abstração mais alto que simplifica a interação com o sistema, o nível lógico, que descreve quais dados estão armazenados no banco de dados e o nível físico, que é o mais baixo e descreve realmente como se dá o armazenamento dos dados (SILBERSCHATZ, 2006). Ramakrishnan (2008) define esses níveis de abstração como conceitual, físico e externo.

Ramakrishnan (2008) introduz o conceito de modelo de dados como uma coleção de construtores de alto nível para descrição dos dados que ocultam diversos detalhes de baixo nível. Assim, um SGBD permite que um usuário defina os dados a serem armazenados em termos de modelo de dados, e a maioria dos SGBDs atuais baseia-se no modelo de dados relacional. Ainda segundo Ramakrishnan, o esquema conceitual, também chamado de esquema lógico, descreve todas as relações que estão armazenadas no banco de dados, contendo informações como entidades e relacionamentos (2008).

Concluindo, ao armazenar dados em um SGBD, é possível utilizar seus recursos para gerenciar os dados de uma forma robusta e eficiente, e seu uso torna-se indispensável em um cenário em que o volume de dados e o número de usuários é crescente (RAMAKRISHNAN, 2008).

2.1.5. MySQL Workbench

Os bancos de dados são sistemas essenciais para a organização eficiente de dados e a escolha de um bom banco de dados influencia diretamente no desenvolvimento de um projeto de software (KLOH, et al., 2023). Ao criar um banco de dados, é necessário definir uma estrutura organizada, estabelecendo o esquema e os tipos de dados. Para interagir com um banco de dados, faz-se necessário um Sistema de Gerenciamento de Banco de Dados (SGBD), ferramenta que gerencia a criação, a manipulação e a recuperação de dados, garantindo a integridade e a segurança dos dados. O SGBD escolhido para o desenvolvimento da aplicação é o MySQL, relacional, de código aberto e que utiliza a linguagem SQL (Linguagem de Consulta Estruturada) como interface.

O MySQL é rápido, confiável, escalonável e fácil de usar. É simples executar consultas complexas usando SQL e também adicionar, atualizar ou excluir tabelas, relações e fazer outras alterações em dados quando necessário, sem alterar a estrutura geral do banco de dados ou afetar aplicativos existentes (GOOGLE CLOUD, s.d.), permitindo que os usuários tanto iniciantes como usuários mais experientes interajam de maneira intuitiva com o banco de dados.

O banco de dados MySQL atua com alguns recursos importantes, como a linguagem de definição de dados, a linguagem de manipulação de dados, a linguagem de consulta de dados, a linguagem de controle de dados e a linguagem de transação de dados, todos respectivamente, essenciais para o sucesso do banco citado (KLOH et al., 2023).

2.1.6. React Native

O React Native, lançado em 2015, é um framework de código aberto e de desenvolvimento cruzado para aplicativos móveis criado pelo Facebook baseado na linguagem JavaScript, que permite a criação de aplicativos móveis renderizados nativamente para iOS e Android, ou seja, a estrutura permite que um aplicativo funcione em diferentes plataformas utilizando a mesma base de código (ALURA, 2023). Esse processo é chamado de programação mobile híbrida, também chamado de desenvolvimento *cross-platform*. O desafio dessa abordagem de desenvolvimento é encontrar pontos em comum entre os diferentes sistemas operacionais, para que dessa forma seja possível criar um software a partir de uma base de código única, possibilitando o aproveitamento e a utilização do máximo de funcionalidades que cada sistema e aparelho possa oferecer (HEITKÖTTER et al. *apud* FALCÃO, 2022).

Uma vez que o React Native permite que o mesmo código utilizado para produzir uma aplicação rode em diferentes plataformas, há um aumento na produtividade, economia de tempo e recursos e também acesso a uma maior base de usuários, pois não é necessário desenvolver o mesmo aplicativo de forma separada e nem utilizar ferramentas diferentes para executar a mesma função (ALURA, 2023). Desenvolver aplicações móveis com React Native conta com vantagens como facilidade em trabalhar, reutilização de código, componentes pré-desenvolvidos e uma base de código executável por várias plataformas.

Ainda acerca da utilização do React Native, a aplicação também contará com o uso do framework Expo. O Expo é um projeto de código aberto que oferece aos desenvolvedores ferramentas poderosas para ajudar na criação e manutenção de aplicativos React Native em qualquer escala. Ele é formado por um conjunto de ferramentas e serviços criados com o foco nos ambientes do React Native e das plataformas nativas, que auxiliam no desenvolvimento,

construção, implantação, testes e na execução de simuladores nos sistemas Android e iOS utilizando a mesma base de código (EXPO, 2025).

> O Expo funciona como uma camada de abstração para uma aplicação React Native. Ele elimina a necessidade do desenvolvedor ter de lidar com códigos das plataformas nativas, o que pode ocorrer quando se está utilizando apenas o React Native. Dessa forma, é preciso lidar somente com JavaScript no ambiente de desenvolvimento (FALCÃO, 2022, p.22).

A documentação oficial do React Native e do Expo são fontes de referências fundamentais para o desenvolvimento, disponibilizando informações detalhadas e atualizadas sobre as funcionalidades, tutoriais e exemplos de código que auxiliam no uso dessas ferramentas.

2.1.7. Scrum

A metodologia Scrum é uma das que mais auxiliam na organização e produtividade durante o desenvolvimento, por meio da separação de tarefas. Também é extremamente efetiva na redução do tempo de entrega, contribuindo para agilizar a conclusão da aplicação.

A metodologia Scrum é uma metodologia ágil que foi idealizada por três programadores que participaram do Manifesto Ágil, este sendo uma declaração de valores e princípios fundamentais para o desenvolvimento de software (CUNNINGHAM, 2001). Os princípios e valores do Manifesto Ágil são de extrema importância e prestam um papel essencial no desenvolvimento de sistemas, tendo como prioridade a satisfação do cliente por meio da entrega contínua, flexibilidade quanto a mudanças nos requisitos e participação do cliente no projeto em conjunto com os desenvolvedores.

> Scrum é fundamentado nas teorias empíricas de controle de processo, ou empirismo. O empirismo afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos. (SCHWABER, 2013, p.4)

As equipes de desenvolvimento utilizam artefatos Scrum, que fornecem informações sobre planejamento e tarefas aos membros, para resolver problemas e gerenciar os projetos. Os três principais artefatos do Scrum são o Product Backlog, Sprint Backlog e Incremento. O Scrum também define três funções específicas para as equipes: o responsável pelo produto, o líder Scrum e o time de desenvolvimento. Além dos artefatos e funções, o Scrum também conta com eventos (ou cerimônias) que podem ser definidas como um conjunto de reuniões regularmente realizadas pelos times. No entanto, o conceito chave abordado no desenvolvimento da aplicação é o conceito de Sprint.

Uma Sprint é o período de tempo definido para concluir as tarefas definidas pela equipe. Uma Sprint dura em torno de duas semanas, porém a metodologia flexível permite que o período seja variável e se adeque às necessidades do projeto. No início de cada Sprint, são definidas as tarefas para o período de tempo definido e, após o fim de cada Sprint, o progresso é revisado e discutido, permitindo a melhoria contínua no desenvolvimento.

Apesar de o Scrum tradicional ser uma metodologia mais voltada para times de desenvolvimento, seus pilares possuem valores significativos que podem ser adaptados para o uso individual. São eles a transparência, a inspeção e a adaptação (SCHWABER, 2013). Em meados de 2016, o Guia do Scrum foi atualizado por seus criadores Ken Schwaber e Jeff Sutherland, e a atualização trouxe os seguintes valores ao Scrum: Coragem, Foco, Comprometimento, Respeito e Abertura. Seguir esses valores e colocá-los em prática no dia a dia são práticas fundamentais que contribuem com a produtividade, a flexibilidade e a organização no desenvolvimento.

2.1.8. Node.js

Node.js é um ambiente de execução de código JavaScript grátis, de código aberto e multiplataforma, que permite que os desenvolvedores criem servidores, aplicações web, ferramentas de linhas de comando e scripts. Foi criado em 2009 para construir aplicações escaláveis e hoje é uma ferramenta popular para quase qualquer tipo de projeto (NODE.JS, 2025).

O Node.js é um ambiente do lado servidor (server side), que na prática se reflete na possibilidade de criar aplicações standalone (autossuficientes) em uma máquina servidora, sem a necessidade do navegador (ALURA, 2023). Diferente dos servidores tradicionais que criam uma nova thread para cada requisição, o Node.js opera em um único processo com um modelo de I/O (input/output, ou entrada/saída) assíncrono e não bloqueante. Isso significa que, ao realizar operações como leitura de arquivos, acesso a banco de dados ou rede, o Node.js não bloqueia o fluxo do programa. Ao invés disso, ele continua executando outras tarefas e retoma a operação quando a resposta chega, conseguindo então lidar com milhares de conexões simultâneas de forma eficiente.

2.1.9. Express.js

O Express é um framework de aplicações web Node.js flexível que oferece um conjunto robusto de funções para aplicações web e mobile, facilitando o desenvolvimento de APIs com diversos utilitários para métodos HTTP (EXPRESS, 2025).

No Express.js, o routing, ou roteamento, é o mecanismo que define como a aplicação responde às requisições feitas a determinados caminhos (endpoints), utilizando métodos HTTP como GET, POST, PUT e DELETE. Ele é configurado por meio de métodos do objeto app, que representam esses métodos HTTP. Por exemplo, app.get() define um manipulador para requisições GET, enquanto app.post() é usado para POST, e assim por diante. Quando uma requisição chega a uma rota definida, o Express chama a função callback associada, permitindo que a aplicação processe a requisição e envie uma resposta (EXPRESS, 2025). Esse sistema de routing torna o framework Express.js muito flexível e modular, de forma que permite que o desenvolvedor organize a lógica da aplicação de forma clara e reutilizável.

2.1.10. Sequelize ORM

A sigla ORM significa Object Relational Mapping, em português, Mapeamento Objeto-Relacional. ORM é uma técnica que tem como objetivo criar uma camada de mapeamento entre um modelo de objetos, como uma aplicação, e o modelo relacional de banco de dados, a fim de abstrair o acesso ao mesmo. Assim, foram criados diversos frameworks ORM ao longo do tempo (SATO, 2013).

Ainda segundo Sato, os frameworks se encontram em uma camada intermediária entre a lógica da aplicação e o SGBD e recebem as solicitações de interação com o SGDB através de objetos, dessa forma gerando, de forma automática, todo o SQL necessário para a operação solicitada. Ao adotar esta técnica, o sistema passa a interagir com o ORM e não com a base de dados de forma direta, aumentando a produtividade por não ter a necessidade de escrever códigos SQL, possibilitando a interação pelo código da aplicação (BARSOTI; GIBERTONI, 2020).

Um desses frameworks é o Sequelize ORM, que foi adotado no desenvolvimento deste projeto. O Sequelize é um ORM para TypeScript e Node.js que oferece um suporte robusto a transações, relacionamentos entre tabelas, replicação de leitura e muito mais; também suporta bancos de dados como Oracle, Postgres, MySQL, MariaDB, entre outros (SEQUELIZE ORG, 2025). Além disso, o Sequelize oferece uma documentação bastante completa que conta com guias que cobrem desde os conceitos básicos de modelagem até tópicos mais avançados. Assim, este frameworks torna-se uma ferramenta poderosa e flexível para projetos Node.js que necessitam de integração com bancos relacionais de forma estruturada e moderna.

2.1.11 Swagger UI

O Swagger UI é uma ferramenta open source do Swagger que permite que qualquer pessoa visualize e interaja com os recursos de uma API, sem precisar de qualquer implementação lógica. A documentação visual facilita na implementação do backend e também no consumo do lado do cliente (SWAGGER, 2025). Por fornecer uma visão detalhada das rotas ao listar todos os endpoints disponíveis, juntamente com os respectivos status HTTP de resposta, esta ferramenta foi utilizada da documentação da API elaborada para este projeto.

3. METODOLOGIA

3.1. Metodologia de Pesquisa

Para a fundamentação teórica, a metodologia utilizada foi a de revisão bibliográfica. Segundo Gil (2008), o primeiro procedimento adotado em uma pesquisa bibliográfica consiste na formulação do problema que se deseja investigar. O problema em questão baseou-se no questionamento de como a tecnologia poderia auxiliar nas questões de mobilidade e diminuição na emissão de gases poluentes por meio da prática de *carpooling*.

3.2. Metodologia de Desenvolvimento de Software

O presente trabalho trata-se do desenvolvimento de um aplicativo mobile para o compartilhamento de caronas solidárias entre os alunos do IFRO, o qual possibilitará que os alunos possam tanto oferecer como solicitar caronas uns aos outros. Sendo desenvolvido em React Native, uma linguagem que permite o desenvolvimento cross-platform, o aplicativo estará disponível tanto para aparelhos iOS como Android, bastando apenas estar conectado à internet.

O aplicativo foi implementado tendo como base a linguagem de programação TypeScript, uma vez que o framework escolhido para desenvolver o front-end foi o React Native. O editor de código utilizado foi o Visual Studio Code e o banco de dados foi construído no SGBD MySQL Workbench. Também foi utilizada a ferramenta Astah UML para a modelagem de diagramas UML e a plataforma Figma para construção do protótipo do aplicativo.

Também foi desenvolvida uma API própria para o projeto. O conceito de API (Application Programming Interface - Interface de Programação de Aplicação) pode ser definido como um mecanismo que permite que dois componentes de software se comuniquem usando um conjunto de definições e protocolos (AWS, 2024). APIs são conjuntos de ferramentas, definições e protocolos para a criação de aplicações de software, funcionando como intermediários que conectam soluções e serviços, sem precisar saber como eles foram implementados (RED HAT, 2023).

Ao utilizar APIs, o processo de desenvolvimento de novas aplicações ou de gerenciar aplicações já existentes é simplificado, pois oferecem flexibilidade para facilitar o design, administração e uso das mesmas, auxiliando ainda na colaboração entre as empresas e equipes de TI. APIs podem ser privadas, onde seu uso é exclusivamente interno; Públicas, onde são disponibilizadas para todos, de forma que terceiros possam desenvolver novas aplicações que interajam com a API; E de parceiros, onde é compartilhada com parceiros de negócios específicos (RED HAT, 2023).

O tipo de API mais popular e flexível encontrado atualmente é o REST, sua principal característica sendo a ausência de estado, ou seja, o servidor não salva dados do cliente entre as solicitações HTTP. Neste modelo,o cliente envia solicitações ao servidor como dados, o servidor usa essa entrada do cliente para iniciar funções internas, e então retorna os dados de saída ao cliente (AWS, 2024).

Para a construção da API, foi utilizado o Node.js, juntamente ao framework Express.js para auxiliar no roteamento e ao Sequelize ORM para estabelecer a comunicação com o banco de dados. Toda a API foi devidamente documentada no Swagger para ter uma melhor visualização dos testes.

Para o desenvolvimento da aplicação, foram seguidos passos fundamentais para o desenvolvimento baseado em boas práticas de programação, tais como:

A criação do documento de requisitos do software. Os principais requisitos funcionais e não funcionais do aplicativo foram levantados e registrados no Documento de Requisitos, conforme as páginas 7-18 do APÊNDICE A.

A criação de um protótipo interativo no Figma, com o design das telas do aplicativo e as respectivas mensagens de confirmação para melhor visualização do sistema, o qual pode ser conferido na página 19 do APÊNDICE A.

A criação de diagramas UML como Diagrama de Caso de Uso. O diagrama permite visualizar os principais usuários que irão interagir com a aplicação e suas interações, conforme a página 20 do APÊNDICE A.

A escrita dos Casos de Uso Expandidos, que descrevem detalhadamente as funcionalidades do aplicativo, conforme as páginas 20-32 do APÊNDICE A.

A criação de um modelo lógico entidade-relacionamento, que descreve a estrutura do banco de dados, conforme a página 33 do APÊNDICE A.

Acerca da metodologia Scrum, foram adotados os artefatos Product Backlog, este sendo uma lista de todas as tarefas a serem feitas, Sprint e Incremento, além das funções Time de desenvolvimento e Product Owner.

4. RESULTADOS E DISCUSSÕES

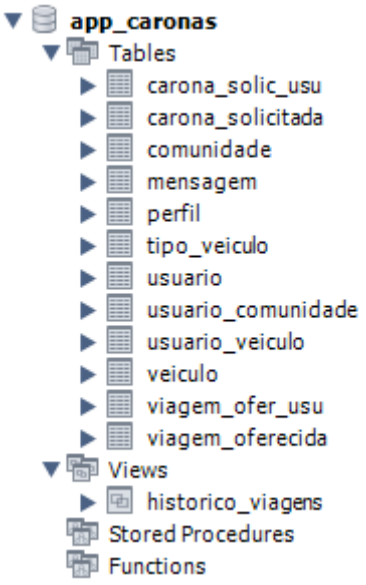
Este projeto teve como resultado um ecossistema próprio que consiste em: um banco de dados relacional bem estruturado construído no SGBD MySQL, uma API REST desenvolvida em Node.js com auxílio dos frameworks Express.js e Sequelize responsável por intermediar a comunicação entre o frontend e o banco de dados, e um aplicativo React Native integrado com Expo, onde todas as partes se comunicam de forma efetiva e coerente.

A API e o banco de dados operam localmente, configurados para interagir via consultas SQL gerenciadas pelo Sequelize. Desta forma, o aplicativo mobile consome os endpoints da API de modo síncrono, garantindo a transferência eficiente dos dados. Além disso, o frontend foi projetado de forma fiel ao protótipo desenvolvido previamente com a ferramenta Figma, o que assegurou o alinhamento entre as expectativas de UX e o produto final.

A seguir, serão apresentadas imagens dos resultados obtidos, seguidas de algumas considerações e explicações acerca de cada funcionalidade.

4.1. Banco de dados

Figura 1 - Estrutura do banco de dados



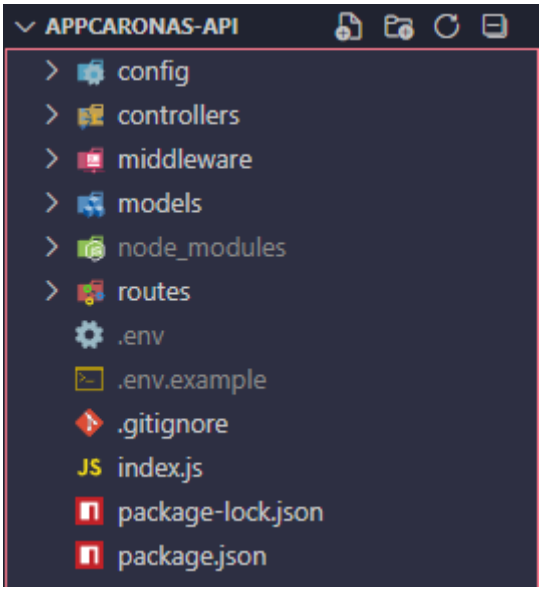
Fonte: Elaboração própria, 2025.

Na imagem, pode-se observar as tabelas presentes no banco. O banco de dados foi construído tendo como base o Diagrama Lógico Entidade-Relacionamento, que foi projetado juntamente ao Projeto de Software. O histórico de viagens foi adicionado em formato de view ao

invés de tabela, uma vez que os dados mostrados no histórico não poderão ser alterados ou excluídos, apenas visualizados. A comunicação do aplicativo com o banco de dados ocorre por meio da API própria, cuja estrutura pode ser conferida na próxima seção.

4.2. API REST

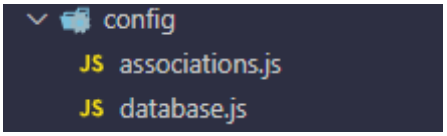
Figura 2 - Estrutura da API



Fonte: Elaboração própria, 2025.

A API está dividida nas pastas config (contém arquivos referentes à configuração com o banco de dados), controllers (contém arquivos responsáveis por receber todas as requisições do usuário em cada rota), middleware (contendo códigos para tratamento de erros), models (contendo os arquivos que representam a camada de dados da aplicação) e routes (contém os arquivos de regras que ligam uma URL a uma ação específica dentro da aplicação, estes que estão de acordo com o padrão do Swagger) e, além disso, possui os arquivos .env e index.js. O .env define as informações de porta, nome do host, nome, porta, usuário e senha do banco de dados. O index.js, por sua vez, é o ponto de entrada principal da aplicação Node, sendo responsável por configurar e iniciar o servidor, além de registrar middlewares, rotas e documentação da API com o Swagger.

Figura 3 - Pasta config



Fonte: Elaboração própria, 2025.

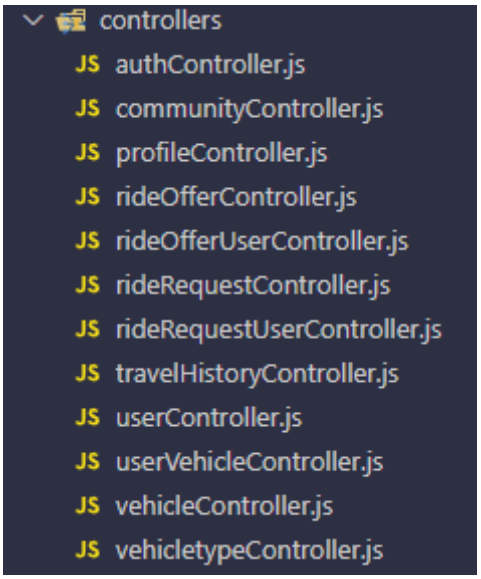
Dentro da pasta config, há os arquivos database.js, onde se encontra a instância sequelize que se conecta com o banco de dados puxando as informações presentes no arquivo .env e, em associations.js, se encontram os códigos de conexão com o banco via sequelize, a importação dos modelos e as associações entre os objetos, representando os relacionamentos entre as tabelas no banco de dados.

Figura 4 - Pasta models

Fonte: Elaboração própria, 2025.

Dentro da pasta models (modelos) estão todos os modelos que representam as tabelas do banco de dados, sendo mapeados pela instância do sequelize. Nos modelos é possível definir o tipo de dado dos atributos, se é chave primária ou chave estrangeira, implementar o auto-incremento e também configurar para não aceitar valores nulos. É importante mencionar que o nome dos atributos devem ser nomeados exatamente como estão no banco, para que a leitura e manipulação dos dados possa ser realizada com êxito.

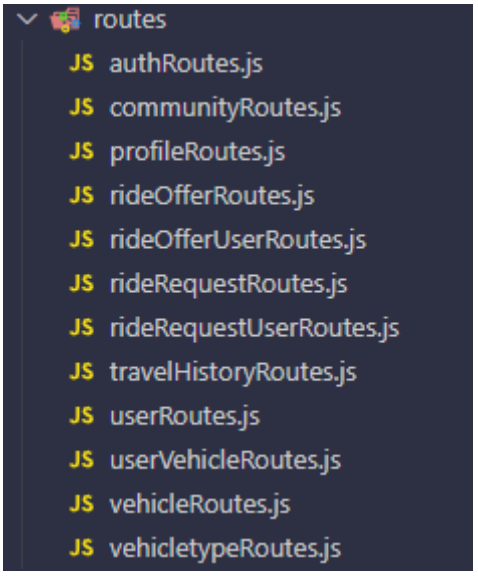
Figura 5 - Pasta controllers



Fonte: Elaboração própria, 2025.

Em controllers (controladores), encontram-se os arquivos que contém a lógica de negócio das rotas da API, incluindo ações como buscar dados no banco, criar registros, atualizar ou deletar informações, além de tratar erros e enviar respostas ao cliente. Em cada arquivo há métodos como getAll, getById, create, put, post e delete.

Figura 6 - Pasta routes



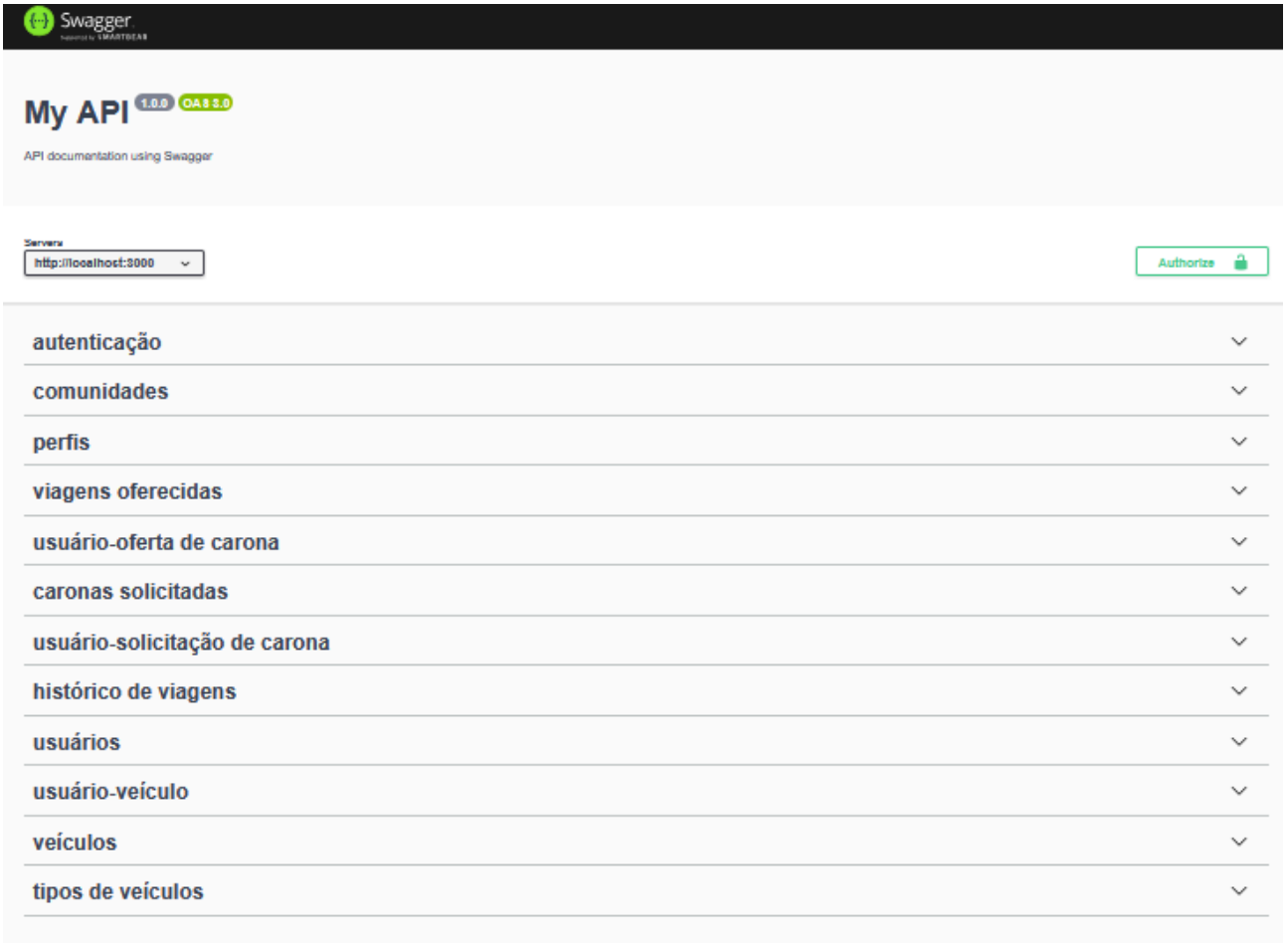
Fonte: Elaboração própria, 2025.

Na pasta routes (rotas), há os arquivos responsáveis por definir os endpoints (caminhos) da API e associá-los aos respectivos controllers que contêm a lógica de execução. Esta pasta serve para

organizar e separar as rotas de forma modular, agrupando-as por recursos como usuários, veículos, caronas, etc. Os arquivos também contêm anotações Swagger no formato de comentários, que servem para documentar cada rota, informando seu funcionamento, parâmetros esperados, tipos de resposta e códigos HTTP. Isso permite gerar automaticamente uma interface de documentação interativa e informativa.

A função dos arquivos na pasta routes é organizar os pontos de entrada da API para que, ao receber uma requisição, o Express saiba exatamente qual função executar. Essa separação de responsabilidades entre routes e controllers é bastante útil para manter o código mais limpo e fácil de testar.

Figura 7 - Swagger UI

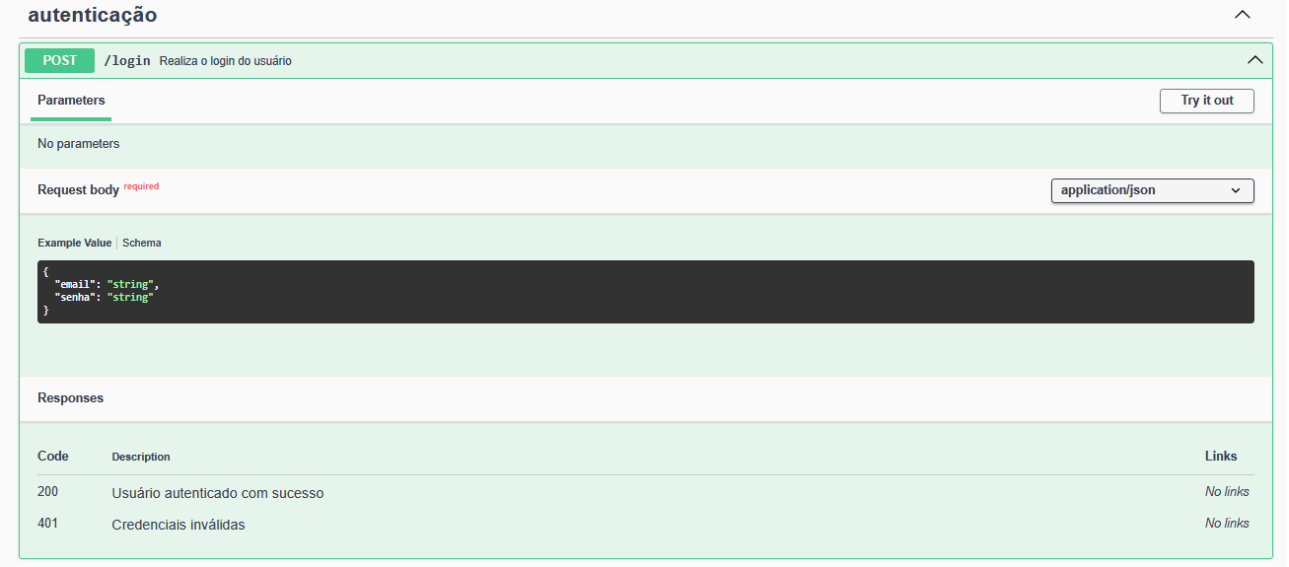


Fonte: Elaboração própria, 2025.

As rotas da API foram documentadas via Swagger. Para esta integração, foi preciso instalar os pacotes necessários e configurar a conexão do Swagger com o Express. Após isso, bastou escrever os comentários especiais no formato definido pela documentação do Swagger dentro da

pasta routes da API. E então, ao acessar http://localhost:3000/api-docs (ou a porta definida na aplicação), o Swagger UI irá exibir automaticamente toda a documentação da API com base nas anotações feitas. Na imagem, visualiza-se todas as rotas da API REST, sendo que cada endpoint (extremidade de uma conexão de API, onde são recebidas as chamadas dos métodos) será abordado individualmente a seguir.

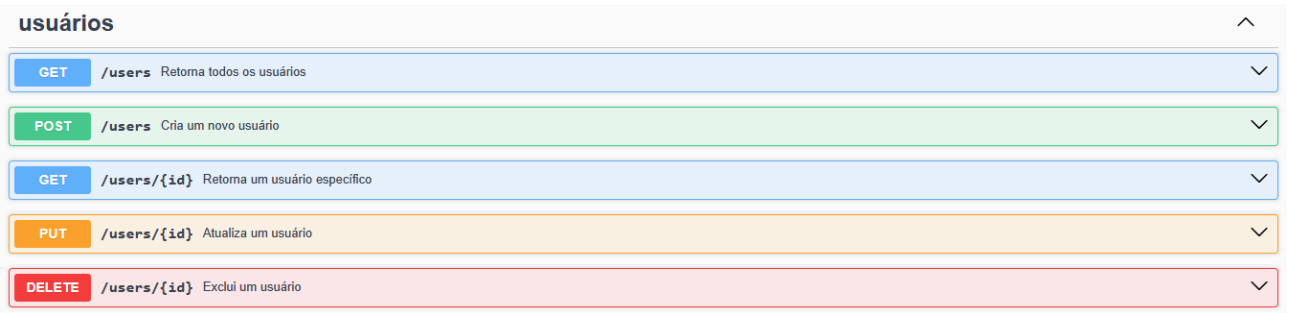
Figura 8 - Endpoint de autenticação



Fonte: Elaboração própria, 2025.

Esta rota possui apenas um método de requisição HTTP, que é o método post. Esta rota é responsável por realizar a autenticação de login do usuário no aplicativo, fazendo a validação das credenciais inseridas.

Figura 9 - Endpoint de usuários

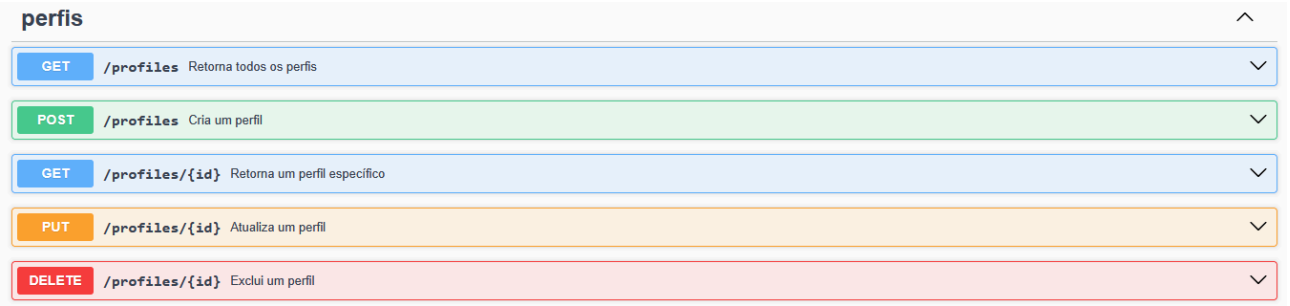


Fonte: Elaboração própria, 2025.

A rota de usuários possui os métodos get, getById, post, put e delete, possibilitando fazer o

controle dos dados de cadastro de um usuário e sendo responsável também por registrar os novos dados de cadastro no banco de dados.

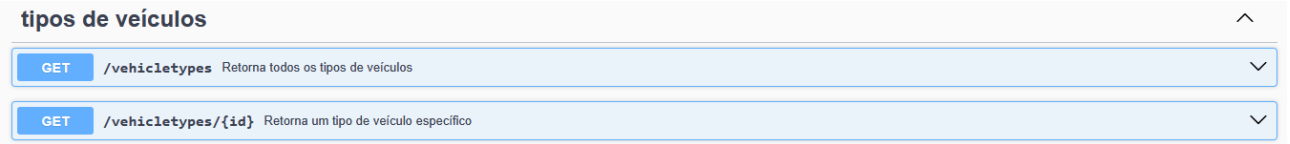
Figura 10 - Endpoint de perfis



Fonte: Elaboração própria, 2025.

Esta rota é responsável pela criação e manutenção dos perfis dentro do aplicativo, seguindo a lógica de separar o cadastro do usuário com seu perfil na plataforma. A partir do momento que o usuário cria sua conta, ele pode editar as informações de seu perfil, customizando o mesmo da forma que mais lhe agrada; No entanto, dados mais sensíveis como telefone, e-mail e senha devem ser alterados em uma seção separada, no caso, na rota de usuários.

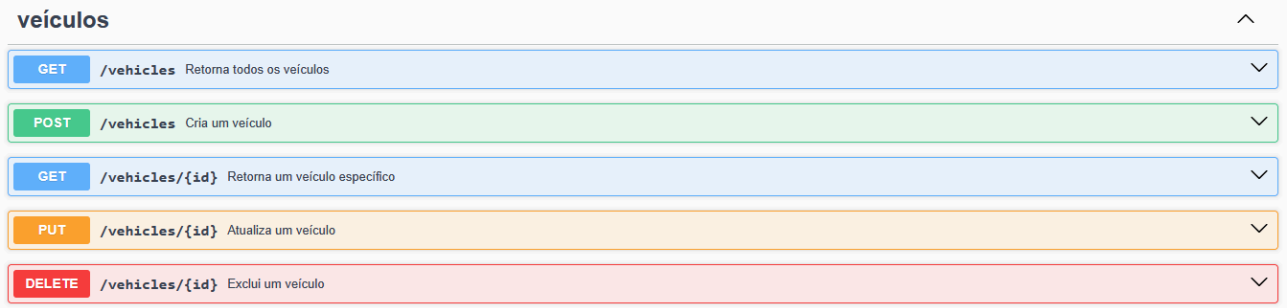
Figura 11 - Endpoint de tipos de veículos



Fonte: Elaboração própria, 2025.

Esta rota possui apenas dois métodos get, uma vez que não é interessante permitir a edição, adição ou exclusão de tipos de veículos. Para evitar duplicidade em registros, este modelo possui dois tipos de veículo pré-cadastrados, estes sendo carro ou moto. A rota permite visualizar estes registros e o id dos mesmos no banco.

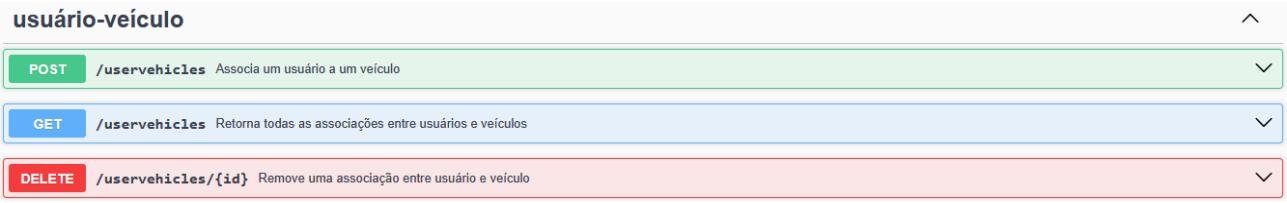
Figura 12 - Endpoint de veículos



Fonte: Elaboração própria, 2025.

A rota de veículos possui todos os métodos necessários para gerenciamento das informações do cadastro de um novo veículo. No cadastro, ao inserir o tipo do veículo, o usuário deverá selecionar se seu veículo é carro ou moto por meio de uma caixa de seleção, o que melhora a experiência do usuário e também evita a duplicidade de registros.

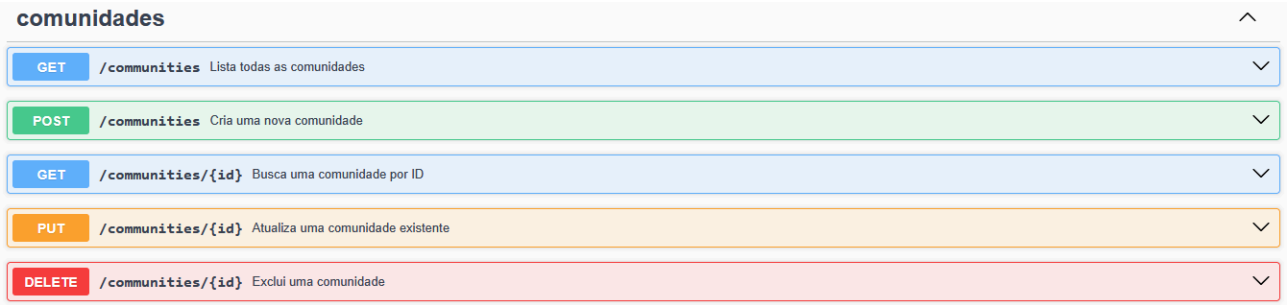
Figura 13 - Endpoint de usuário-veículo



Fonte: Elaboração própria, 2025.

Esta rota é responsável por fazer a ligação entre um registro de usuário com um registro de veículo cadastrado anteriormente. Foi elaborada seguindo o raciocínio de que um usuário pode ter um ou mais veículos, assim como um veículo pode ter mais de um motorista, então fez-se necessária a criação de uma terceira tabela para registrar este relacionamento muitos-para-muitos no banco, refletindo, assim, na criação de uma rota intermediária para realizar essa ligação.

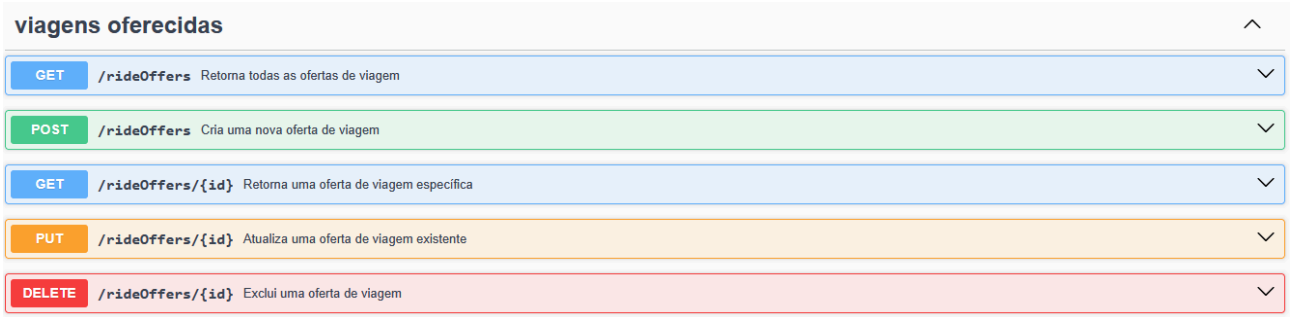
Figura 14 - Endpoint de comunidades



Fonte: Elaboração própria, 2025.

A rota de comunidades permite visualizar, criar, editar e excluir comunidades dentro do aplicativo. O intuito das comunidades é possibilitar a criação de espaços nos quais os usuários possam entrar, facilitando o compartilhamento de caronas entre os mesmos.

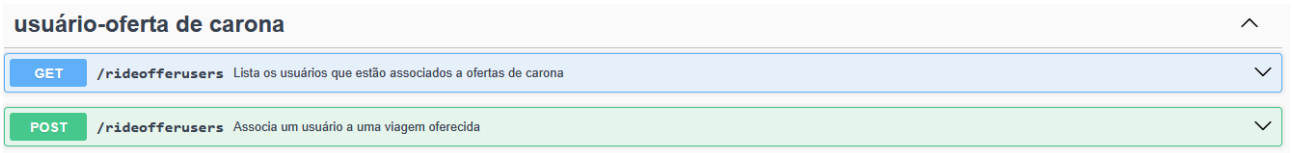
Figura 15 - Endpoint de viagens oferecidas



Fonte: Elaboração própria, 2025.

As viagens dentro da API foram divididas em dois tipos: viagens oferecidas e caronas solicitadas. Isso se dá à característica do aplicativo de possibilitar que o mesmo usuário possa ser tanto motorista, ao oferecer uma viagem, como passageiro, ao fazer uma solicitação de carona. A rota de viagens oferecidas serve para fazer o cadastro, edição, visualização ou exclusão de uma carona pelo usuário motorista.

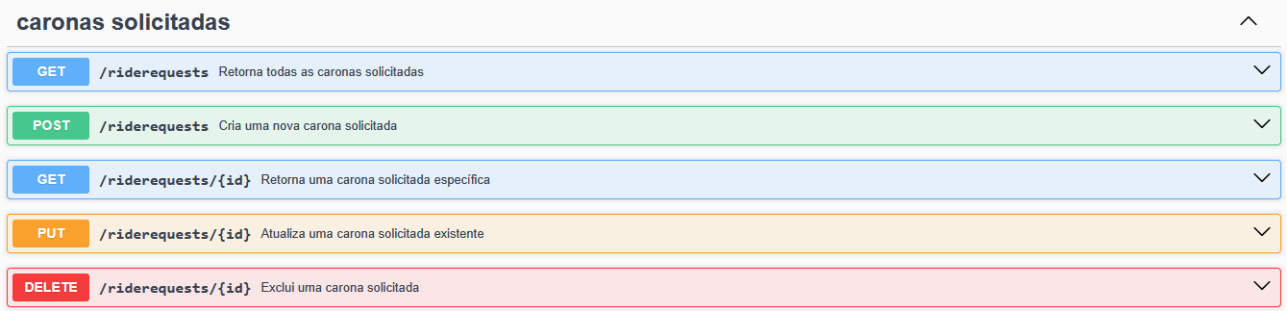
Figura 16 - Endpoint de usuário-oferta de carona



Fonte: Elaboração própria, 2025.

O relacionamento de usuário com viagem oferecida é de muitos-para-muitos, uma vez que mais de um usuário terá ligação com a oferta de carona e uma oferta de carona poderá ter ligação com mais de um usuário. Seguindo esta lógica, fez-se necessária a criação de uma terceira tabela para registrar este relacionamento muitos-para-muitos no banco, refletindo, assim, na criação de uma rota intermediária para realizar essa ligação.

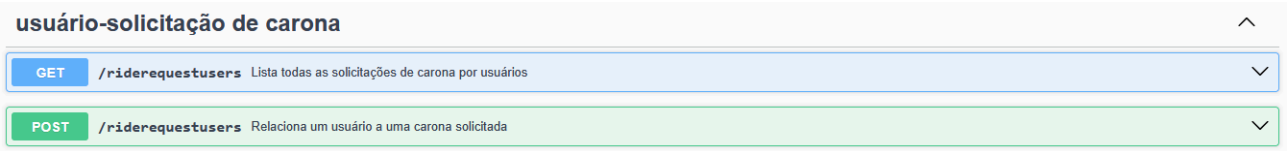
Figura 17 - Endpoint de caronas solicitadas



Fonte: Elaboração própria, 2025.

A rota de caronas solicitadas serve para fazer o cadastro, edição, visualização ou exclusão de uma carona pelo usuário passageiro.

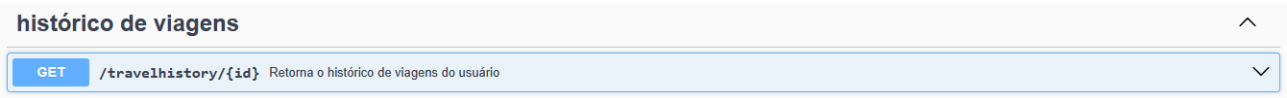
Figura 18 - Endpoint de usuário-solicitação de caronas



Fonte: Elaboração própria, 2025.

Semelhante ao relacionamento de usuário com viagem oferecida, o relacionamento de usuário com solicitação de carona também é de muitos-para-muitos. Seguindo esta lógica, fez-se necessária a criação de uma terceira tabela para registrar este relacionamento no banco, refletindo, assim, na criação de uma rota intermediária para realizar essa ligação.

Figura 19 - Endpoint de histórico de viagens



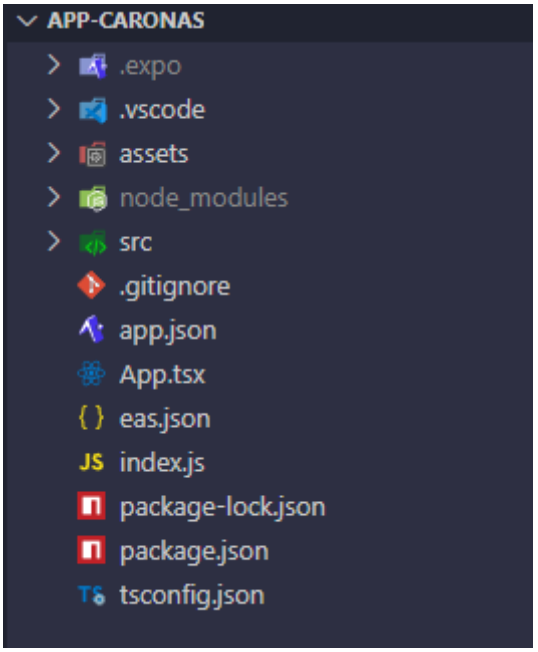
Fonte: Elaboração própria, 2025.

O histórico de viagens consiste em não uma tabela, mas sim em uma view criada no banco de dados que apenas seleciona os registros de viagens oferecidas e caronas solicitadas, então não é possível realizar alterações nos dados que já foram registrados e, por isso, a rota possui apenas um método. Dessa forma, é possível visualizar o resultado da consulta através do método get.

4.3. Aplicativo mobile

O aplicativo foi desenvolvido utilizando a linguagem de programação React Native, utilizando a IDE Visual Studio Code. Uma vez que a stack (conjunto de tecnologias usadas em desenvolvimento de software) definida foi: MySQL para o banco de dados, Node.js com Express e Sequelize para a API e React Native com Expo para o aplicativo, além do uso de TypeScript em todas as camadas do projeto, houve uma vantagem significativa que contribuiu com a produtividade e escalabilidade da aplicação. A stack escolhida proporcionou um ambiente de desenvolvimento eficiente, oferecendo uma base sólida para a construção do aplicativo.

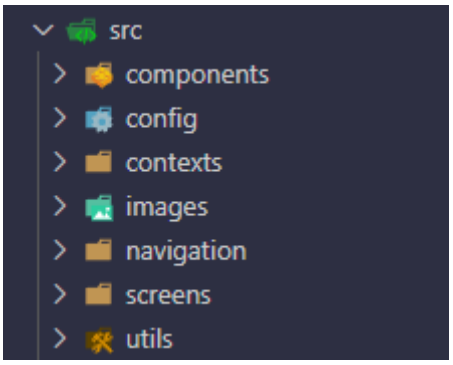
Figura 20 - Estrutura de pastas do projeto React Native



Fonte: Elaboração própria, 2025.

Na imagem a seguir, é possível visualizar a estrutura presente dentro da pasta src, que carrega os principais componentes do projeto.

Figura 21 - Pasta src



Fonte: Elaboração própria, 2025.

A pasta components é responsável por conter os componentes personalizados, como botões, campos de input e pop-ups de alerta, o que oferece diversas vantagens para o desenvolvimento do frontend do projeto. Ao criar componentes personalizados, é possível evitar a repetição do código em várias telas, tornando o projeto mais limpo. Além disso, o projeto se torna mais organizado e padronizado visualmente, uma vez que os componentes garantem uma consistência de aparência e comportamento em toda a aplicação.

O banco de dados e a API estão rodando de forma local na máquina e, diante disso, foi necessário utilizar uma variável de estado para armazenar o endereço e a porta da API. A pasta config possui o arquivo api.ts, que é o responsável por guardar esses dados, permitindo a integração das aplicações.

A pasta contexts guarda o arquivo de contexto de autenticação, responsável por gerenciar o estado de autenticação do usuário em toda a aplicação. A pasta images é auto explicativa, contendo todas as imagens utilizadas dentro do aplicativo.

A pasta navigation contém os componentes responsáveis pela navegação do aplicativo, centralizando a lógica dos fluxos de navegação e possibilitando a modificação de rotas. Essa estrutura de navegação auxilia na redução de código duplicado e agrupa as rotas por contexto.

A pasta screens, por sua vez, possui todos os arquivos de telas do aplicativo, estas que fazem uso dos arquivos de todas as outras pastas. Esse ambiente de desenvolvimento organizado proporciona facilidade para localizar os arquivos necessários, melhorando o processo de construção do aplicativo.

As telas foram tendo como base o design do protótipo do aplicativo construído no Figma, resultando em um produto final cuja interface se assemelha ao protótipo inicialmente pensado. A princípio, o projeto foi testado e visualizado com uso de um emulador de um aparelho Android utilizando Android Studio, porém, posteriormente, passou a ser testado em um aparelho real

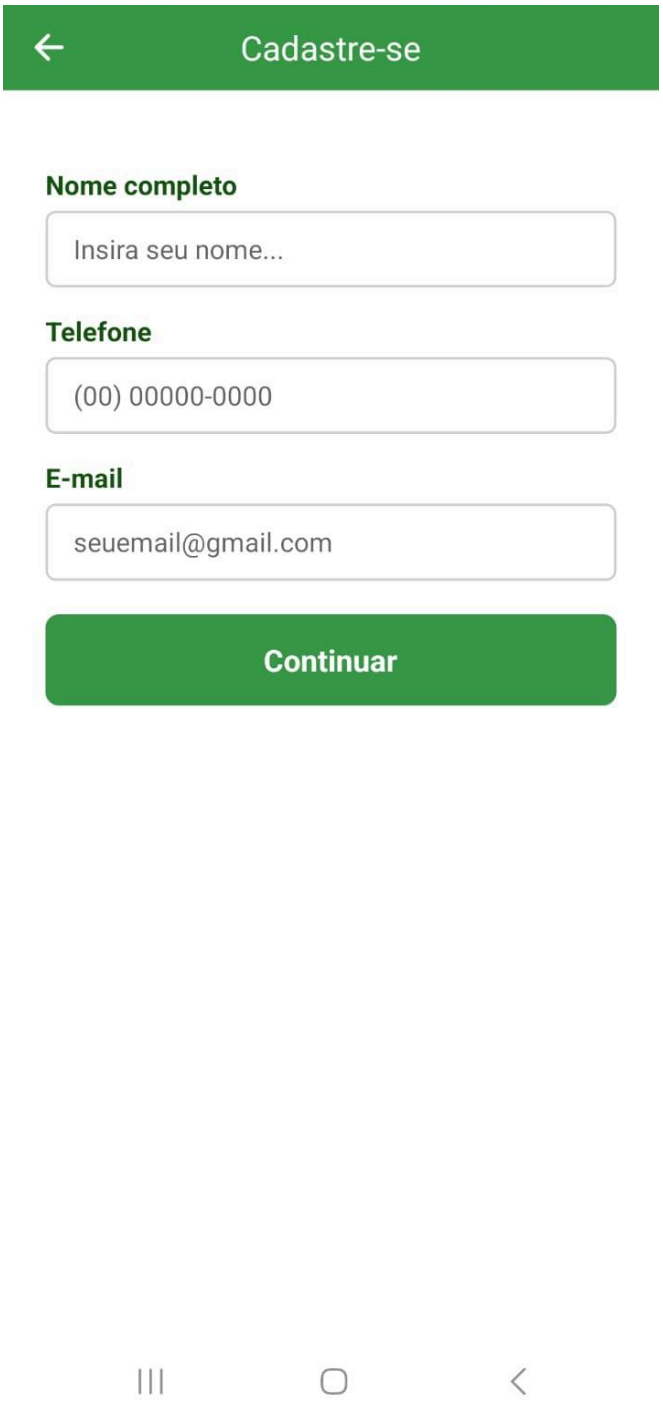
utilizando o aplicativo sandbox (um ambiente isolado e controlado usado para testes e execução de código) do Expo chamado Expo Go. A seguir, é possível visualizar os prints do aplicativo rodando em um aparelho Android real.

Figura 22 - Tela inicial do aplicativo

Fonte: Elaboração própria, 2025.

Na tela inicial do aplicativo, temos uma ilustração acompanhada de texto e dois botões. Ao clicar no botão Cadastrar-se, o usuário é levado para a próxima tela, que é o início do fluxo de cadastro de um novo usuário no aplicativo.

Figura 23 - Primeira tela de cadastro



Fonte: Elaboração própria, 2025.

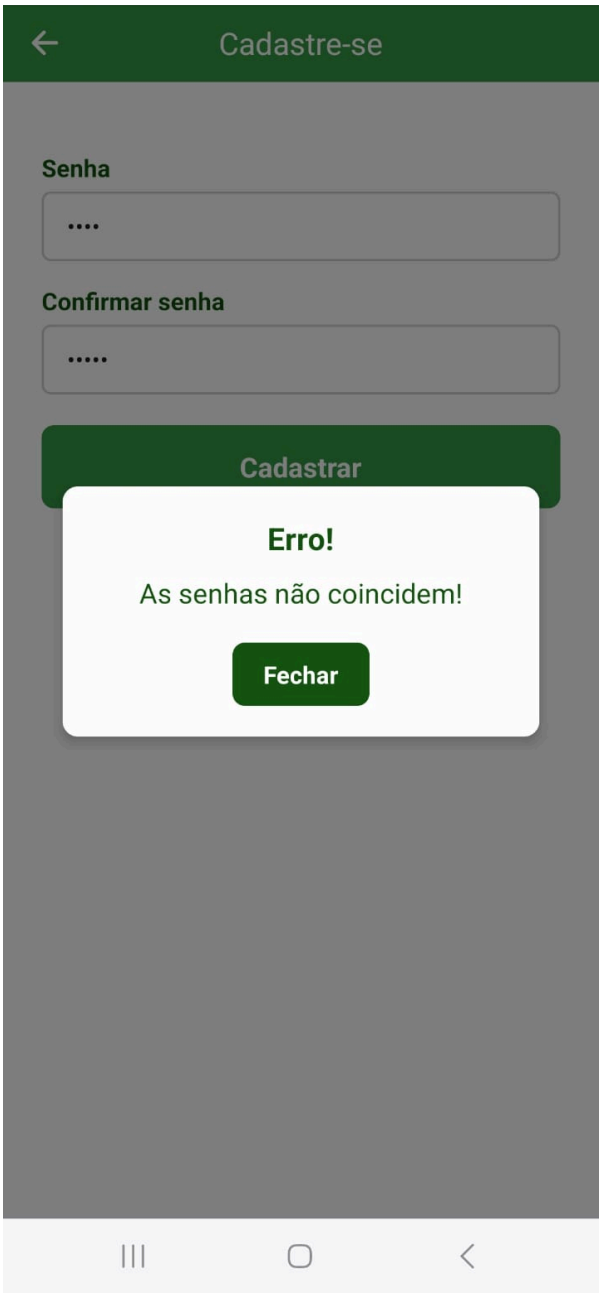
Nessa tela, são solicitadas as informações de nome completo, telefone e e-mail do usuário. O campo de telefone possui máscara e formata o valor automaticamente conforme o usuário digita. O campo de e-mail tem uma condição de validação que permite apenas endereços de e-mail válidos.

Figura 24 - Segunda tela de cadastro

Fonte: Elaboração própria, 2025.

A segunda parte do fluxo de cadastro solicita que o usuário defina uma senha. Na tela, há campos para inserir a senha e inserir novamente para confirmação. Há uma validação que verifica se as duas senhas digitadas são iguais.

Figura 25 - Alerta de erro: senhas não coincidem



Fonte: Elaboração própria, 2025.

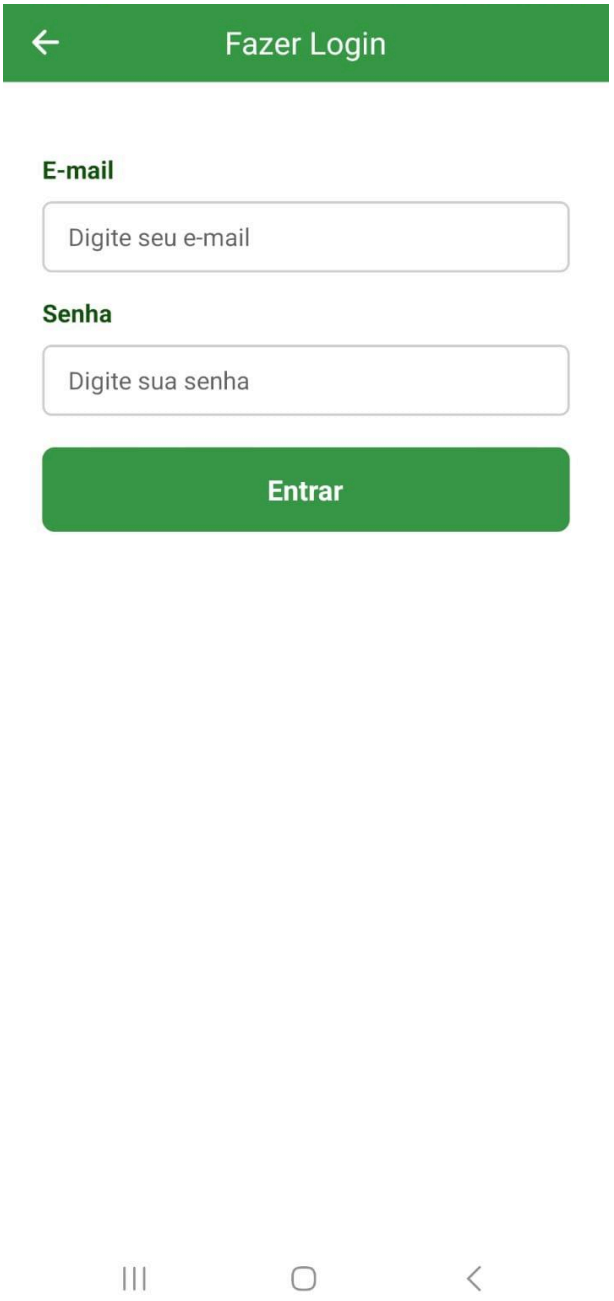
Caso a senha inserida no segundo campo não corresponda à primeira, o aplicativo mostra uma mensagem de alerta para o usuário para alertá-lo.

Figura 26 - Mensagem de confirmação: cadastro realizado

Fonte: Elaboração própria, 2025.

Caso os dados solicitados no cadastro estejam todos válidos, o aplicativo emite uma mensagem de confirmação informando ao usuário que seu cadastro foi realizado com sucesso. Após fechar este alerta, o usuário é automaticamente direcionado à tela de login, que também pode ser acessada por usuário que já possuem cadastro ao clicar no botão Entrar disponível na tela inicial do aplicativo.

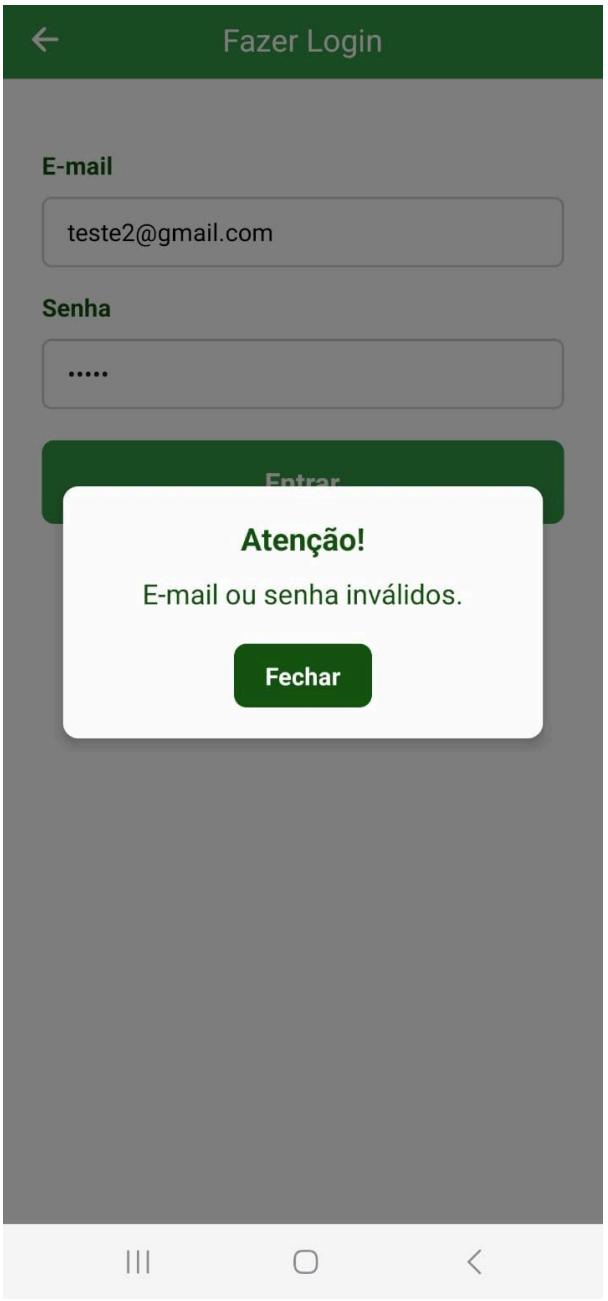
Figura 27 - Tela de login



Fonte: Elaboração própria, 2025.

A tela de login pede as credenciais e-mail e senha que o usuário inseriu no momento de seu cadastro. Caso o aplicativo não encontre os dados digitados nesses campos no banco de dados, uma mensagem de alerta é emitida para o usuário:

Figura 28 - Alerta de erro: credenciais inválidas



Fonte: Elaboração própria, 2025.

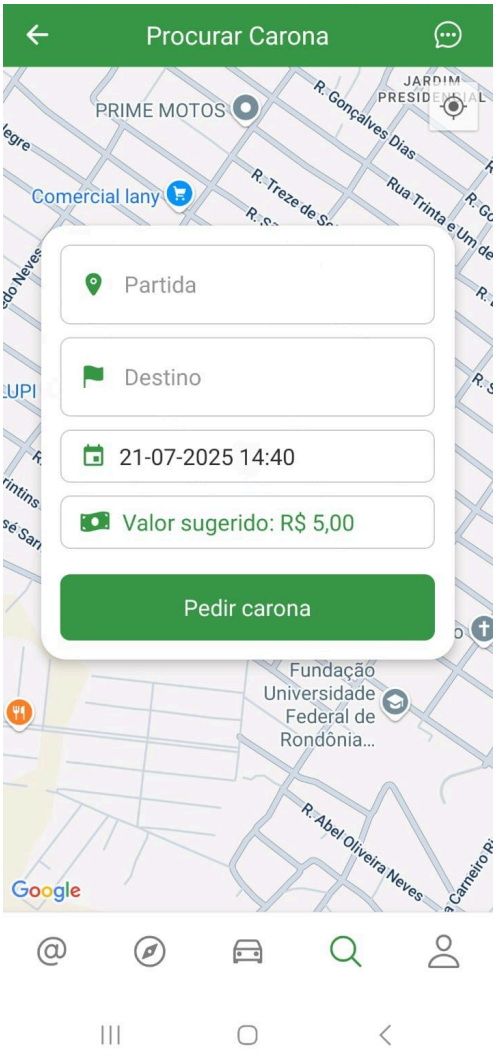
O login bem sucedido permite que o usuário entre em sua conta no aplicativo, redirecionando-o para a tela de Procurar Carona de forma padrão.

Figura 29 - Menu de navegação

Fonte: Elaboração própria, 2025.

A navegação dentro do aplicativo pode ser feita por meio do menu inferior, que possui cinco ícones. O ícone de arroba dá acesso à tela de Seus Amigos, onde o usuário pode visualizar outros usuários que possui adicionados como amigo e também buscar outros usuários. O ícone de bússola dá acesso à tela Suas Viagens, onde o usuário pode visualizar seus pedidos ou oferecimentos de carona em aberto. O ícone de carro dá acesso à tela Oferecer Carona, onde o usuário pode criar uma viagem oferecida. O ícone de lupa abre a tela Procurar Carona, que, por padrão, é a tela inicial que aparece ao realizar login. Por fim, o ícone de perfil dá acesso à tela Perfil de Usuário.

Figura 30 - Tela de Procurar Carona



Fonte: Elaboração própria, 2025.

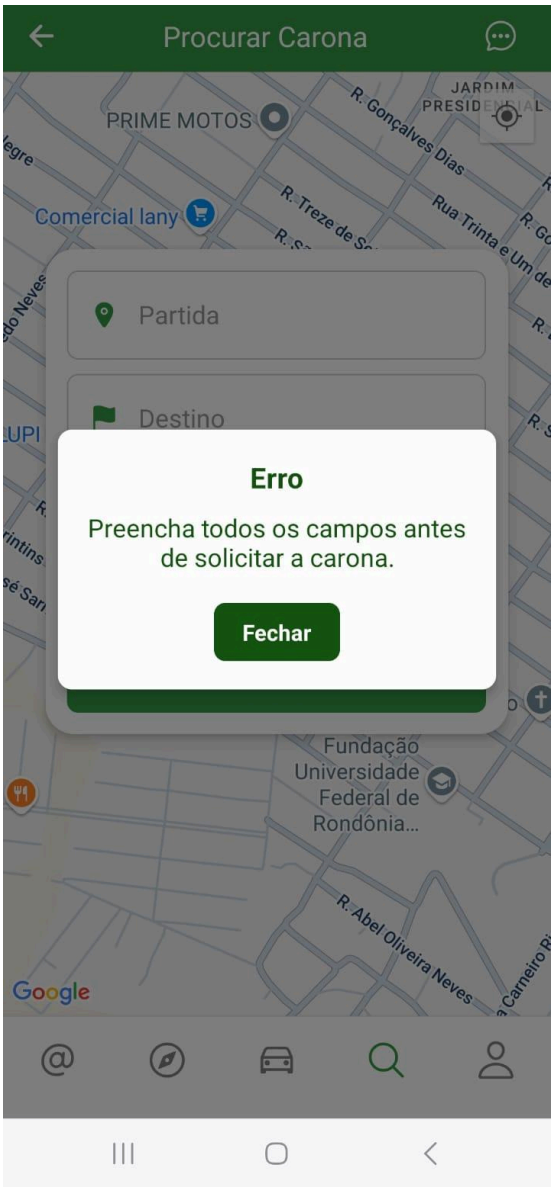
Ao ser direcionado para esta tela, o usuário recebe uma solicitação do aplicativo para acessar sua localização. Isso ocorre devido às bibliotecas react-native-maps e expo-location:

enquanto a react-native-maps fornece um componente de mapa que utiliza o Google Maps em aparelhos android e o Apple Maps ou Google Maps em aparelhos iOS, a expo-location fornece o acesso à leitura de informações de geolocalização do dispositivo. Ao utilizar ambas, foi possível definir um componente de mapa com a localização atual do usuário como plano de fundo para a tela de Procurar Carona.

No centro da tela, há um card com campos para que o usuário possa inserir seu endereço de partida, endereço de destino e data e hora da carona, além de mostrar um valor simbólico sugerido com base no trajeto.

Também é possível visualizar o menu inferior, que serve como um componente voltado à navegação dentro do aplicativo. Ao clicar no ícone da lupa, independentemente da tela em que estiver no momento, o usuário é levado à tela de Procurar Carona. O mesmo ocorre com os demais botões e suas telas correspondentes.

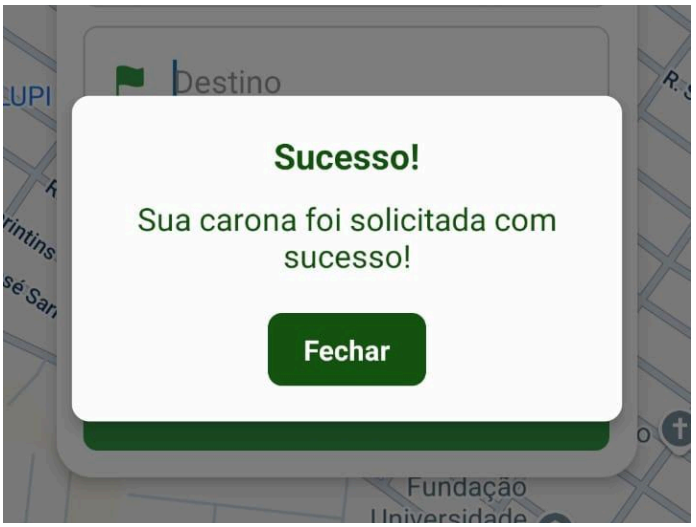
Figura 31 - Alerta de erro: campos obrigatórios



Fonte: Elaboração própria, 2025.

Caso o usuário tente criar uma solicitação de carona sem inserir informações nos campos obrigatórios, o aplicativo mostra uma mensagem de erro para informá-lo.

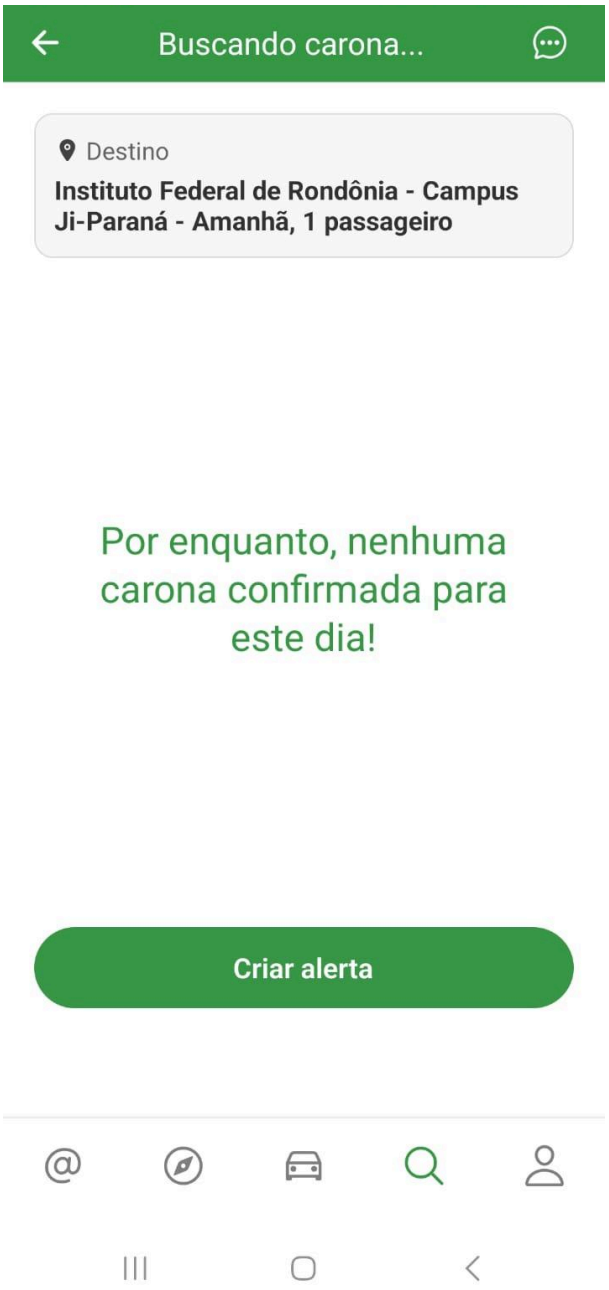
Figura 32 - Mensagem de confirmação: solicitação de carona criada com sucesso



Fonte: Elaboração própria, 2025.

Ao preencher todas as informações corretamente, a solicitação de carona é criada e o registro é inserido no banco de dados, tanto na tabela de carona solicitada como na tabela intermediária muitos-para-muitos, que registra o id da carona e também o id do usuário logado.

Figura 33 - Tela de Buscando Carona



Fonte: Elaboração própria, 2025.

Após criar sua solicitação de carona, o usuário é levado à tela de Buscando Carona, onde poderá visualizar se existem viagens oferecidas para o destino e data escolhidos. Caso não haja registros, ele poderá criar um alerta para receber uma notificação quando outro usuário aceitar sua solicitação de carona e, caso haja, poderá aceitar aquela carona.

Figura 34 - Tela de Perfil de Usuário

Fonte: Elaboração própria, 2025.

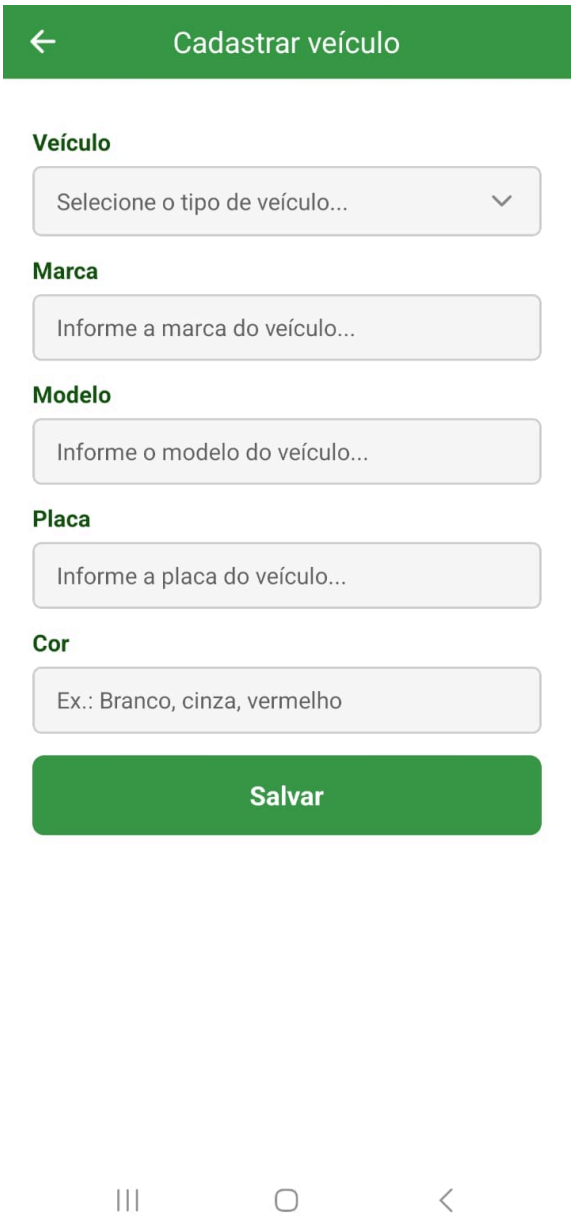
Ao clicar no ícone de perfil no menu inferior, o usuário é levado para a tela de Perfil de Usuário, onde poderá realizar diversas configurações para customizá-lo conforme sua vontade. Ao clicar no botão de Sair, uma mensagem de alerta é mostrada.

Figura 35 - Mensagem de alerta: fazer logout

Fonte: Elaboração própria, 2025.

A mensagem de alerta possui dois botões, um botão vermelho para Cancelar e um botão verde para Confirmar. Caso clique em Confirmar, o usuário é deslogado de sua conta e o aplicativo o redireciona para a Tela Inicial. Caso clique em Cancelar, a operação é cancelada e o aplicativo fecha o alerta, permanecendo na tela atual.

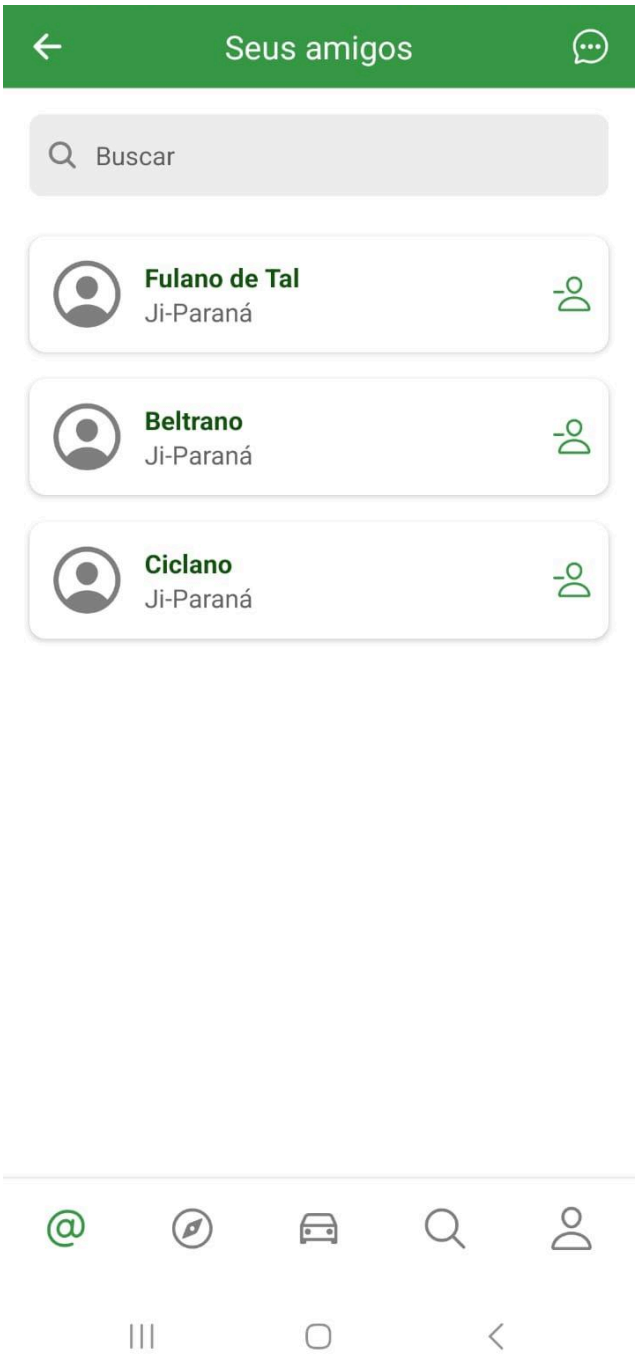
Figura 36 - Tela de Cadastro de Veículo



Fonte: Elaboração própria, 2025.

Para cadastrar um novo veículo, basta clicar no botão Adicionar veículo disponível na tela de Perfil de Usuário. Ao clicar neste botão, o aplicativo abre a tela Cadastrar Veículo, onde são solicitadas algumas informações do veículo como o tipo (por padrão, carro ou moto como consta na definição pela API), marca, modelo, placa e cor. É necessário ter pelo menos um veículo cadastrado antes de poder oferecer uma carona dentro do aplicativo.

Figura 37 - Tela de Seus Amigos



Fonte: Elaboração própria, 2025.

Ao clicar no ícone de arroba, o usuário poderá acessar a tela Seus Amigos. Nela, pode visualizar os usuários adicionados, ou seja, que aceitaram sua solicitação de amizade, e também fazer uma busca por novos usuários.

Durante o desenvolvimento deste aplicativo, que tem como objetivo o compartilhamento de caronas, ocorreram algumas mudanças em seu escopo inicialmente projetado. A princípio, o propósito do aplicativo era facilitar o compartilhamento de caronas apenas entre alunos do IFRO;

no entanto, posteriormente surgiu a ideia de abranger o público geral como público alvo, ao invés de limitar o funcionamento somente aos alunos. Dessa forma, o aplicativo passou a ser projetado tendo em vista as características deste público, priorizando a experiência do usuário.

5. CONCLUSÃO

A realização deste trabalho teve como objetivo principal o desenvolvimento de um aplicativo mobile voltado para o compartilhamento de caronas solidárias, inspirado na prática do carpooling. A proposta inicial, que visava atender exclusivamente à comunidade discente do Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), foi reformulada ao longo do projeto, ampliando-se para atender ao público em geral. Essa mudança de escopo contribuiu para um projeto mais inclusivo e com maior potencial de impacto social.

Com base nas tecnologias adotadas, essas sendo: React Native com Expo para o aplicativo mobile, Node.js com Express e Sequelize para a API REST, além do MySQL como banco de dados relacional, foi possível desenvolver uma aplicação robusta, modular e escalável. O uso de TypeScript em todas as camadas também contribuiu para uma maior segurança e padronização do código. O banco de dados foi construído de forma relacional, com tabelas bem definidas, além da criação de views para melhorar a segurança e integridade dos dados, como no caso do histórico de viagens.

A API REST, documentada com o auxílio do Swagger, desempenhou um papel essencial como intermediadora entre o frontend e o banco de dados, estruturando endpoints claros e bem definidos para cada funcionalidade do sistema. Por sua vez, o aplicativo mobile foi projetado com base no protótipo desenvolvido no Figma, garantindo um produto final visualmente coerente e funcional, levando em consideração a experiência do usuário em todas as etapas ao emitir feedbacks conforme o uso.

Dessa forma, os objetivos geral e específicos foram alcançados, evidenciando o êxito da aplicação proposta. O sistema desenvolvido apresenta-se como uma alternativa que, caso implementada, poderia ser viável para auxiliar na mobilidade urbana, especialmente no contexto do estado de Rondônia, contribuindo com a sustentabilidade e a otimização de recursos através do compartilhamento de caronas.

Espera-se que este projeto sirva como base para futuras melhorias e expansões, podendo contar com a adição de novas funcionalidades, como a possibilidade de realizar cadastro e login via conta Google, um sistema básico de troca de mensagens entre usuários e integração com meios de pagamento digitais, ampliando ainda mais o seu alcance e utilidade à população.

REFERÊNCIAS

ALURA. **React Native: o que é e tudo sobre o Framework.** 2023. Disponível em: https://www.alura.com.br/artigos/react-native?srsltid=AfmBOoryfO3h57ppKjRy_4ZXX09nS8-P_46ZwESxKCjKCGy0IOwIKeVv. Acesso em 01 nov. 2024.

AMAZON WEB SERVICES. **O que é uma API?** AWS, 2025. Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 18 jul. 2025.

BARSOTI, Nathan; GIBERTONI, Daniela. IMPACTO QUE O SEQUELIZE TRAZ PARA O DESENVOLVIMENTO DE UMA API CONSTRUÍDA EM NODE.JS COM EXPRESS.JS. **Revista Interface Tecnológica**, Taquaritinga, SP, v. 17, n. 2, p. 231–243, 2020. DOI: 10.31510/infa.v17i2.964. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/964. Acesso em: 18 jul. 2025.

CUNNINGHAM, Ward. **MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT,** 2001. Disponível em: https://agilemanifesto.org/. Acesso em 19 mar. 2025.

CUSTÓDIO, Fernando. **Prototipação de software: Importância no desenvolvimento de aplicativos**. FWC Tecnologia, 2023. Disponível em: https://fwctecnologia.com/blog/post/importancia-prototipacao-software. Acesso em 21 mar. 2025.

ELMASRI, Ramez. NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. Pearson Addison Wesley. 6a Edição, 2011.

EXPO. **Expo Documentation**. Disponível em: https://docs.expo.dev/. Acesso em 15 mar. 2025.

EXPRESS.JS. **Express – Documentação oficial**. Disponível em: https://expressjs.com/. Acesso em: 18 jul. 2025.

FALCÃO, Filipe Dourado. **Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native**. TCC (Graduação em Sistemas e Mídias Digitais) – Universidade Federal do Ceará. Fortaleza - CE, 2022. Disponível em: https://repositorio.ufc.br/handle/riufc/69029. Acesso em 17 mar. 2025.

FIGMA. **O que é o Figma?** Disponível em: https://www.figma.com/pt-br/prototyping/. Acesso em 17 mar. 2025.

FLANAGAN, David. **JavaScript: O guia definitivo.** Editora Bookman. 6a Edição, 2013.

GIL, Antonio Carlos. Métodos e Técnicas de Pesquisa Social. São Paulo, SP: Atlas, 2008. p.75-88.

IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Frota de veículos**. IBGE, 2024. Disponível em: https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?ano=2024. Acesso em: 21 de março de 2025.

INSTITUTO DE ENERGIA E MEIO AMBIENTE. **Dia do Meio Ambiente: veículos são os principais culpados pela poluição do ar em centros urbanos.** Disponível em: https://energiaeambiente.org.br/dia-do-meio-ambiente-veiculos-sao-os-principais-culpados-pela-poluicao-do-ar-em-centros-urbanos-2-20190101. Acesso em 16 nov. de 2024.

KLOH, Gustavo Minuzzi. HABITZREITER, Taiane Tais. BARCAROLI, Velcir. **Peculiaridades de um banco MySQL.** nº 11. Revista Conexão, 2024. Disponível em: https://revistas.uceff.edu.br/conexao/article/view/531/476. Acesso em 17 mar. 2025.

LACERDA, Guilherme Mangueira. **A influência do uso de aplicativo de carona na mobilidade urbana: um estudo de caso do Blablacar.** TCC (Bacharelado em Engenharia Civil) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. Cajazeiras - PB, 2023. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/3630. Acesso em 31 out. 2024.

MELO, Lidiane da Silva. **Projeto e prototipação do virtual biblioteca utilizando a ferramenta figma**. TCC (Licenciatura em Computação) – Universidade Federal Rural do Semi-Árido, 2024. Disponível em: https://repositorio.ufersa.edu.br/items/ac8bfb89-deda-4584-a78b-da9df610743d. Acesso em 17 mar. 2025.

MIRO. **Modelo para Diagrama UML**. Disponível em: https://miro.com/pt/modelos/diagramasuml/. Acesso em 18 mar. 2025.

MOZILLA. **O que é JavaScript?** MDN Web Docs, 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Core/Scripting/What_is_JavaScript. Acesso em 22 mar. 2025.

NODE.JS. **Introduction to Node.js.** Node.js Documentation. 2025. Disponível em: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. Acesso em: 18 jul. 2025.

RAMAKRISHNAN, Raghu e GEHRKE, Johannes. **Sistemas de Gerenciamento de Banco de Dados**. Editora Artmed. 3a Edição, 2008.

RED HAT. **O que são APIs (Application Programming Interfaces)?** Red Hat, 2025. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 18 jul. 2025.

SARMENTO, D. **Ordem Constitucional Econômica, Liberdade e Transporte Individual de Passageiros: o " caso Uber"**. Rio de Janeiro,[s.n], 10 jul. 2015. Disponível em: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.conjur.com.br/dl/pa/paracer-legalidade-uber.pdf&ved=2ahUKEwjllfz1hJKMAxWhDrkGHfY2AqIQFnoECCEQAQ&usg=AOvVaw2NZqVGbEBHoP6erm9DbpdZ . Acesso em: 17 mar. 2025.

SATO, P. M. **ORM - Object Relational Mapping**. DEVMEDIA, 2013. Disponível em: <https://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158>. Acesso em: 18 jul. 2025.

SCHWABER, Ken. SUTHERLAND, Jeff. **Guia do Scrum.** 2013. Disponível em: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf&ved=2ahUKEwiVi4Gg05SMAxX2ErkGHWK8AukQFnoECCAQAQ&usg=AOvVaw2da_5liRyb0pv_VlYt2Ehh. Acesso em 17 mar. 2025.

SEQUELIZE. Sequelize – ORM for Node.js. Disponível em: https://sequelize.org/. Acesso em: 18 jul. 2025.

SILBERSCHATZ, Abraham, KORTH, Henry F. e SUDARSHAN, S. **Sistema de Banco de Dados.** Editora Campus. 5a Edição, 2006.

SOUSA, Lucas Papa de. **Análise da implementação sustentável de veículos elétricos no Brasil.** Orientador: André Antonio Zanatto. 2024. Artigo Científico, apresentado como Trabalho de Conclusão de Curso (Ensino Técnico em Logística) – ETECAMP: ETEC de Campo Limpo Paulista, Campo Limpo Paulista, 2024. Disponível em: https://ric.cps.sp.gov.br/handle/123456789/30271. Acesso em 21 mar. 2025.

SWAGGER. **Swagger UI – Ferramenta de documentação de APIs.** Disponível em: https://swagger.io/tools/swagger-ui/. Acesso em: 18 jul. 2025.

TYPESCRIPT. **TypeScript in 5 Minutes.** TypeScript Documentation, 2025. Disponível em: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Acesso em: 18 jul. 2025.

APÊNDICE A: Projeto do Software

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DE RONDÔNIA – CAMPUS JI-PARANÁ

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

APLICATIVO DE COMPARTILHAMENTO DE CARONAS ENTRE OS ALUNOS DO IFRO

Maria Júlia Souza de Albuquerque Lins

Ji-Paraná - RO, 2025

Maria Júlia Souza de Albuquerque Lins

APLICATIVO DE COMPARTILHAMENTO DE CARONAS ENTRE OS ALUNOS DO IFRO

Projeto de Conclusão de Curso SUPERIOR EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS, como parte dos requisitos necessários para a aprovação no curso Tecnólogo em Análise e Desenvolvimento de Sistemas. Área de Concentração: Ciências Exatas e da Terra.

Orientador(a): Prof. Me. Reinaldo Lima Pereira

Ji-Paraná - RO, 2025

LISTA DE FIGURAS

SUMÁRIO

1. DOCUMENTO DE ESPECIFICAÇÃO DE SOFTWARE

1.1. Produto

Trata-se do desenvolvimento de um aplicativo mobile para o compartilhamento de caronas solidárias entre os alunos do IFRO, o qual possibilitará que os alunos possam tanto oferecer como solicitar caronas uns aos outros. Sendo desenvolvido em React Native, uma linguagem que permite o desenvolvimento *cross-platform*, o aplicativo estará disponível tanto para aparelhos iOS como Android, bastando apenas estar conectado à internet.

1.2. Aspectos Técnicos do Produto

O aplicativo será implementado tendo como base a linguagem de programação JavaScript, uma vez que o framework escolhido para desenvolver o front-end foi o React Native. O editor de código será o Visual Studio Code e o banco de dados será construído no SGBD (Sistema de Gerenciamento de Banco de Dados) MySQL Workbench. Também será utilizada a ferramenta Astah UML para a modelagem de diagramas UML e a plataforma Figma para construção do protótipo do aplicativo.

2. DOCUMENTO DE REQUISITOS

Após a fase de levantamento de requisitos, foram identificados e coletados os seguintes requisitos para o software:

2.1. Cadastrar Usuário

Requisito Funcional			
Nome:	Cadastrar Usuário	**Código:**	F1
Descrição:	O sistema deve registrar todas as informações necessárias para cadastrar uma nova conta de usuário.		
Estimativa de Esforço: 5h		**Prioridade:** 100 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A função deve registrar os seguintes dados: Nome e Sobrenome, Telefone, E-mail e Senha.	Especificação
1.2	Deve possibilitar o cadastro por meio de uma conta Google já existente.	Usabilidade
1.3	Haverá validações para impedir a duplicidade de dados.	Confiabilidade
1.4	O sistema será uma aplicação para dispositivos móveis, funcionando em qualquer dispositivo com acesso à internet, tanto iOS como Android.	Portabilidade
1.5	Após preenchimento dos dados de cadastro, a solicitação será analisada por um outro usuário para verificar se o usuário possui vínculo com o IFRO de fato.	Segurança
1.6	A função deve verificar se os dados de Telefone e E-mail informados são válidos. Caso sejam inválidos, aparecerá uma mensagem de erro solicitando correção dos dados.	Segurança
1.7	A tela possuirá branco como a cor predominante e tons de verde como secundária. Possuirá um botão para Continuar com Google, um botão Continuar na cor #379846 e labels de especificação na cor #155310.	Interface

2.2. Validar Vínculo Estudantil

Requisito Funcional			
Nome:	Validar Vínculo Estudantil	**Código:**	F2
Descrição:	O administrador deve analisar o cadastro de um usuário e validar ou anular o mesmo.		
Estimativa de Esforço: 6h		**Prioridade:** 100 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**

1.1	O usuário só poderá oferecer e solicitar caronas caso seu vínculo com o Instituto seja confirmado, ou seja, quando aceitar pedidos de amizade ou quando seus pedidos de amizade forem aceitos por outros alunos.	Segurança
1.2	A validação será feita pelos próprios usuários, podendo tanto aceitar um usuário em sua rede para possibilitar a troca de viagens como bloquear usuários para não receber suas solicitações.	Segurança

2.3. Fazer Login

Requisito Funcional			
Nome:	Fazer Login	**Código:**	F3
Descrição:	O sistema deve permitir que um usuário entre em sua conta.		
Estimativa de Esforço: 5h		**Prioridade:** 100 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A tela de Login terá campos solicitando os dados de E-mail e Senha previamente cadastrados pelo usuário.	Especificação
1.2	O Login também poderá ser realizado por meio de uma conta Google já existente.	Usabilidade
1.3	A função deve verificar se os dados informados são válidos. Caso sejam inválidos ou incorretos, aparecerá uma mensagem de erro solicitando correção dos dados.	Segurança
1.4	A função terá um label escrito “Esqueci minha senha” que, quando for clicado, irá encaminhar para o usuário instruções para modificar sua senha de acesso via e-mail.	Segurança
1.5	A tela possuirá branco como a cor predominante e tons de verde como secundária. Possuirá um botão para fazer login com Google, um botão Entrar na cor #379846 e labels de especificação na cor #155310.	Interface

2.4. Solicitar Carona

Requisito Funcional			
Nome:	Solicitar Carona	**Código:**	F4
Descrição:	O sistema deve registrar todas as informações necessárias para que o usuário possa solicitar uma carona.		
Estimativa de Esforço: 7h		**Prioridade:** 100 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Lupa localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá um mapa com sua localização aproximada de fundo, um card com campos para receber as informações do usuário, um botão Pedir carona na cor	Interface

	#379846 e labels de especificação na mesma cor. Possuirá o título “Procurar carona” no topo da tela, juntamente com um botão para visualizar suas Mensagens, ambos na cor branca.	
1.3	Na primeira tela deste fluxo, serão solicitados: endereço de partida, endereço de destino e data e hora da viagem. O usuário poderá selecionar mais de um dia no calendário que irá aparecer quando clicar no campo Data. Também será calculado um valor sugerido, caso o usuário deseje contribuir com os custos da viagem.	Especificação
1.4	Conforme o usuário estiver digitando os endereços de Partida e Destino, o aplicativo irá sugerir o endereço completo.	Usabilidade
1.5	O usuário poderá também pesquisar o endereço diretamente pelo CEP através de um sistema de validação.	Confiabilidade
1.6	Após preencher as informações, o usuário é levado para a próxima tela, na qual a aplicação apresentará as caronas com aquele destino (o respectivo Campus do IFRO) para o período selecionado. Caso haja alguma carona criada, será disponibilizada em formato de card; caso não haja, aparecerá um texto informando que não há. O usuário poderá criar um alerta para sua solicitação, que irá aparecer para seus amigos adicionados como notificação.	Especificação
1.7	A tela possuirá o título “Buscando carona…” na cor branca, um card na parte superior que apresentará o endereço de destino inserido pelo usuário e um botão para Criar um alerta para sua solicitação de carona.	Interface

2.5. Visualizar Pedido de Carona em Andamento

Requisito Funcional			
Nome:	Visualizar Pedido de Carona em Andamento	**Código:**	F5
Descrição:	O sistema deve mostrar os pedidos de carona em aberto.		
Estimativa de Esforço: 5h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Bússola localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá a cor branca como predominante, um card com campos para mostrar as informações de cada carona ao usuário. Possuirá o título “Suas viagens” no topo da tela, juntamente com um botão para visualizar suas Mensagens, ambos na cor branca. Terá também botões em formato de ícones para Editar ou Excluir a solicitação.	Interface
1.3	Nesta tela, serão mostrados: endereço de partida, endereço de destino e data e hora da viagem. Também será mostrado o valor sugerido e o Status do pedido de carona, este podendo ser “Aguardando aceite” ou então “Carona aceita!”. Neste caso, será também mostrado o nome da pessoa que aceitou a viagem.	Especificação

2.6. Editar Pedido de Carona

Requisito Funcional			
Nome:	Editar Pedido de Carona	**Código:**	F6
Descrição:	O usuário poderá alterar informações em sua solicitação de carona.		
Estimativa de Esforço: 6h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no botão de Editar no card da carona que o usuário deseja modificar.	Usabilidade
1.2	A tela terá a cor branca como predominante, o tom de verde #379846 como secundária e campos editáveis para que o usuário possa modificar as informações de cada carona. Possuirá o título “Editar pedido de carona” no topo da tela na cor branca. Terá também um botão Salvar na cor #379846.	Interface
1.3	Nesta tela, o usuário poderá modificar o endereço de partida, o endereço de destino e data e hora da viagem.	Especificação
1.4	Ao clicar no botão de Salvar, o sistema irá mostrar um pop-up perguntando se o usuário deseja confirmar as alterações. Caso sim, mostrará em seguida uma mensagem de confirmação.	Usabilidade

2.7. Cancelar Pedido de Carona

Requisito Funcional			
Nome:	Cancelar Pedido de Carona	**Código:**	F7
Descrição:	O usuário poderá excluir a sua solicitação de carona.		
Estimativa de Esforço: 6h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no botão de Excluir no card da carona que o usuário deseja modificar.	Usabilidade
1.2	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja excluir sua solicitação de carona. Caso sim, mostrará em seguida uma mensagem de confirmação da exclusão.	Usabilidade
1.3	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Excluir, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface

2.8. Oferecer Carona

Requisito Funcional			
Nome:	Oferecer Carona	**Código:**	F8
Descrição:	O sistema deve registrar todas as informações necessárias para que o usuário possa oferecer uma carona.		
Estimativa de Esforço: 7h		**Prioridade:** 100 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**

1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Carro localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá a cor branca como predominante, o tom de verde #379846 como secundária e campos de busca editáveis com placeholders. Possuirá o título “Oferecer carona” no topo da tela na cor branca. Terá também um botão Continuar na cor #379846.	Interface
1.3	Na primeira tela deste fluxo, serão solicitados os endereços de saída e de destino.	Especificação
1.4	Conforme o usuário estiver digitando os endereços de Partida e Destino, o aplicativo irá sugerir o endereço completo.	Usabilidade
1.5	O usuário poderá também pesquisar o endereço diretamente pelo CEP através de um sistema de validação.	Confiabilidade
1.6	Após preencher as informações, o usuário é levado para a próxima tela, na qual irá solicitar que escolha o seu Veículo.	Especificação
1.7	A tela possuirá uma caixa de seleção na cor cinza que, ao ser clicada, mostrará todos os veículos cadastrados para que o usuário selecione. Terá também um botão Continuar na cor #379846.	Interface
1.8	Um usuário só poderá oferecer uma carona caso tenha cadastrado um veículo previamente. Caso não tenha, deverá primeiramente cadastrar um em seu Perfil ou então clicando no botão “Cadastrar novo veículo”.	Segurança
1.9	Após preencher as informações, o usuário é levado para a próxima tela, na qual irá definir quantos passageiros poderá levar em seu veículo.	Especificação
1.10	A tela mostrará, por padrão, o número 1. O usuário pode adicionar ou diminuir o número de passageiros ao clicar nos botões de mais e menos. Terá também um botão Continuar na cor #379846.	Interface
1.11	Após preencher as informações, o usuário é levado para a próxima tela, na qual irá definir quais os dias em que fará o trajeto.	Especificação
1.12	A tela mostrará um campo de seleção que, ao ser clicado, abrirá um calendário. O usuário pode definir os dias ao clicar neles, podendo adicionar mais de um, e os dias selecionados serão destacados. Terá também um botão Continuar na cor #379846.	Interface
1.13	Após preencher as informações, o usuário é levado para a próxima tela, na qual irá definir o horário em que fará o trajeto.	Especificação
1.14	A tela mostrará dois campos, um para selecionar as horas e o outro para selecionar os minutos. Os valores poderão ser selecionados ou digitados. Terá também um botão Continuar na cor #379846.	Interface
1.15	Após preencher as informações, o usuário é levado para a próxima tela, na qual poderá ou não convidar amigos com antecedência para aquela carona.	Especificação
1.16	A tela terá um campo para que o usuário possa pesquisar seus amigos adicionados por nome. Os resultados da busca aparecerão conforme o usuário digita.	Usabilidade
1.17	Os resultados da busca aparecerão em formato de card, com o nome da pessoa, sua cidade, foto de perfil e um ícone para adicionar. Terá também um botão Oferecer carona na cor #379846.	Interface
1.18	Após preencher as informações, por fim, o usuário é levado para a próxima tela, na qual poderá visualizar o resumo da viagem que criou. Nela, poderá	Especificação

	consultar o endereço de partida, de destino, a quantidade de passageiros, a data e a hora da viagem.	
1.19	A tela terá a cor branca como predominante e o tom de verde #379846 como secundária. Possuirá o título “Resumo da viagem” no topo da tela na cor branca. Terá também um botão Alterar na cor #379846 e um botão Salvar na cor #155310.	Interface
1.20	O aplicativo mostrará um pop-up perguntando se o usuário deseja salvar a carona. Caso sim, mostrará em seguida uma mensagem de confirmação.	Usabilidade
1.21	O usuário também pode oferecer uma carona ao clicar na notificação de solicitação de carona criada por algum amigo.	Usabilidade

2.9. Visualizar Carona

Requisito Funcional			
Nome:	Visualizar Carona	**Código:**	F9
Descrição:	O sistema deve mostrar as caronas em aberto.		
Estimativa de Esforço: 5h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Bússola localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá a cor branca como predominante, um card com campos para mostrar as informações de cada carona ao usuário. Possuirá o título “Suas viagens” no topo da tela, juntamente com um botão para visualizar suas Mensagens, ambos na cor branca. Terá também botões em formato de ícones para Editar ou Excluir a carona, além de um botão para Iniciar o trajeto na cor #379846.	Interface
1.3	Nesta tela, serão mostrados: endereço de partida, endereço de destino e data e hora da viagem.	Especificação

2.10. Editar Carona

Requisito Funcional			
Nome:	Editar Carona	**Código:**	F10
Descrição:	O usuário poderá alterar informações na carona que está oferecendo.		
Estimativa de Esforço: 6h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no botão de Editar no card da carona que o usuário deseja modificar.	Usabilidade
1.2	A tela terá a cor branca como predominante, o tom de verde #379846 como secundária e campos editáveis para que o usuário possa modificar as	Interface

	informações de cada carona. Possuirá o título “Editar carona” no topo da tela na cor branca. Terá também um botão Salvar na cor #379846.	
1.3	Nesta tela, o usuário poderá modificar o endereço de partida, o endereço de destino, a quantidade de passageiros e a data e hora da viagem.	Especificação
1.4	Ao clicar no botão de Salvar, o sistema irá mostrar um pop-up perguntando se o usuário deseja confirmar as alterações. Caso sim, mostrará em seguida uma mensagem de confirmação.	Usabilidade

2.11. Cancelar Carona

Requisito Funcional			
Nome:	Cancelar Carona	**Código:**	F11
Descrição:	O usuário poderá excluir a carona que está oferecendo.		
Estimativa de Esforço: 6h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no botão de Excluir no card da carona que o usuário deseja modificar.	Usabilidade
1.2	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja excluir a carona. Caso sim, mostrará em seguida uma mensagem de confirmação da exclusão.	Usabilidade
1.3	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Excluir, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface

2.12. Cadastrar Veículo

Requisito Funcional			
Nome:	Cadastrar Veículo	**Código:**	F12
Descrição:	O sistema deve registrar todas as informações necessárias para cadastrar um novo veículo.		
Estimativa de Esforço: 5h		**Prioridade:** 90 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A função deve registrar os seguintes dados: Veículo, Marca, Modelo, Placa e Cor.	Especificação
1.2	A tela terá a cor branca como predominante, o tom de verde #379846 como secundária e campos editáveis para que o usuário possa inserir as informações do veículo. Possuirá o título “Cadastrar veículo” no topo da tela na cor branca. Terá também um botão Salvar na cor #379846.	Interface
1.3	Para selecionar um veículo, o usuário irá clicar no dropdown Veículo e selecionar uma das duas opções padrão: Carro ou Moto.	Usabilidade

1.4	Ao clicar no botão de Salvar, o sistema irá mostrar um pop-up perguntando se o usuário deseja confirmar as alterações. Caso sim, mostrará em seguida uma mensagem de confirmação.	Usabilidade

2.13. Editar Perfil

Requisito Funcional			
Nome:	Editar Perfil	**Código:**	F13
Descrição:	O usuário poderá visualizar e modificar seu perfil do aplicativo.		
Estimativa de Esforço: 7h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Perfil localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá a cor branca como predominante e o tom de verde #379846 como secundária. Títulos importantes estarão na cor #155310. Possuirá também um botão Sair na cor #FF4F4F.	Interface
1.3	Na parte superior há duas seções: Seu perfil e Sua conta. A tela mostrará a foto de perfil do usuário em formato de círculo, terá uma seção para adicionar uma minibiografia e também uma seção que mostra os veículos cadastrados. Permite acessar o histórico de caronas e alterar a foto de perfil.	Especificação
1.4	O usuário também pode cadastrar um veículo nessa seção.	Usabilidade
1.5	A seção Sua conta mostrará as informações de cadastro, como o Nome completo, Telefone e E-mail previamente registrados. O usuário pode alterar seus dados pessoais ou sua senha nessa seção.	Especificação
1.6	Para alterar a senha, o usuário deve clicar no label “Alterar senha” que, quando for clicado, irá encaminhar para o usuário instruções para modificar sua senha de acesso via e-mail.	Segurança

2.14. Visualizar Histórico de Caronas

Requisito Funcional			
Nome:	Visualizar Histórico de Caronas	**Código:**	F14
Descrição:	O sistema deve mostrar o histórico das caronas solicitadas e realizadas.		
Estimativa de Esforço: 5h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no label “Histórico de caronas” em Seu perfil.	Usabilidade

1.2	A tela terá a cor branca como predominante, um card com campos para mostrar as informações de cada carona ao usuário. Possuirá o título “Histórico de caronas” no topo da tela na cor branca.	Interface
1.3	Nesta tela, serão mostrados: ID da viagem, endereço de partida, endereço de destino, quantidade de passageiros e data e hora da viagem.	Especificação

2.15. Buscar Usuário

Requisito Funcional			
Nome:	Buscar Usuário	**Código:**	F15
Descrição:	O usuário poderá buscar outros usuários pelo nome, visualizar seu perfil e adicionar como amigo.		
Estimativa de Esforço: 8h		**Prioridade:** 90 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Arroba localizado no menu inferior da aplicação. Todos os ícones são da cor cinza #5F6368, porém, quando o usuário clicar em um deles para acessar sua funcionalidade, o mesmo mudará para a cor #379846.	Usabilidade
1.2	A tela terá a cor branca como predominante e um campo de busca na cor cinza. Possuirá o título “Seus amigos” no topo da tela, juntamente com um botão para visualizar suas Mensagens, ambos na cor branca. Os amigos já adicionados aparecerão em um conjunto de cards, um abaixo do outro, com botões para Excluir.	Interface
1.3	Conforme o usuário estiver digitando o nome da pessoa que está buscando, o aplicativo irá mostrar os resultados com base na busca.	Usabilidade
1.4	Para visualizar o perfil de um usuário, basta clicar no card correspondente à busca.	Usabilidade
1.5	Ao visualizar o perfil de um usuário, é possível visualizar a foto de perfil, o nome completo, a minibiografia e os veículos cadastrados por ele.	Especificação
1.6	Ao clicar em um card de um amigo já adicionado, tela terá a cor branca como predominante. Possuirá o título “Informações do perfil” no topo da tela, juntamente com os botões Bloquear usuário e Excluir amigo.	Interface
1.7	Para bloquear um usuário, basta clicar no botão de Bloquear enquanto estiver visualizando o perfil daquela pessoa.	Usabilidade
1.8	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja bloquear o usuário. Caso sim, mostrará em seguida uma mensagem de confirmação do bloqueio.	Usabilidade
1.9	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Excluir, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface

1.10	Para excluir um amigo, basta clicar no botão de Excluir enquanto estiver visualizando o perfil daquela pessoa.	Usabilidade
1.11	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja excluir o usuário como amigo. Caso sim, mostrará em seguida uma mensagem de confirmação da exclusão.	Usabilidade
1.12	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Excluir, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface
1.13	Para adicionar um amigo, é preciso clicar no card de seu perfil após o resultado da busca para abrir a tela de Informações do perfil ou então clicar no botão de adicionar no card.	Usabilidade
1.14	Ao clicar para adicionar um amigo, o aplicativo mostrará um pop-up informando que a solicitação de amizade foi enviada com sucesso.	Usabilidade

2.16. Visualizar chat de mensagens

Requisito Funcional			
Nome:	Visualizar chat de mensagens	**Código:**	F16
Descrição:	O usuário poderá visualizar e gerenciar suas notificações e mensagens.		
Estimativa de Esforço: 9h		**Prioridade:** 80 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no ícone de Mensagens localizado nas diversas telas da aplicação.	Usabilidade
1.2	O ícone possui formato de balão de texto e receberá um ponto vermelho quando houver novas mensagens. A tela terá a cor branca como predominante e o tom de verde #379846 como secundária.	Interface
1.3	Na parte superior há duas seções: Notificações e Conversas. A tela mostrará as notificações em formato de card, informando pedidos de carona abertos por amigos adicionados e confirmações de carona.	Especificação
1.4	Cada card de notificação possuirá um botão com um ícone de “+” para mostrar mais detalhes e outro com um ícone de lixeira para excluir.	Interface
1.5	O usuário também pode oferecer caronas por meio dessa seção.	Usabilidade
1.6	A seção Conversas mostrará os chats ativos do usuário, possibilitando que a última mensagem enviada ou recebida seja visualizada. Ao clicar na conversa, é possível ler as últimas mensagens e escrever mensagens novas. O usuário também pode excluir conversas.	Especificação
1.7	Conversas sem confirmação de leitura aparecem destacadas na cor verde e, ao serem lidas, mudam para uma cor cinza.	Interface

1.8	Para excluir uma mensagem ou notificação, basta clicar no botão de Excluir com o ícone de lixeira.	Usabilidade
1.9	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja excluir o chat ou a notificação. Caso sim, mostrará em seguida uma mensagem de confirmação da exclusão.	Usabilidade
1.10	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Excluir, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface

2.17. Fazer Logout

Requisito Funcional			
Nome:	Fazer Logout	**Código:**	F17
Descrição:	O sistema deve permitir que o usuário saia de sua conta.		
Estimativa de Esforço: 5h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no botão “Sair” em Seu perfil.	Usabilidade
1.2	O aplicativo mostrará um pop-up perguntando se o usuário realmente deseja sair de sua conta. Caso sim, a conta será deslogada e o usuário será redirecionado à tela inicial do aplicativo.	Usabilidade
1.3	O pop-up possuirá um botão Cancelar vermelho, na cor #7D1616, e um Confirmar, na cor #155310. Possuirá também um ícone de X que também cancelaria a solicitação.	Interface
1.4	Ao sair e entrar em sua conta, as informações e solicitações registradas permanecerão no aplicativo, sem risco de perda dos dados.	Confiabilidade

2.18. Visualizar Usuários Bloqueados

Requisito Funcional			
Nome:	Visualizar Usuários Bloqueados	**Código:**	F18
Descrição:	O sistema deve permitir que o usuário visualize os usuários que bloqueou anteriormente.		
Estimativa de Esforço: 5h		**Prioridade:** 70 pontos	

Requisitos Não funcionais		
ID NF	**Descrição**	**Categoria**
1.1	A funcionalidade poderá ser acessada ao clicar no label “Usuários bloqueados” em Seu perfil.	Usabilidade
1.2	A tela terá a cor branca como predominante, um card com campos para mostrar as informações de cada usuário. Possuirá o título “Usuários bloqueados” no topo da tela na cor branca.	Interface

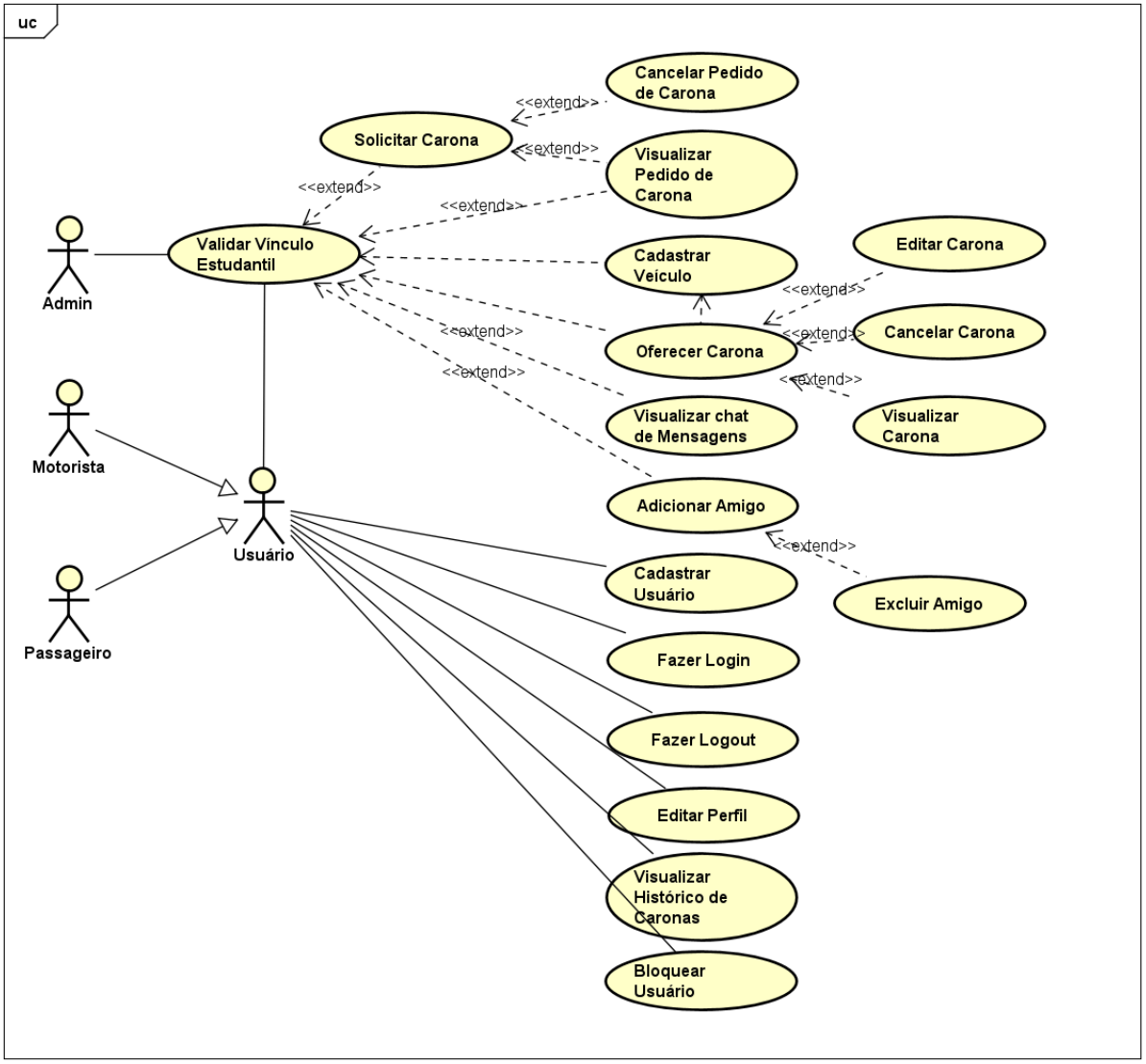
1.3	Para desbloquear um usuário, é preciso clicar no card de seu perfil após o resultado da busca para abrir a tela de Informações do perfil ou então clicar no botão de desbloquear no card.	Usabilidade
1.4	Ao clicar para desbloquear um usuário, o aplicativo mostrará um pop-up pedindo confirmação para a solicitação.	Usabilidade
1.5	Caso o usuário confirme, o aplicativo mostrará em seguida uma mensagem de confirmação do desbloqueio.	Usabilidade

3. PROTÓTIPO DO SISTEMA

Foi construído um protótipo no Figma, tendo telas para cada um dos requisitos funcionais identificados e também com mensagens de confirmação. O protótipo navegável do sistema pode ser acessado através do link: Protótipo no Figma.

4. CASO DE USO

Figura 1 - Diagrama UML de Caso de Uso.



Fonte: elaboração própria.

4.1. Caso de Uso Expandido

4.1.1. Caso de Uso – Cadastrar Usuário

Descrição: O usuário realiza seu cadastro, criando uma conta no aplicativo.

Ator Primário: Usuário

Atores Secundários: Motorista, Passageiro

Precondições: Possuir número de telefone e e-mail válidos, ou então uma conta Google já existente.

Fluxo Principal

1. O usuário clica no botão “Cadastre-se” na tela inicial do aplicativo.
2. O aplicativo abre a tela “Cadastre-se” com todos os campos em branco, com placeholders indicando o formato dos dados.
3. O aplicativo solicita os seguintes dados: Nome completo, Telefone e E-mail.
4. O usuário preenche os dados solicitados e clica no botão “Continuar”.
5. O sistema enviará uma verificação no e-mail informado.
6. O usuário confirma a solicitação.
7. O aplicativo disponibilizará um campo para que o usuário defina sua senha e um campo para que confirme sua senha.
8. O usuário cadastra uma senha para acessar sua conta.
9. O aplicativo mostra uma mensagem de confirmação, informando que o usuário foi cadastrado.
10. Caso deseje realizar seu cadastro por meio de sua conta Google, o usuário clica no botão “Continuar com Google” e cria sua conta.

4.1.2. Caso de Uso – Validar Vínculo Estudantil

Descrição: O usuário tem seu vínculo com o Instituto Federal de Rondônia validado.
Ator Primário: Usuário
Atores Secundários: Usuário
Precondições: Possuir uma conta cadastrada no aplicativo.
Fluxo Principal

1. O usuário clica no botão com ícone de arroba no menu inferior da aplicação.
2. O usuário realiza uma busca por outros usuário.
3. O usuário envia uma solicitação de amizade.
4. Ao ter seu pedido de amizade aceito, o usuário poderá oferecer e solicitar caronas aos usuários adicionados.

4.1.3. Caso de Uso – Fazer Login

Descrição: O usuário entra em sua conta no aplicativo.
Ator Primário: Usuário
Precondições: Possuir uma conta cadastrada no aplicativo.

Fluxo Principal

1. O usuário clica no botão “Entrar” na tela inicial do aplicativo.
2. O aplicativo abre a tela “Fazer Login” com todos os campos em branco, com placeholders indicando o formato dos dados.
3. O aplicativo solicita os seguintes dados: E-mail e Senha.
4. O usuário preenche os dados solicitados e clica no botão “Entrar”.
5. Caso deseje realizar seu login por meio de sua conta Google, o usuário clica no botão “Entrar com Google” e entra em sua conta.
6. O sistema fará a verificação dos dados informados.
7. Caso as informações estejam corretas, o sistema permitirá que o usuário entre em sua conta.

4.1.4. Caso de Uso – Solicitar Carona

Descrição: O usuário faz uma solicitação de carona pelo aplicativo.

Ator Primário: Passageiro

Precondições: Ter tido seu vínculo estudantil validado.

Fluxo Principal

1. O usuário clica no botão com ícone de lupa no menu inferior do aplicativo.
2. O sistema abre uma tela que possui o mapa com sua localização aproximada de fundo e um um card com campos para receber as informações do usuário.
3. O sistema solicita os seguintes dados: Endereço de partida, Endereço de destino, Data e Hora da viagem.
4. O usuário preenche as informações necessárias.
5. O usuário clica no ícone de calendário para escolher a data da carona.
6. O sistema abre um calendário com dias selecionáveis.
7. O usuário seleciona um ou mais dias.
8. O sistema automaticamente calcula um valor sugerido com base no percurso informado.
9. O usuário clica no botão “Pedir carona”.
10. O sistema grava o registro da carona solicitada no banco de dados.
11. O sistema abre uma outra tela apresentando as caronas já existentes para aquele mesmo destino com as datas correspondentes.
12. O usuário poderá criar um alerta para sua solicitação clicando no botão “Criar alerta”.
13. O usuário mostrará o pedido de carona como notificação aos usuários adicionados.
14. Após criado o pedido, o sistema mostrará a tela “Suas viagens”, com o resumo das informações da carona.

4.1.5. Caso de Uso – Visualizar Pedido de Carona em Andamento

Descrição: O usuário visualiza seus pedidos de carona em aberto.

Ator Primário: Passageiro

Precondições: Ter tido seu vínculo estudantil validado e ter realizado solicitações de carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Data e Hora da viagem, Valor sugerido e Status do pedido.

4.1.6. Caso de Uso – Editar Pedido de Carona

Descrição: O usuário edita informações em um pedido de carona.

Ator Primário: Passageiro

Precondições: Ter realizado solicitações de carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Data e Hora da viagem, Valor sugerido e Status do pedido.
4. O usuário clica no botão de Editar no card da carona que deseja modificar.
5. O sistema torna os campos Endereço de partida, Endereço de destino, Data e Hora editáveis.
6. O usuário realiza as alterações nas informações que deseja.
7. O usuário clica no botão “Salvar”.
8. O sistema mostra um pop-up solicitando a confirmação das alterações.
9. O usuário clica no botão “Confirmar”.
10. O sistema mostra uma mensagem de confirmação das alterações.
11. O sistema grava as alterações no banco de dados.

12. As informações são atualizadas na tela “Suas viagens”.

4.1.7. Caso de Uso – Cancelar Pedido de Carona

Descrição: O usuário poderá cancelar um pedido de carona.

Ator Primário: Passageiro

Precondições: Ter realizado solicitações de carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Data e Hora da viagem, Valor sugerido e Status do pedido.
4. O usuário clica no botão de Excluir no card da carona que deseja modificar.
5. O sistema mostra um pop-up solicitando a confirmação da exclusão.
6. O usuário clica no botão “Excluir”.
7. O sistema mostra uma mensagem de confirmação da exclusão.
8. O sistema grava as alterações no banco de dados.
9. O card da carona é excluído da tela “Suas viagens”.
10. Caso a carona já tenha sido aceita, a exclusão aparecerá como notificação para o usuário Motorista.

4.1.8. Caso de Uso – Oferecer Carona

Descrição: O usuário cria uma viagem, disponibilizando uma carona no aplicativo.

Ator Primário: Motorista

Precondições: Ter tido seu vínculo estudantil validado e ter criado um veículo anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de carro no menu inferior da aplicação.
2. O sistema abre a tela “Oferecer carona”.
3. O sistema solicita o endereço de saída.
4. O usuário insere o endereço e clica no botão “Continuar”.
5. O sistema solicita que o usuário selecione um veículo previamente cadastrado.
6. O usuário seleciona seu veículo e clica no botão “Continuar”.

7. O sistema pede que o usuário informe a quantidade de passageiros.
8. O usuário seleciona o número de passageiros e clica no botão “Continuar”.
9. O sistema solicita que o usuário informe os dias em que fará o trajeto.
10. O usuário clica no campo com um ícone de calendário.
11. O sistema abre um calendário com os dias selecionáveis.
12. O usuário seleciona os dias desejados e clica no botão “Continuar”.
13. O sistema solicita o horário do trajeto.
14. O usuário seleciona o horário e clica no botão “Continuar”.
15. O sistema dá a opção de já convidar os amigos adicionados para o trajeto.
16. Caso deseje, o usuário pode convidar os amigos para a carona.
17. O usuário clica no botão “Oferecer carona”.
18. O sistema abre a tela “Resumo da viagem”, mostrando todas as informações da carona.
19. O usuário clica no botão “Salvar”.
20. O sistema mostra um pop-up solicitando a confirmação.
21. O usuário clica no botão “Confirmar”.
22. O sistema mostra uma mensagem de confirmação de criação da viagem.
23. O sistema grava as alterações no banco de dados.

4.1.9. Caso de Uso – Visualizar Carona

Descrição: O usuário visualiza as viagens em aberto que está oferecendo.

Ator Primário: Motorista

Precondições: Ter tido seu vínculo estudantil validado e ter oferecido uma carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Quantidade de pessoas, Data e Hora da viagem.

4.1.10. Caso de Uso – Editar Carona

Descrição: O usuário edita informações de uma carona que está oferecendo.

Ator Primário: Motorista

Precondições: Ter oferecido uma carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Quantidade de pessoas, Data e Hora da viagem.
4. O usuário clica no botão de Editar no card da carona que deseja modificar.
5. O sistema torna os campos Endereço de partida, Endereço de destino, Quantidade de pessoas, Data e Hora editáveis.
6. O usuário realiza as alterações nas informações que deseja.
7. O usuário clica no botão “Salvar”.
8. O sistema mostra um pop-up solicitando a confirmação das alterações.
9. O usuário clica no botão “Confirmar”.
10. O sistema mostra uma mensagem de confirmação das alterações.
11. O sistema grava as alterações no banco de dados.
12. As informações são atualizadas na tela “Suas viagens”.
13. Caso já tenha amigos confirmados na carona, a modificação aparecerá como notificação para o usuário Passageiro.

4.1.11. Caso de Uso – Cancelar Carona

Descrição: O usuário poderá cancelar uma carona que está oferecendo.

Ator Primário: Motorista

Precondições: Ter oferecido uma carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de bússola no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Suas viagens”, que retorna uma lista de cards que mostram as informações de cada carona em aberto.
3. O sistema mostra os seguintes dados: Endereço de partida, Endereço de destino, Quantidade de pessoas, Data e Hora da viagem.
4. O usuário clica no botão de Excluir no card da carona que deseja modificar.

5. O sistema mostra um pop-up solicitando a confirmação da exclusão.
6. O usuário clica no botão “Excluir”.
7. O sistema mostra uma mensagem de confirmação da exclusão.
8. O sistema grava as alterações no banco de dados.
9. O card da carona é excluído da tela “Suas viagens”.
10. Caso a carona já tenha sido aceita, a exclusão aparecerá como notificação para o usuário Passageiro.

4.1.12. Caso de Uso – Cadastrar Veículo

Descrição: O usuário poderá cadastrar um novo veículo no aplicativo.

Ator Primário: Usuário

Precondições: Ter tido seu vínculo estudantil validado.

Fluxo Principal

1. O usuário clica no botão com ícone de perfil no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Visualizar perfil”, que retorna duas abas: “Seu perfil” e “Sua conta”.
3. Em “Seu perfil”, o usuário clica no botão “Adicionar veículo”.
4. O sistema abre a tela “Cadastrar veículo”.
5. O sistema solicita os seguintes dados: Veículo, Marca, Modelo, Placa e Cor.
6. O usuário preenche as informações necessárias e clica no botão “Salvar”.
7. O sistema mostra um pop-up solicitando a confirmação.
8. O usuário clica no botão “Confirmar”.
9. O sistema mostra uma mensagem de confirmação de cadastro do veículo.
10. O sistema grava as alterações no banco de dados.
11. O sistema mostra os veículos cadastrados na aba “Seu perfil”.

4.1.13. Caso de Uso – Editar Perfil

Descrição: O usuário visualizar e editar os dados de seu perfil.

Ator Primário: Usuário

Precondições: Ter criado uma conta no aplicativo.

Fluxo Principal

1. O usuário clica no botão com ícone de perfil no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Visualizar perfil”, que retorna duas abas: “Seu perfil” e “Sua conta”.
3. O usuário pode editar sua foto de perfil, adicionar uma minibiografia e visualizar seus veículos adicionados na aba “Seu perfil”.
4. O usuário pode alterar as informações da sua conta, como e-mail, telefone e senha, na aba “Sua conta”.

4.1.14. Caso de Uso –Visualizar Histórico de Caronas

Descrição: O usuário visualizar o histórico de caronas oferecidas e solicitadas.

Ator Primário: Usuário

Precondições: Ter tido seu vínculo estudantil validado e ter feito solicitações de carona anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de perfil no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Visualizar perfil”, que retorna duas abas: “Seu perfil” e “Sua conta”.
3. O usuário clica no botão “Histórico de caronas”.
4. O sistema abre a tela “Histórico de caronas”, que retorna uma lista de cards que mostram as informações de cada carona.
5. O sistema mostra os seguintes dados: ID da viagem, Endereço de partida, Endereço de destino, Quantidade de pessoas, Data e Hora da viagem.

4.1.15. Caso de Uso – Visualizar Amigos

Descrição: O usuário poderá visualizar os amigos adicionados.

Ator Primário: Usuário

Precondições: Ter tido seu vínculo estudantil validado.

1. O usuário clica no botão com ícone de arroba no menu inferior da aplicação.
2. O sistema abre a tela “Seus amigos”.
3. O sistema mostra uma lista de cards contendo os amigos adicionados.
4. O sistema mostra as seguintes informações: Foto de perfil, Nome e Cidade.

5. O usuário clica no card do amigo que deseja visualizar.
6. O sistema abre a tela “Informações do perfil”.
7. O sistema mostra as seguintes informações: Foto de perfil, Nome, Cidade, Biografia e Veículos.

4.1.16. Caso de Uso – Adicionar Amigos

Descrição: O usuário poderá enviar pedidos de amizade para outros usuários.

Ator Primário: Usuário

Precondições: Ter criado uma conta no aplicativo.

1. O usuário clica no botão com ícone de arroba no menu inferior da aplicação.
2. O sistema abre a tela “Seus amigos”.
3. O usuário digita o nome do usuário que deseja buscar no campo de busca.
4. O sistema consulta o banco de dados e retorna uma lista com os resultados.
5. O usuário clica no botão de “Adicionar”, ou então clica no card do usuário que deseja adicionar.
6. O sistema abre a tela “Informações do perfil”.
7. O sistema mostra as seguintes informações: Foto de perfil, Nome, Cidade, Biografia e Veículos.
8. O usuário clica no botão “Adicionar amigo”.
9. O sistema mostra um pop-up solicitando a confirmação.
10. O usuário clica no botão “Confirmar”.
11. O sistema mostra uma mensagem de confirmação de envio da solicitação de amizade.
12. O sistema envia a solicitação ao outro usuário como notificação.
13. Após a solicitação ser aceita, o sistema mostrará um card com o nome do usuário na tela “Seus amigos”.

4.1.17 Caso de Uso – Excluir Amigos

Descrição: O usuário poderá excluir um amigo que tem adicionado.

Ator Primário: Usuário

Precondições: Ter criado uma conta no aplicativo e ter amigos adicionados.

1. O usuário clica no botão com ícone de arroba no menu inferior da aplicação.
2. O sistema abre a tela “Seus amigos”.
3. O usuário digita o nome do usuário que deseja buscar no campo de busca.
4. O sistema consulta o banco de dados e retorna uma lista com os resultados.
5. O usuário clica no botão de “Excluir”, ou então clica no card do usuário que deseja adicionar.
6. O sistema abre a tela “Informações do perfil”.
7. O sistema mostra as seguintes informações: Foto de perfil, Nome, Cidade, Biografia e Veículos.
8. O usuário clica no botão “Excluir amigo”.
9. O sistema mostra um pop-up solicitando a confirmação.
10. O usuário clica no botão “Confirmar”.
11. O sistema mostra uma mensagem confirmando a exclusão de amizade.
12. Após a exclusão, o sistema não mostrará mais o card com o nome do usuário na tela “Seus amigos”.

4.1.18 Caso de Uso – Bloquear Usuários

Descrição: O usuário poderá bloquear um usuário.

Ator Primário: Usuário

Precondições: Ter criado uma conta no aplicativo.

1. O usuário clica no botão com ícone de arroba no menu inferior da aplicação.
2. O sistema abre a tela “Seus amigos”.
3. O usuário digita o nome do usuário que deseja buscar no campo de busca.
4. O sistema consulta o banco de dados e retorna uma lista com os resultados.
5. O usuário clica no card do usuário que deseja adicionar.
6. O sistema abre a tela “Informações do perfil”.
7. O sistema mostra as seguintes informações: Foto de perfil, Nome, Cidade, Biografia e Veículos.
8. O usuário clica no botão “Bloquear usuário”.
9. O sistema mostra um pop-up solicitando a confirmação.
10. O usuário clica no botão “Confirmar”.
11. O sistema mostra uma mensagem confirmando o bloqueio.

12. Após o bloqueio, o sistema não mostrará caronas solicitadas ou oferecidas ao usuário bloqueado.

4.1.19. Caso de Uso – Visualizar chat de mensagens

Descrição: O usuário pode visualizar as suas notificações e mensagens.

Ator Primário: Usuário

Precondições: Ter tido seu vínculo estudantil validado e ter feito solicitações de carona ou ter oferecido caronas anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de mensagem que aparece em algumas telas da aplicação.
2. O sistema abre a tela de chat de mensagens, que possui duas abas: “Notificações” e “Mensagens”.
3. Na aba “Notificações”, o sistema mostra as notificações em formato de card.
4. O usuário pode clicar no botão com ícone de “+”.
5. O sistema abre o card para mostrar mais detalhes da notificação.
6. O usuário acessa a aba “Conversas”.
7. O sistema retorna uma lista com as mensagens do usuário.
8. O usuário clica na mensagem que deseja visualizar.
9. O sistema abre a conversa selecionada.

4.1.20. Caso de Uso – Visualizar Usuários Bloqueados

Descrição: O usuário pode visualizar os usuários que bloqueou.

Ator Primário: Usuário

Precondições: Ter tido seu vínculo estudantil validado e ter bloqueado usuários anteriormente.

Fluxo Principal

1. O usuário clica no botão com ícone de perfil no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Visualizar perfil”, que retorna duas abas: “Seu perfil” e “Sua conta”.
3. O usuário clica no botão “Usuários bloqueados”.

4. O sistema abre a tela “Usuários bloqueados”, que retorna uma lista de cards que mostram as informações de cada usuário bloqueado.
5. O sistema mostra as seguintes informações: Foto de perfil, Nome e Cidade.
6. O usuário clica no card do amigo que deseja visualizar.
7. O sistema abre a tela “Informações do perfil”.
8. O sistema mostra as seguintes informações: Foto de perfil, Nome, Cidade, Biografia e Veículos.
9. O usuário clica no botão “Desbloquear usuário”.
10. O sistema mostra um pop-up solicitando a confirmação.
11. O usuário clica no botão “Confirmar”.
12. O sistema mostra uma mensagem de confirmação de desbloqueio.
13. O sistema grava as alterações no banco de dados.

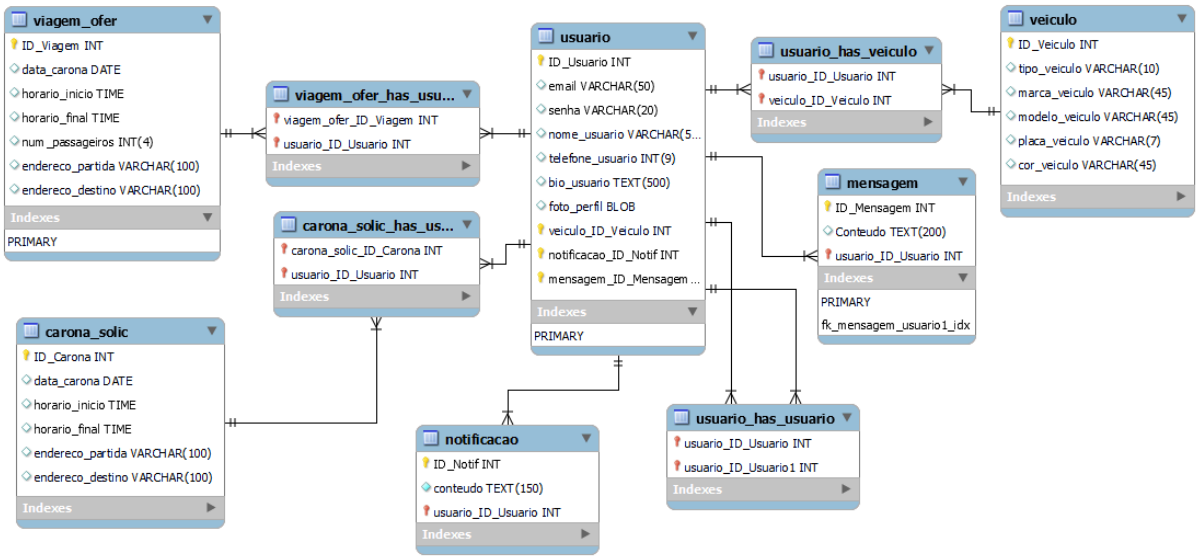
4.1.21. Caso de Uso – Fazer Logout

Descrição: O usuário sai da sua conta no aplicativo.
Ator Primário: Usuário
Precondições: Estar logado em sua conta.
Fluxo Principal

1. O usuário clica no botão com ícone de perfil no menu inferior da aplicação.
2. O sistema disponibiliza a tela “Visualizar perfil”, que retorna duas abas: “Seu perfil” e “Sua conta”.
3. O usuário clica no botão “Sair”.
4. O sistema mostra um pop-up solicitando a confirmação.
5. O usuário clica no botão “Confirmar”.
6. O sistema desloga da conta.
7. O sistema mostra a Tela inicial do aplicativo.

5. BANCO DE DADOS

Figura 2 - Diagrama Lógico Entidade-Relacionamento.



Fonte: elaboração própria.

6. REFERÊNCIAS

AMAZON. **O que é o Scrum?** Disponível em: https://aws.amazon.com/pt/what-is/scrum/. Acesso em 20 de março de 2025.

EXPO. **Expo Documentation**. Disponível em: https://docs.expo.dev/. Acesso em 15 de março de 2025.

FALCÃO, Filipe Dourado. **Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native**. TCC (Graduação em Sistemas e Mídias Digitais) – Universidade Federal do Ceará. Fortaleza - CE, 2022. Disponível em: https://repositorio.ufc.br/handle/riufc/69029. Acesso em 17 de março de 2025.

FIGMA. **O que é o Figma?** Disponível em: https://www.figma.com/pt-br/prototyping/. Acesso em 17 de março de 2025.

KLOH, Gustavo Minuzzi. HABITZREITER, Taiane Tais. BARCAROLI, Velcir. **Peculiaridades de um banco MySQL.** nº 11. Revista Conexão, 2024. Disponível em: https://revistas.uceff.edu.br/conexao/article/view/531/476. Acesso em 17 de março de 2025.

MELO, Lidiane da Silva. **Projeto e prototipação do virtual biblioteca utilizando a ferramenta figma**. TCC (Licenciatura em Computação) – Universidade Federal Rural do Semi-Árido, 2024. Disponível em: https://repositorio.ufersa.edu.br/items/ac8bfb89-deda-4584-a78b-da9df610743d. Acesso em 17 de março de 2025.

MIRO. **Modelo para Diagrama UML**. Disponível em: https://miro.com/pt/modelos/diagramasuml/. Acesso em 18 de março de 2025.

SCHWABER, Ken. SUTHERLAND, Jeff. **Guia do Scrum.** 2013. Disponível em: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf&ved=2ahUKEwiVi4Gg05SMAxX2ErkGHWK8AukQFnoECCAQAQ&usg=AOvVaw2da_5liRyb0pv_VlYt2Ehh. Acesso em 17 de março de 2025.