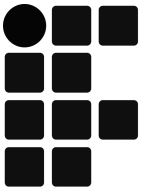


INSTITUTO FEDERAL
Farroupilha
Campus Uruguaiana

CURSO TÉCNICO EM INFORMÁTICA INTEGRADO AO ENSINO MÉDIO

EDUARDO ANTUNES DA SILVA

OMBO:

APLICATIVO DE MOBILIDADE URBANA PÚBLICA

URUGUAIANA

2025

EDUARDO ANTUNES DA SILVA

OMBO:

APLICATIVO DE MOBILIDADE URBANA PÚBLICA

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Informática Integrado ao Ensino Médio do *Campus* Uruguaiana do Instituto Federal de Educação, Ciência e Tecnologia Farroupilha como requisito parcial para a obtenção do título de Técnico em Informática.

Orientadores:

Eduardo Henrique Spies

Michel Michelon

URUGUAIANA

2025

Antunes da Silva, Eduardo.
Ombo: APLICATIVO DE MOBILIDADE URBANA / Eduardo Antunes da Silva. — 2025.
[qtd. de folhas] f.
Trabalho de Conclusão de Curso Técnico – Instituto Federal de Educação, Ciência e Tecnologia Farroupilha, Uruguaiana, 2025.
1. Mobilidade Urbana. 2. Aplicativo. 3. Geolocalização. I. OMBO: Aplicativo de Mobilidade Urbana Pública.

CDD [número da CDD].

EDUARDO ANTUNES DA SILVA

OMBO:

APLICATIVO DE MOBILIDADE URBANA PÚBLICA

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Informática Integrado ao Ensino Médio do *Campus* Uruguaiana do Instituto Federal de Educação, Ciência e Tecnologia Farroupilha como requisito parcial para a obtenção do título de Técnico em Informática.

Este trabalho foi defendido e aprovado pela banca em 09/01/2026.

BANCA EXAMINADORA

_____

Prof. Me. Eduardo Henrique Spies
Orientador

_____

Prof. Me. Michel Michelon
Coorientador

_____

Prof.ª Dr.ª Melina Mörschbächer
Avaliadora

_____

Prof. Me. Thiago Cassio Krug
Avaliador

Dedico este trabalho aos meus orientadores, Eduardo Henrique Spies, Michel Michelon e Stéphane Rodrigues Dias, aos meus amigos, família e minha namorada, que sempre contribuíram para que este trabalho fosse desenvolvido da melhor maneira possível.

AGRADECIMENTOS

Gostaria de agradecer a todos que me auxiliaram na realização deste Trabalho de Conclusão de Curso, independentemente do meio. Em especial, agradeço a Michel Michelon, Eduardo Henrique Spies e Stéphane Rodrigues Dias, meus orientadores durante o desenvolvimento do projeto. Sem eles, este trabalho jamais teria saído do papel.

Agradeço à minha família, que sempre me apoiou ao longo da minha jornada acadêmica, principalmente nos momentos em que me senti incapaz de concluir o Trabalho de Conclusão de Curso.

À minha namorada, Daniella Camponogara Valdez, por sempre me incentivar a continuar o projeto e por recordar minha plena capacidade de finalizá-lo.

Aos meus amigos, carinhosamente chamados de "*Os Bola*", por estarem ao meu lado em momentos difíceis, trazendo leveza aos dias, como se o peso da vida acadêmica não recaísse sobre meus ombros.

Por fim, agradeço ao Instituto Federal Farroupilha (IFFar) – *Campus* Uruguaiana, por me proporcionar a oportunidade de ampliar meus conhecimentos e testar meus limites.

Nem sempre tudo parece bem, mas sempre tudo parece bem.

FERNANDO ASSENÇO COSTA

RESUMO

Este Trabalho de Conclusão de Curso descreve o desenvolvimento de um aplicativo de mobilidade urbana destinado à população de Uruguaiana (RS), criado para suprir a carência de informações acessíveis sobre o transporte público, como horários, rotas e paradas de ônibus. A proposta oferece, de forma gratuita e prática, dados completos de todas as linhas, com visualização das rotas em mapa, localização das paradas, horários previstos e acompanhamento em tempo real dos ônibus. Desenvolvido com foco na acessibilidade, o aplicativo apresenta uma interface simples e intuitiva, permitindo que usuários com diferentes níveis de familiaridade tecnológica encontrem facilmente a melhor opção de deslocamento. Dessa forma, o projeto busca melhorar o planejamento das viagens, fortalecer a mobilidade urbana, incentivar a digitalização dos serviços públicos e garantir acesso a informações confiáveis sobre o transporte coletivo da cidade.

Palavras-chave: Mobilidade Urbana; Aplicativo; Geolocalização.

ABSTRACT

This Course Conclusion Project describes the development of an urban mobility application aimed at the population of Uruguaiana (RS), created to address the lack of accessible information about public transportation, such as bus schedules, routes, and stops. The proposal offers complete data on all bus lines in a practical and free manner, including route visualization on a map, stop locations, estimated schedules, and real-time bus tracking. Developed with a focus on accessibility, the application features a simple and intuitive interface, allowing users with different levels of technological familiarity to easily find the best travel option. Thus, the project seeks to improve trip planning, strengthen urban mobility, promote the digitalization of public services, and ensure access to reliable information about the city's public transportation system.

Keywords: Urban Mobility; Application; Geolocation.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IFFar	Instituto Federal de Educação, Ciência e Tecnologia Farroupilha
JSON	JavaScript Object Notation
PPI	Projeto Prático Integrado

SUMÁRIO

1 INTRODUÇÃO

A mobilidade urbana pública é um direito de todo cidadão. Para que uma sociedade contemporânea funcione, é necessário que os cidadãos possam se mover pela cidade livremente para exercer suas funções com êxito. É necessária a disponibilidade de meios de transporte públicos por parte do Estado, não só por ser um direito, mas também para deslocar as pessoas e consequentemente impactar positivamente a economia.

Quando o cidadão utiliza o transporte público como principal meio de locomoção, determinados direitos devem ser garantidos, entre os quais se destacam o direito à segurança, à acessibilidade e à informação. Porém, nem todos são cumpridos. Na realidade de Uruguaiana (RS), a falta de acesso à informação básica para o uso do transporte público coletivo é muito clara. Tanto as linhas de ônibus quanto os pontos de ônibus não são registrados e divulgados de maneira acessível à população, conforme assegurado pela Lei da Mobilidade Urbana (Brasil, 2012).

A partir desse cenário, entende-se que melhorar a comunicação é a chave principal para resolver o déficit criado em volta da distribuição precária de informações em relação às linhas pelo setor responsável pela mobilidade urbana de Uruguaiana. Cidadãos que vêm de outras cidades ou que nunca utilizaram transporte coletivo acabam não podendo exercer seu direito ao transporte público plenamente. Caso existisse um meio mais eficiente de comunicação em relação aos ônibus, trajetos e horários, isso incentivaria o uso do transporte público, reduzindo o uso de carros, o que traria mais sustentabilidade para a cidade. Além disso, uma vez que os cidadãos fossem mais informados, teriam mais motivação para utilizar o transporte público.

Assim, a proposta desse Trabalho de Conclusão de Curso consiste na criação de uma plataforma gratuita que disponibiliza, de forma clara e acessível, dados sobre todas as linhas de ônibus, suas respectivas rotas e os horários de chegada em cada ponto. O aplicativo busca resolver o problema na comunicação dessas informações, facilitando o uso do transporte coletivo por novos usuários e pela população em geral. Sua principal característica é a interface de fácil utilização, que traz as informações em um mapa da cidade, tirando a necessidade de o usuário

conhecer previamente o nome das ruas ou pontos específicos para realizar sua pesquisa.

1.1 JUSTIFICATIVA

Em uma tentativa de resolver problemas da cidade, no ano de 2024, a câmara de vereadores de Uruguaiana propôs para os alunos do IFFar[1] problemas regionais, a partir da Prática Profissional Integrada (PPI) *Design Thinking* - Arquitetos do Amanhã. Dentre os problemas, a falta de informação sobre o transporte urbano foi escolhida pelo seguinte grupo: Eduardo Antunes da Silva, Arthur do Amaral Jacques e Samuel Alvarenga Prates. Como solução, foi proposta a criação de um site que mostrasse informações sobre as linhas e paradas de ônibus de forma acessível. Para fazer isso, o projeto foi validado a partir de entrevistas, infográficos e *pitches*[2] para apresentação da proposta do projeto. Entretanto, não houve o desenvolvimento do aplicativo.

Diante disso, este Trabalho de Conclusão de Curso tem como objetivo dar continuidade ao desenvolvimento do projeto na forma de um aplicativo, a partir da proposta criada na PPI, que atenda às necessidades da população de Uruguaiana à respeito da falta de acesso à informação sobre o transporte público.

A falta de informações sobre as rotas e horários do transporte público de Uruguaiana leva à restrição desse direito básico. A criação de um aplicativo que colabore na resolução dessa demanda é essencial, especialmente para garantir aos cidadãos o incentivo ao uso e acesso a outras formas de locomoção. Além disso, ao aumentar a utilização de ônibus, trará mais sustentabilidade à cidade, reduzindo a pegada de carbono de Uruguaiana. Por fim, o projeto atinge uma demanda real da comunidade e trará a modernização e digitalização de seus serviços públicos, o que justifica sua relevância e necessidade.

---

[1] IFFar: Instituto Federal de Educação, Ciência e Tecnologia Farroupilha, instituição pública de ensino que oferece cursos técnicos, de graduação e pós-graduação no Rio Grande do Sul.
[2] Pitch: apresentação curta e objetiva de uma ideia, projeto ou produto, com o objetivo de comunicar seu valor principal e despertar o interesse do público ou de avaliadores.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um aplicativo que exibe informações sobre o transporte coletivo disponível na cidade, mais precisamente mapas com as linhas, paradas e horários dos ônibus de Uruguaiana.

1.2.2 Objetivos Específicos

- Exibir mapas com as respectivas rotas de cada linha de ônibus.
- Exibir a localização das paradas de ônibus de cada rota nos mapas.
- Disponibilizar informações dos horários previstos de chegada dos ônibus em cada parada.
- Permitir o recebimento da localização do condutor/condução, para a representação do deslocamento do ônibus no mapa.
- Informar automaticamente os usuários sobre eventuais problemas com a linha ou atrasos decorrentes dos mesmos.
- Garantir acessibilidade da informação ao público, por meio de uma interface intuitiva.

1.3 METODOLOGIA

A fase de ideação do projeto do aplicativo de mobilidade urbana foi iniciada na PPI *Design Thinking* – Arquitetos do Amanhã em forma de grupo. Sua continuação como Trabalho de Conclusão foi separada entre dois integrantes do grupo: Samuel Alvarenga Prates e Eduardo Antunes da Silva. Sendo assim, foi decidido a segmentação do trabalho entre usuários do aplicativo. Nesse sentido, o projeto foi dividido em duas partes, sendo que o aplicativo que apresenta a visão do passageiro e administração de linhas e paradas foi desenvolvido por mim, Eduardo, enquanto que ao aplicativo que apresenta a visão do motorista, permitindo o envio

da localização e status do ônibus foi implementada pelo colega Samuel. Os dois aplicativos dialogam entre si, garantindo a correta comunicação sobre o funcionamento das linhas para o usuário no aplicativo aqui apresentado.

Assim, para a realização deste trabalho foram realizadas as seguintes etapas metodológicas:

Definir o tema: O tema foi definido a partir de uma PPI realizada no primeiro semestre de 2024. A PPI teve como objetivo solucionar problemas regionais, sendo alguns desses problemas apresentados pela câmara de vereadores de Uruguaiana. Dos problemas apresentados foi escolhido resolver o da falta de informações sobre o transporte público de Uruguaiana, por meio de um aplicativo. E esse aplicativo serviu de inspiração para o tema do trabalho de conclusão de curso.

Pesquisar trabalhos correlatos: Foram pesquisados os trabalhos correlatos a este nas plataformas Google Acadêmico, Pesquisa Google e Arandu do IFFar. Os trabalhos com maior semelhança a este foram selecionados e resumidos. Os resumos estão descritos na seção 2.1.

Pesquisar sistemas semelhantes: A pesquisa de sistemas semelhantes busca apresentar erros e acertos de sistemas relacionados ao mesmo tema deste. Assim, facilitando o aprimoramento do sistema com as funcionalidades que deram certo em outros sistemas e não colocar funcionalidades desnecessárias. Os sistemas semelhantes foram encontrados a partir de pesquisas e utilizações anteriores.

Criar wireframes: Os wireframes foram feitos provisoriamente em uma folha de caderno tanto para melhorar compreensão do sistema pelos orientadores quanto para melhor visualização e desenvolvimento dos casos de uso do software. Assim, dando início ao planejamento do sistema para que não haja problemas lógicos e que fosse visto o que não seria possível implementar no prazo.

Criar o banco de dados: O banco de dados foi a primeira etapa do desenvolvimento do sistema, com sua criação, é iniciada a programação do sistema. Ele foi criado a partir dos wireframes do sistema e ajuda a visualizar a lógica dos itens no sistema.

Desenvolver o sistema: O sistema foi desenvolvido em Kotlin para que seja utilizado como aplicativo de celular. Isso se deve à necessidade de versatilidade no uso do sistema, uma vez que é mais confortável utilizar um aplicativo no celular do que um site.

Escrever a documentação de requisitos: Os requisitos foram identificados por um código único, seguido do nome do requisito entre colchetes. A referência a eles deve usar a subseção onde estão descritos. Quanto à prioridade, os requisitos podem ser classificados como: Essencial, sem o qual o sistema não funciona; Importante, cuja ausência compromete a qualidade, mas não impede o funcionamento; e Desejável, que não afeta as funções principais e pode ser adiado para versões futuras.

Escrever a documentação de casos de uso: A documentação de casos de uso apresenta cada ação do sistema e usuário em relação aos casos de uso, estes serão mostrados em forma de diagrama e detalhados conforme o necessário. Sendo que cada ação do usuário tem uma reação do sistema descrita na documentação.

Escrever o trabalho: Depende de todas as outras etapas da metodologia. A documentação dos processos é necessária para facilitar a visualização de inconsistências do sistema, evitando problemas futuros, além de possibilitar a implementação de sistemas semelhantes para atender demanda análoga em outros municípios.

2 REVISÃO BIBLIOGRÁFICA

O objetivo da revisão bibliográfica é construir o embasamento teórico do Trabalho de Conclusão de Curso, relacionando este com outros projetos semelhantes, ou seja, aplicativos de mobilidade urbana com sistema de geolocalização.

2.1 TRABALHOS SEMELHANTES

O documento “Mobilidade Urbana” (CREA-PR, 2018) aborda os principais conceitos, fatores e desafios relacionados à mobilidade nas cidades brasileiras. O texto destaca que o crescimento urbano desordenado, aliado à priorização histórica do transporte individual motorizado e à falta de investimentos públicos, agravou os congestionamentos e a exclusão social no acesso aos espaços urbanos.

Defende-se, nesse documento, a mobilidade urbana sustentável, baseada em políticas que priorizem o transporte coletivo e os modos não motorizados, como caminhar e pedalar, garantindo acessibilidade universal e integração entre diferentes modais. A literatura também enfatiza a importância do planejamento urbano integrado, do uso racional do solo e da reestruturação institucional e participativa das políticas públicas de transporte, como caminho para cidades mais inclusivas, eficientes e ambientalmente equilibradas (Vaccari Fanini, 2011).

O artigo “O Plano de Mobilidade Urbana e o Futuro das Cidades”, de Bárbara Rubim e Sérgio Leitão (2013), apresenta um panorama histórico da mobilidade urbana, mostrando como a priorização do transporte individual, antes com cavalos e hoje com automóveis, gerou graves problemas sociais, ambientais e de saúde pública. Os autores analisam a Política Nacional de Mobilidade Urbana (Lei nº 12.587/2012), destacando seus avanços, como o incentivo ao transporte coletivo e não motorizado, e suas fragilidades, como a falta de planejamento efetivo e de participação social. O texto defende a necessidade urgente de reduzir a dependência do automóvel, fortalecer o transporte público e devolver aos cidadãos o direito à cidade e a um ambiente urbano mais sustentável e inclusivo (Rubim; Leitão, 2013).

O trabalho “Uma Visão da Mobilidade Urbana Sustentável”, de Vânia Barcellos Gouvêa Campos, discute a busca por uma mobilidade que concilie desenvolvimento urbano, equidade social e preservação ambiental. A autora propõe que a mobilidade sustentável deve equilibrar aspectos sociais, econômicos e ambientais, promovendo o acesso eficiente a bens e serviços sem comprometer as gerações futuras. O texto destaca a importância de políticas que integrem o uso do solo e os transportes, incentivem o transporte público e não motorizado, reduzam o uso do automóvel e adotem tecnologias limpas. Defende-se ainda o papel essencial do planejamento urbano e da atuação conjunta entre governo, operadoras e fabricantes para garantir cidades mais justas, acessíveis e ambientalmente equilibradas (Campos, 2006).

2.2 SISTEMAS SEMELHANTES

O aplicativo Moovit[3] busca oferecer informações de transporte público em tempo real em diversos modais de transporte, incluindo ônibus, trólebus, bondes, trens, metrô, balsas e barcas, além de bicicletas e patinetes compartilhados; e outros componentes da micromobilidade. Os usuários podem acessar um mapa ao vivo e ver as paradas e estações próximas com base em sua localização GPS atual, bem como planejar rotas nos diversos modos de transporte, com base em dados em tempo real. O aplicativo usa tanto dados oficiais como informações da comunidade - que integra dados estáticos de transporte público de operadores de trânsito, com dados em tempo real coletados de usuários. O Moovit usa então o sistema de *crowdsourcing* (ambiente online que utiliza a contribuição coletiva para realizar tarefas, gerar ideias e criar projetos) com horários de transporte público para melhorar os resultados do plano de viagem com base nas condições atuais; e compartilhar esses dados com a comunidade de usuários. Além das atualizações sobre o transporte público, o aplicativo ainda oferece suporte para caminhos feitos a pé.

O aplicativo Cittamobi[4] funciona como um facilitador da sua rotina no

[3] O software Moovit pode ser encontrado e consultado no site oficial do Moovit por meio do seguinte link: https://moovitapp.com. Acesso em: Julho de 2025.
[4] O software CittaMobi pode ser encontrado e consultado na PlayStore por meio do seguinte link: https://play.google.com/store/apps/details?id=br.com.cittabus&hl=pt-BR

transporte público. No aplicativo, é possível ver as previsões das linhas em tempo real ou estimado, favoritar as linhas mais usadas e traçar rotas. Além disso, a depender da sua localidade, você consegue cadastrar seu cartão de transporte, como também solicitar atendimento digital junto às empresas de ônibus.

Quadro 1 – Comparação entre Sistemas Semelhantes

Sistemas	**Pontos Positivos**	**Pontos Negativos**
Moovit	- Informações em tempo real para diversos modais (ônibus, trens, bicicletas, etc.). - Combina dados oficiais com crowdsourcing. - Oferece mapa ao vivo com base na localização GPS. - Planejamento de rotas com dados atualizados. - Suporte para caminhos a pé.	- Depende da colaboração da comunidade, o que pode afetar a precisão. - Complexidade na integração dos dados pode causar inconsistências momentâneas.
Cittamobi	- Possibilidade de cadastrar cartão de transporte. - Permite favoritar linhas usadas. - Previsões de linhas em tempo real ou estimado. - Atendimento digital junto às empresas de ônibus.	- Tem menos modais e dados comparado ao Moovit. - Funcionalidades podem variar dependendo da localidade.

Fonte: Autoria Própria (2025).

3 DESENVOLVIMENTO

Esta seção apresenta o processo do desenvolvimento do sistema proposto neste trabalho. Ela está dividida em 4 partes: requisitos funcionais, casos de uso, diagrama do banco de dados e interfaces do sistema.

3.1 DOCUMENTAÇÃO DE REQUISITOS

Esta seção especifica os requisitos do sistema, que fornecem ao desenvolvedor as informações necessárias para a implementação do sistema.

3.1.1 Requisitos funcionais do sistema

Abaixo são listados os requisitos funcionais previstos para o sistema.

[RF01] Manter Cadastro	
Descrição:	O usuário pode cadastrar-se, excluir sua conta, listar e editar seus dados.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF02] Selecionar Parada	
Descrição:	O usuário pode selecionar uma parada e o sistema exibe as informações da mesma.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF03] Selecionar Ônibus	
Descrição:	O usuário pode selecionar um ônibus e o sistema exibe as informações do mesmo.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF04] Mostrar Horários da Rota	
Descrição:	O usuário pode selecionar os horários da rota e o sistema exibe os horários.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF05] Realizar Login	
Descrição:	O usuário realiza o login e é redirecionado para a tela inicial do aplicativo.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF06] Acessar Ouvidoria	
Descrição:	O usuário pode acessar um número de Whatsapp para relatar problemas do aplicativo.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF07] Acompanhar Rotas	
Descrição:	O usuário pode acessar o mapa com as rotas, paradas e ônibus rastreados em tempo real.
Prioridade:	■ Essencial □ Importante □ Desejável

Fonte: Autoria Própria (2025).

[RF08] Manter Paradas	
Descrição:	O funcionário pode cadastrar, excluir, listar e editar paradas.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF09] Manter Pontos da Rota	
Descrição:	O funcionário pode cadastrar, excluir, listar e editar Pontos da Rota.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF10] Manter Rotas	
Descrição:	O funcionário pode cadastrar, excluir, listar e editar
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

[RF11] Manter Usuários	
Descrição:	O funcionário pode cadastrar, excluir e listar usuários e editar seus dados.
Prioridade:	□ Essencial □ Importante ■ Desejável

Fonte: Autoria Própria (2025).

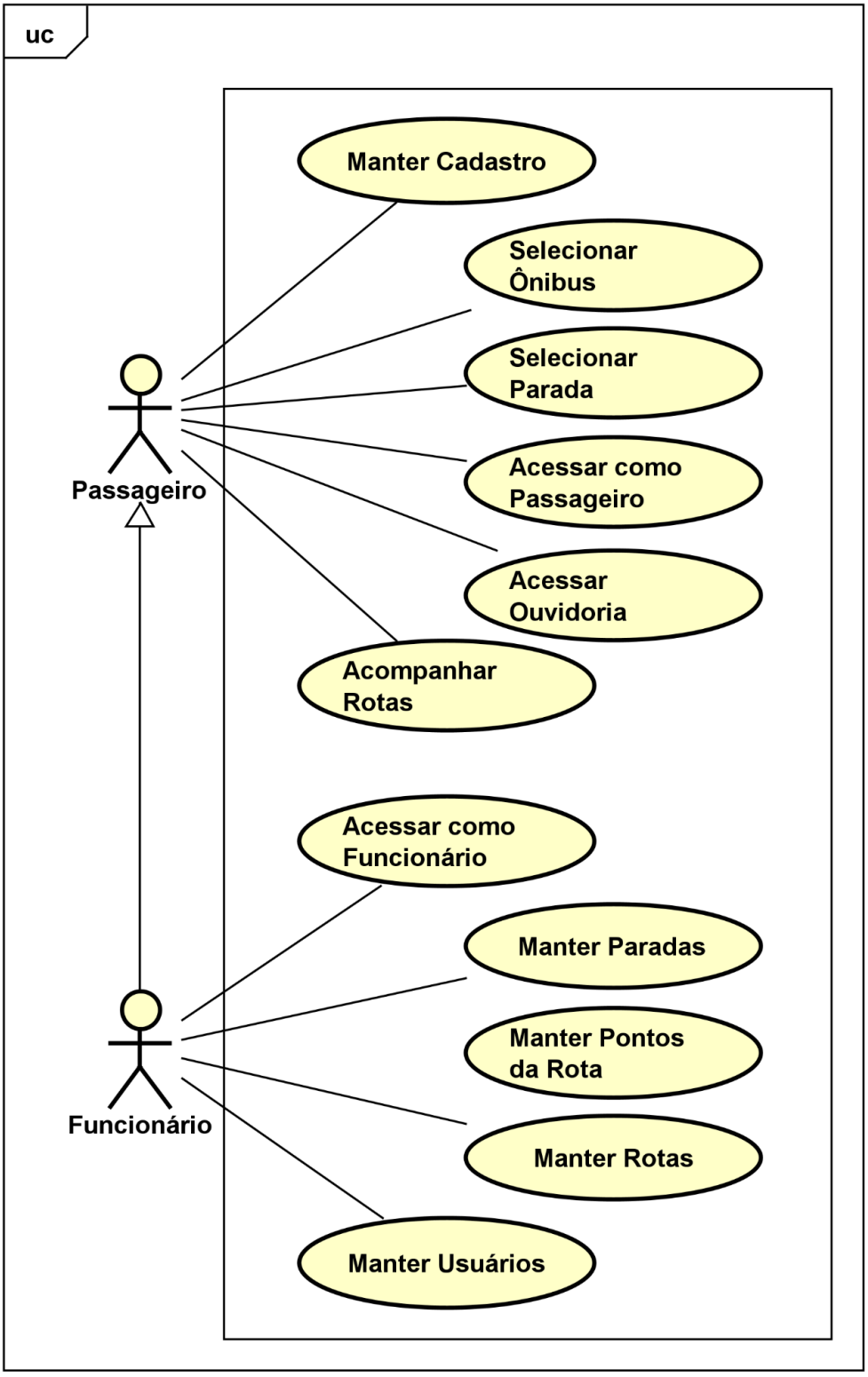
3.2 CASOS DE USO

A Figura 1 apresenta o Diagrama de Casos de Uso do sistema, o qual tem como objetivo representar de forma visual as interações possíveis entre os atores e as funcionalidades disponibilizadas pelo aplicativo. O diagrama evidencia dois atores principais: o Passageiro e o Funcionário, cada um com permissões e responsabilidades distintas dentro do sistema.

O ator Passageiro é responsável pelas ações relacionadas ao uso cotidiano do aplicativo, podendo realizar o manter cadastro, que inclui cadastrar-se, editar ou excluir seus dados. Além disso, o passageiro pode acessar como passageiro, selecionar ônibus, selecionar parada, acompanhar rotas no mapa e acessar a ouvidoria, funcionalidade destinada ao contato para registro de reclamações ou sugestões.

Já o ator Funcionário possui acesso a todas funcionalidades do Passageiro e as funcionalidades administrativas do sistema. Ele pode acessar como funcionário e realizar operações de gerenciamento, como manter paradas, manter pontos da rota, manter rotas e manter usuários permitindo o controle e a atualização das informações essenciais para o funcionamento correto do aplicativo.

Figura 1 – Diagrama de Casos de Uso.



Fonte: Autoria Própria (2025).

3.2.1 Documentação de casos de uso

A presente seção apresenta os quadros com as descrições dos casos de uso requisitados com as especificações de atores, pré-condições, pós-condições, fluxo principal, fluxo alternativo e fluxo de exceção de cada caso necessário para o desenvolvimento do sistema. Os atores do sistema estão distribuídos em nível de funcionário/administrador e passageiro.

Quadro 2 – Manter Cadastro

[UC01] Manter Cadastro	
Atores:	Passageiro e Funcionário.
Pré-Condições:	Para cadastrar uma conta no sistema não há pré-condições, mas para editar, listar e excluir informações da conta, o usuário deve estar cadastrado e logado no sistema.
Pós-Condições:	Um usuário cadastrado, excluído, editado ou listado no sistema.
FLUXO PRINCIPAL	
Cadastrar conta: C1. O usuário solicita o formulário de cadastro para o sistema. C2. O sistema exibe o formulário de cadastro para o usuário. C3. O usuário preenche o formulário com as informações da sua conta e solicita o cadastro da conta no sistema. C4. O sistema registra a conta do usuário e envia um email de verificação de email para o usuário. C5. O usuário verifica o seu email. C6. O sistema libera o acesso do usuário ao sistema. Alterar conta: A1. O usuário acessa a página do seu perfil. A2. O sistema exibe as informações da conta do usuário. A3. O usuário realiza as alterações necessárias e solicita o registro dessas alterações no sistema. A4. O sistema registra as alterações realizadas pelo usuário.	

Excluir conta: E1. O usuário acessa a página do seu perfil. E2. O sistema exibe as informações da conta do usuário e disponibiliza a opção de exclusão da conta. E3. O usuário seleciona a opção de exclusão da conta. E4. O sistema solicita a confirmação de exclusão da conta. E5. O usuário confirma a exclusão da sua conta. E6. O sistema realiza a exclusão da conta do usuário. Listar cadastro: L1. O usuário acessa a página do seu perfil. L2. O sistema realiza a listagem de todas as informações da conta do usuário.
FLUXOS ALTERNATIVOS
E3. O usuário seleciona a opção de cancelar. E3.1. O sistema exibe as informações da conta do usuário e disponibiliza a opção de exclusão da conta.
FLUXOS DE EXCEÇÃO
C3, A3, E3. Erro ao realizar cadastro, alteração ou exclusão de conta. C3, A3, E3.1. O sistema apresenta um erro ao cadastrar, alterar ou excluir as informações da conta selecionada e exibe uma mensagem de erro.

Fonte: Autoria Própria (2025).

Quadro 3 – Selecionar Parada

[UC02] Selecionar Parada	
Atores:	Passageiro e Funcionário.
Pré-Condições:	O usuário logado no sistema e pelo menos uma rota com uma parada cadastrada nela. Além de precisar ter um funcionário ou passageiro cadastrado e logado no sistema.
Pós-Condições:	A parada selecionada, com suas informações exibidas
FLUXO PRINCIPAL	
1. O usuário seleciona uma parada em determinada rota. 2. O sistema exibe as informações dessa parada	

FLUXOS ALTERNATIVOS
Não há.
FLUXOS DE EXCEÇÃO
2. Erro ao comunicar com o banco de dados. 2.1. O sistema falha ao se comunicar com o banco de dados e dá um erro.

Fonte: Autoria Própria (2025).

Quadro 4 – Selecionar Ônibus

[UC03] Selecionar Ônibus	
Atores:	Passageiro e Funcionário.
Pré-Condições:	O usuário logado no sistema e pelo menos uma rota com um ônibus sendo rastreado nela. Além de precisar ter um funcionário ou passageiro cadastrado e logado no sistema.
Pós-Condições:	O ônibus selecionado com suas informações exibidas.
FLUXO PRINCIPAL	
1. O usuário seleciona um ônibus em determinada rota. 2. O sistema exibe as informações desse ônibus.	
FLUXOS ALTERNATIVOS	
Não há.	
FLUXOS DE EXCEÇÃO	
2. Erro ao comunicar com o banco de dados. 2.1. O sistema falha ao se comunicar com o banco de dados e dá um erro.	

Fonte: Autoria Própria (2025).

Quadro 5 – Realizar Login

[UC04] Realizar login	
Atores:	Passageiro e Funcionário.
Pré-Condições:	O passageiro ou funcionário cadastrado no sistema.
Pós-Condições:	O passageiro ou funcionário logado no sistema.
FLUXO PRINCIPAL	
1. O sistema exibe o tela de login para os usuários. 2. O usuário informa as suas credenciais “email e senha” e solicita a entrada no sistema.	

3. O sistema verifica que o usuário está cadastrado no sistema e se suas credenciais estão corretas. 4. O sistema redireciona o usuário para a sua tela inicial.
FLUXOS ALTERNATIVOS
2. O usuário esqueceu sua senha. 2.1. O usuário esquece sua senha e solicita o formulário de recuperação de senha. 2.2. O sistema exibe o formulário de recuperação. 2.3. O usuário informa o seu email cadastrado no sistema para a recuperação de senha. 2.4. O sistema verifica se o e-mail informado está cadastrado no sistema e envia um email para a recuperação de senha. 2.5 O usuário responde o formulário de recuperação de senha com sua nova senha. 2.6 O sistema altera a senha do usuário.
FLUXOS DE EXCEÇÃO
2. O usuário informa dados errados ou inexistentes.. 2.1. O sistema verifica que os dados estão errados e ou não existem na base de dados e exibe uma mensagem.

Fonte: Autoria Própria (2025).

Quadro 6 – Acessar Ouvidoria

[UC05] Acessar Ouvidoria	
Atores:	Passageiro e Funcionário.
Pré-Condições:	O usuário logado no sistema.
Pós-Condições:	O relato de um problema do sistema.
FLUXO PRINCIPAL	
1. O usuário percebe algum problema no sistema e acessa a ouvidoria para relatá-lo. 2. O sistema envia o usuário para o Whatsapp do administrador do sistema. 3. O usuário relata os problemas encontrados.	
FLUXOS ALTERNATIVOS	
Não há.	
FLUXOS DE EXCEÇÃO	

Não há

Fonte: Autoria Própria (2025).

Quadro 7 – Acompanhar Rotas

[UC06] Acompanhar Rotas	
Atores:	Passageiro e Funcionário.
Pré-Condições:	Uma rota com pelo menos 2 pontos cadastrada no sistema. Além de precisar ter um funcionário cadastrado e logado no sistema.
Pós-Condições:	Uma rota com paradas, uma linha e ônibus sendo exibidos.
FLUXO PRINCIPAL	
1. O usuário seleciona uma rota a ser exibida. 2. O sistema busca pelo ID da rota e mostra a linha, paradas e ônibus atrelados à ela.	
FLUXOS ALTERNATIVOS	
Não há.	
FLUXOS DE EXCEÇÃO	
2. Erro ao comunicar com o banco de dados. 2.1. O sistema falha ao se comunicar com o banco de dados e dá um erro.	

Fonte: Autoria Própria (2025).

Quadro 8 – Manter Paradas

[UC07] Manter Paradas	
Atores:	Funcionário.
Pré-Condições:	Para cadastrar uma parada no sistema é necessário estar cadastrando uma rota. Para editar, listar e excluir uma parada, ela precisa estar cadastrada no sistema. Além de precisar ter um funcionário cadastrado e logado no sistema.
Pós-Condições:	Uma parada cadastrada, listada, editada ou excluída do sistema.
FLUXO PRINCIPAL	
Cadastrar Parada: C1. O funcionário solicita o cadastro ou edição de uma rota para o sistema.	

C2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. C3. O funcionário seleciona o lugar no mapa que quer adicionar a parada e clica para salvar. C4. O sistema registra a parada na rota selecionada. Alterar Parada: A1. O funcionário solicita o cadastro ou edição de uma rota para o sistema. A2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. A3. O funcionário seleciona a parada que quer alterar a posição, seleciona a sua nova posição e clica para salvar. A4. O sistema altera a posição da parada na rota selecionada. Excluir Parada: E1. O funcionário solicita o cadastro ou edição de uma rota para o sistema. E2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. E3. O funcionário seleciona a parada que quer excluir. E4. O sistema solicita a confirmação de exclusão da parada. E5. O funcionário confirma a exclusão da parada e clica para salvar. E6. O sistema realiza a exclusão da parada na rota selecionada. Listar Paradas: L1. O funcionário solicita o cadastro, edição ou visualização de uma rota para o sistema. L2. O sistema exibe a tela de cadastro, edição ou visualização da rota para o funcionário com as suas respectivas paradas.
FLUXOS ALTERNATIVOS
E3. O funcionário seleciona a opção de cancelar. E3.1. O sistema exibe a tela da rota da parada com suas respectivas paradas.
FLUXOS DE EXCEÇÃO
C3, A3, E3. Erro ao realizar cadastro, alteração ou exclusão da parada.

C3, A3, E3.1. O sistema apresenta um erro ao cadastrar, alterar ou excluir as informações da parada selecionada e exibe uma mensagem de erro.

Fonte: Autoria Própria (2025).

Quadro 9 – Manter Pontos da Rota

[UC08] Manter Pontos da Rota	
Atores:	Funcionário.
Pré-Condições:	Para cadastrar um ponto de uma rota no sistema é necessário estar cadastrando uma rota. Para editar, listar e excluir um ponto de uma rota, o ponto precisa estar cadastrado no sistema. Além de precisar ter um funcionário cadastrado e logado no sistema.
Pós-Condições:	Um ponto de uma rota cadastrado, listado, editado ou excluído do sistema.
FLUXO PRINCIPAL	
Cadastrar Ponto da Rota: C1. O funcionário solicita o cadastro ou edição de uma rota para o sistema. C2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. C3. O funcionário seleciona o lugar no mapa que quer adicionar o ponto da rota e clica para salvar. C4. O sistema registra o ponto da rota selecionada. Alterar Ponto da Rota: A1. O funcionário solicita o cadastro ou edição de uma rota para o sistema. A2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. A3. O funcionário seleciona o ponto da rota que quer alterar a posição, seleciona a sua nova posição e clica para salvar. A4. O sistema altera a posição do ponto na rota selecionada. Excluir Ponto da Rota: E1. O funcionário solicita o cadastro ou edição de uma rota para o sistema.	

E2. O sistema exibe a tela de cadastro ou edição da rota de cadastro para o funcionário. E3. O funcionário seleciona o ponto da rota que quer excluir. E4. O sistema solicita a confirmação de exclusão do ponto da rota. E5. O funcionário confirma a exclusão do ponto da rota e clica para salvar. E6. O sistema realiza a exclusão do ponto na rota selecionada. Listar Pontos da Rota: L1. O funcionário solicita o cadastro ou edição de uma rota para o sistema. L2. O sistema exibe a tela de cadastro ou edição da rota para o funcionário com os seus respectivos pontos.
FLUXOS ALTERNATIVOS
E3. O funcionário seleciona a opção de cancelar. E3.1. O sistema exibe a tela da rota dos com seus respectivos pontos.
FLUXOS DE EXCEÇÃO
C3, A3, E3. Erro ao realizar cadastro, alteração ou exclusão do ponto na rota. C3, A3, E3.1. O sistema apresenta um erro ao cadastrar, alterar ou excluir as informações do ponto na rota selecionada e exibe uma mensagem de erro.

Fonte: Autoria Própria (2025).

Quadro 10 – Manter Rotas

[UC09] Manter Rotas	
Atores:	Funcionário.
Pré-Condições:	Para cadastrar uma rota no sistema não há pré-condições, mas para editar, listar e excluir informações da rota, a rota deve estar cadastrada e logada no sistema. Além de precisar ter um funcionário cadastrado e logado no sistema.
Pós-Condições:	Uma rota cadastrada, excluída, editada ou listada no sistema.
FLUXO PRINCIPAL	
Cadastrar Rota: C1. O funcionário solicita a tela de cadastro de rotas. C2. O sistema exibe a tela de cadastro de rotas para o funcionário. C3. O funcionário adiciona as informações da rota e solicita o cadastro da rota.	

C4. O sistema registra a rota e redireciona o funcionário para a tela inicial. Alterar Rota: A1. O funcionário acessa a tela de edição da rota selecionada. A2. O sistema exibe as informações da rota para o funcionário. A3. O funcionário realiza as alterações necessárias na rota e solicita o registro dessas alterações. A4. O sistema registra as alterações realizadas pelo funcionário na rota. Excluir Rota: E1. O usuário acessa a página inicial. E2. O sistema exibe as rotas cadastradas no sistema e disponibiliza a opção de exclusão das rotas. E3. O usuário seleciona a opção de exclusão da rota. E4. O sistema solicita a confirmação de exclusão da rota. E5. O usuário confirma a exclusão da rota. E6. O sistema realiza a exclusão da rota. Listar Rotas: L1. O usuário acessa a página inicial. L2. O sistema realiza a listagem de todas as rotas cadastradas no sistema.
FLUXOS ALTERNATIVOS
E3. O usuário seleciona a opção de cancelar. E3.1. O sistema exibe as informações da rota e disponibiliza a opção de exclusão da conta.
FLUXOS DE EXCEÇÃO
C3, A3, E3. Erro ao realizar cadastro, alteração ou exclusão de rota. C3, A3, E3.1. O sistema apresenta um erro ao cadastrar, alterar ou excluir as informações da rota selecionada e exibe uma mensagem de erro.

Fonte: Autoria Própria (2025).

Quadro 11 – Manter Usuários

[UC10] Manter Usuários

Atores:	Funcionário.
Pré-Condições:	Para cadastrar um usuário no sistema não há pré-condições, mas para editar, listar e excluir informações do usuário, o usuário deve estar cadastrado e logado no sistema. Além de precisar ter um funcionário cadastrado e logado no sistema.
Pós-Condições:	Um usuário cadastrado, excluído, editado ou listado no sistema.
FLUXO PRINCIPAL	
Cadastrar Usuário: C1. O funcionário solicita o formulário de cadastro de usuário. C2. O sistema exibe o formulário de usuário. C3. O funcionário preenche o formulário com as informações da conta e solicita o cadastro no sistema. C4. O sistema registra a conta e envia um email de verificação de email para o usuário. C5. O funcionário verifica o email. C6. O sistema libera o acesso do funcionário ao sistema. Alterar Usuário: A1. O funcionário seleciona um usuário e solicita o formulário para editar o usuário. A2. O sistema exibe o formulário com as informações do usuário. A3. O funcionário altera as informações do usuário e solicita o registro. A4. O sistema registra as informações alteradas e exibe uma mensagem. Excluir Usuário: E1. O funcionário seleciona um usuário e solicita a exclusão do usuário. E2. O sistema solicita a confirmação da exclusão do usuário. E3. O funcionário confirma a exclusão do usuário. E4. O sistema exclui o usuário e apresenta uma mensagem. Listar Usuários: L1. O funcionário solicita a lista de usuários do sistema. L2. O sistema exibe a lista de usuários do sistema.	

FLUXOS ALTERNATIVOS
E3. O funcionário seleciona a opção de cancelar. E3.1. O sistema exibe as informações da conta do usuário e disponibiliza a opção de exclusão da conta.
FLUXOS DE EXCEÇÃO
C3, A3, E3. Erro ao realizar cadastro, alteração ou exclusão de conta. C3, A3, E3.1. O sistema apresenta um erro ao cadastrar, alterar ou excluir as informações da conta selecionada e exibe uma mensagem de erro.

Fonte: Autoria Própria (2025).

Quadro 12 – Mostrar Horários da Rota

[UC11] Mostrar Horários da Rota	
Atores:	Funcionário e Passageiro.
Pré-Condições:	Para mostrar os horários da rota é necessário um usuário logado e uma rota cadastrada com seus respectivos horários.
Pós-Condições:	Horários da rota exibidos.
FLUXO PRINCIPAL	
1. O usuário acompanha uma linha e seleciona o ícone de horários. 2. O sistema exibe os horários da respectiva rota.	
FLUXOS ALTERNATIVOS	
Não há.	
FLUXOS DE EXCEÇÃO	
2. Erro ao comunicar com o banco de dados. 2.1. O sistema falha ao se comunicar com o banco de dados e dá um erro.	

Fonte: Autoria Própria (2025).

3.3 TECNOLOGIAS UTILIZADAS

Esta seção apresenta as tecnologias adotadas no desenvolvimento do aplicativo e descreve como cada uma delas contribuiu para solucionar problemas específicos, implementar funcionalidades essenciais e garantir o atendimento aos requisitos propostos pelo sistema. A seleção dessas ferramentas foi resultado de uma análise que considerou fatores como desempenho, facilidade de manutenção,

curva de aprendizagem, integração com o ecossistema Android e compatibilidade com as necessidades funcionais do projeto.

3.3.1 Kotlin

Kotlin foi a linguagem central utilizada no desenvolvimento por ser moderna, concisa e segura. Sua sintaxe clara facilitou a implementação das regras de negócio, validação de dados, manipulação de estados e integração com o Firebase. Além disso, sua interoperabilidade com o ecossistema Android permitiu a integração eficiente de mapas, navegação e demais bibliotecas utilizadas, resultando em um código mais organizado, compreensível e de fácil manutenção. Além de fazer parte dos conteúdos da matéria de Tópicos Emergentes na Informática do ano de 2025, facilitando a compreensão da linguagem.

3.3.2 Jetpack Compose

A construção da interface do aplicativo foi realizada utilizando o Jetpack Compose, tecnologia moderna que substitui os tradicionais arquivos XML por uma abordagem totalmente declarativa. Nesse modelo, a interface é construída a partir de funções (@Composable) que descrevem como a tela deve se parecer de acordo com o estado atual do sistema, permitindo que a UI se atualize automaticamente sempre que esses estados forem modificados. Isso torna o desenvolvimento mais intuitivo e reduz a necessidade de código repetitivo. Além disso, o Compose facilita a criação de telas responsivas, capazes de se adaptar à rotação e a diferentes tamanhos de dispositivos, ao mesmo tempo em que oferece pré-visualização em tempo real e componentes reutilizáveis que aceleram o desenvolvimento, o que ajuda na modelagem da interface. Essa tecnologia proporcionou maior fluidez e consistência visual ao sistema.

3.3.3 Firebase Authentication

O Firebase Authentication foi utilizado para implementar a autenticação segura de passageiros e administradores dentro do aplicativo. Por meio desse serviço, tornou-se possível realizar o cadastro de usuários, validar credenciais de acesso e empregar diferentes funções de autenticação já fornecidas pela plataforma. A solução ofereceu um processo de *login* seguro, padronizado e de forma fácil, eliminando a necessidade de construir e manter um servidor próprio dedicado à autenticação. Dessa forma, o Firebase Authentication contribuiu de maneira significativa para a robustez e a segurança do fluxo de acesso no sistema.

3.3.4 Firebase Firestore

O Firestore foi adotado como banco de dados principal do sistema por oferecer uma estrutura flexível baseada em coleções e documentos, facilitando o armazenamento de usuários, paradas, rotas, ônibus e permissões. Sua sincronização em tempo real permitiu que atualizações fossem refletidas imediatamente no mapa e nas telas do aplicativo, garantindo precisão nas informações exibidas. Além disso, a integração direta com Kotlin e Jetpack Compose simplificou o desenvolvimento das funcionalidades de gerenciamento, como Manter Paradas e Manter Pontos da Rota. Um ponto essencial foi sua capacidade de interligar os dois aplicativos do projeto, do motorista e o do passageiro, permitindo que ambos trocassem informações de forma contínua e confiável. Apesar de não ser um banco relacional, sua organização hierárquica tornou mais simples visualizar a estrutura do sistema, como rotas contendo suas respectivas paradas e pontos, além de facilitar a consulta e o carregamento desses dados durante o uso do aplicativo.

3.3.5 Firebase Realtime Database

O Firebase Realtime Database foi utilizado como um banco de dados complementar no projeto, com foco no armazenamento e na atualização contínua de

informações em tempo real. Trata-se de um banco NoSQL baseado em estrutura JSON, no qual os dados são organizados de forma hierárquica, permitindo fácil leitura e escrita. Sua principal contribuição para o sistema foi possibilitar o envio e a recepção imediata de dados dinâmicos, como a localização, status, velocidade e horário de atualização dos ônibus. Sempre que uma informação é alterada no banco, todos os dispositivos conectados recebem essa atualização instantaneamente, sem a necessidade de novas requisições. Essa característica foi essencial para o acompanhamento em tempo real dos veículos no mapa, garantindo maior precisão e confiabilidade ao usuário. Além disso, a integração direta com Kotlin facilitou o consumo desses dados no aplicativo, tornando a comunicação entre o aplicativo do motorista e o do passageiro rápida, estável e eficiente.

3.3.6 OSMdroid

A biblioteca OSMdroid desempenhou o papel mais importante no desenvolvimento do aplicativo ao permitir a exibição de mapas utilizando dados abertos do OpenStreetMap, eliminando a necessidade de serviços pagos ou proprietários. Com ela, foi possível implementar a visualização completa das rotas, exibir paradas cadastradas e atualizar em tempo real a posição dos ônibus, garantindo uma experiência dinâmica e intuitiva ao usuário. Além disso, sua ampla variedade de recursos, como a criação de *polylines*[5] para representar trajetos, a adição de *markers*[6] personalizados para identificar paradas e a renderização de múltiplas camadas visuais, tornou o desenvolvimento mais flexível e eficiente. Essas funcionalidades permitiram representar facilmente as linhas, curvas e pontos das rotas, melhorando a clareza visual e a compreensão do mapa. A escolha do OSMdroid atendeu com êxito ao requisito de acompanhamento das rotas, oferecendo um conjunto robusto de ferramentas avançadas sem custo adicional e garantindo controle total sobre o comportamento e personalização do mapa dentro do aplicativo.

---

[5] Uma *polyline* é uma linha formada pela conexão de vários pontos geográficos no mapa. Ela é utilizada para representar trajetos, rotas ou caminhos de forma visual, permitindo ao usuário identificar claramente o percurso dos ônibus.

[6] Um *marker* é um ícone ou ponto fixo exibido no mapa para indicar um local específico, como paradas de ônibus. Ele funciona como um marcador visual que destaca elementos importantes da rota.

3.3.7 Navigation Compose

A biblioteca Navigation Compose foi empregada para gerenciar a navegação entre as telas do aplicativo de forma clara, organizada e totalmente integrada ao Jetpack Compose. Com ela, foi possível estruturar fluxos consistentes, garantindo que cada ação do usuário, como clicar em botões ou selecionar opções, enviasse corretamente informações para outras telas sempre que necessário. Sua abordagem declarativa tornou a configuração das rotas mais simples e modular, reduzindo erros e facilitando a manutenção do código. Além disso, o Navigation Compose assegurou transições estáveis e previsíveis, contribuindo para uma experiência de uso fluida e bem organizada ao longo de todo o aplicativo.

3.3.8 AndroidX Lifecycle e LiveData

As bibliotecas AndroidX Lifecycle e LiveData foram fundamentais para implementar uma lógica reativa e organizada dentro do aplicativo. Elas permitiram que os dados provenientes do Firebase fossem observados de forma contínua, garantindo que qualquer mudança no banco fosse automaticamente refletida nos componentes gráficos sem necessidade de atualizações manuais. Além disso, proporcionaram uma separação entre interface e regras de negócio, tornando o código mais modular e fácil de manter. Graças a esses recursos, o aplicativo pôde lidar com estados e atualizações de forma consistente, mesmo diante de mudanças de contexto, como alternância entre telas ou diferentes ciclos de vida dos componentes.

3.4 O SISTEMA

Nesta seção, serão mostradas as partes mais importantes e que envolvem mais lógica do sistema com seus respectivos nomes e descrição de funcionamento.

3.4.1 Figuras das interfaces

Abaixo serão apresentadas as figuras e as explicações das interfaces mais importantes do sistema.

A Figura 2, que apresenta a tela de registro do aplicativo, funciona a partir da integração direta entre Kotlin, Jetpack Compose, Firebase Authentication, Firebase Firestore, Navigation Compose e os componentes de AndroidX Lifecycle/LiveData, formando um fluxo seguro, reativo e bem estruturado. Em Kotlin, toda a lógica é organizada de forma clara, permitindo validar campos, gerenciar estados e se comunicar eficientemente com o ViewModel. A interface construída em Jetpack Compose renderiza dinamicamente cada elemento, campos de entrada, botões e mensagens de estado, atualizando-se automaticamente conforme o ViewModel altera o estado da tela.

O Firebase Authentication é responsável por criar contas, registrar usuários e enviar o e-mail de verificação, utilizando seus recursos nativos de autenticação segura. Paralelamente, o Firebase Firestore armazena dados complementares, como nome, tipo de acesso e status da conta, garantindo persistência confiável no banco de dados. O Navigation Compose controla a navegação pós-cadastro, direcionando o usuário para a tela apropriada após a verificação por e-mail. Já AndroidX Lifecycle e LiveData permitem que a interface observe continuamente o progresso da autenticação, refletindo de forma automática e segura transições entre carregamento, sucesso, erro e confirmação de e-mail.

Assim, a tela de registro combina uma interface declarativa, lógica reativa e serviços robustos de autenticação e armazenamento em nuvem, oferecendo um fluxo de cadastro estável, responsivo e intuitivo.

Figura 2 – Tela de Cadastro.

Fonte: Autoria Própria (2025).

A Figura 3, que apresenta a tela *Home* do administrador, integra diversos componentes do Android e serviços do Firebase para exibir, gerenciar e editar as rotas cadastradas no sistema de maneira dinâmica e responsiva. O MapViewModel coordena toda a comunicação com o Firestore, buscando rotas, convertendo seus pontos em *polylines* e atualizando o mapa em tempo real sempre que alguma mudança ocorre. O carregamento das rotas é automático, enquanto a exclusão é realizada pelo próprio ViewModel, que identifica o documento correspondente no Firestore, remove-o e sincroniza imediatamente tanto o mapa quanto a lista exibida na interface.

A interface construída com Jetpack Compose organiza o conteúdo em um layout moderno, combinando menu lateral, mapa em tela cheia e um card inferior com a lista das linhas. O OSMDroid é responsável pela renderização do mapa, apresentando cada rota com duas camadas de linha (uma base preta e outra na cor principal) para melhorar a visibilidade. A lista permite visualizar, editar e excluir rotas, enquanto o Navigation Compose gerencia a transição entre telas de edição e visualização. O Firebase Authentication garante um processo seguro de logout do administrador. Dessa forma, a tela reúne visualização geográfica, ferramentas de gerenciamento, acesso em tempo real ao Firestore e navegação fluida para oferecer um painel administrativo eficiente e intuitivo.

Figura 3 – Tela Inicial – Funcionário.

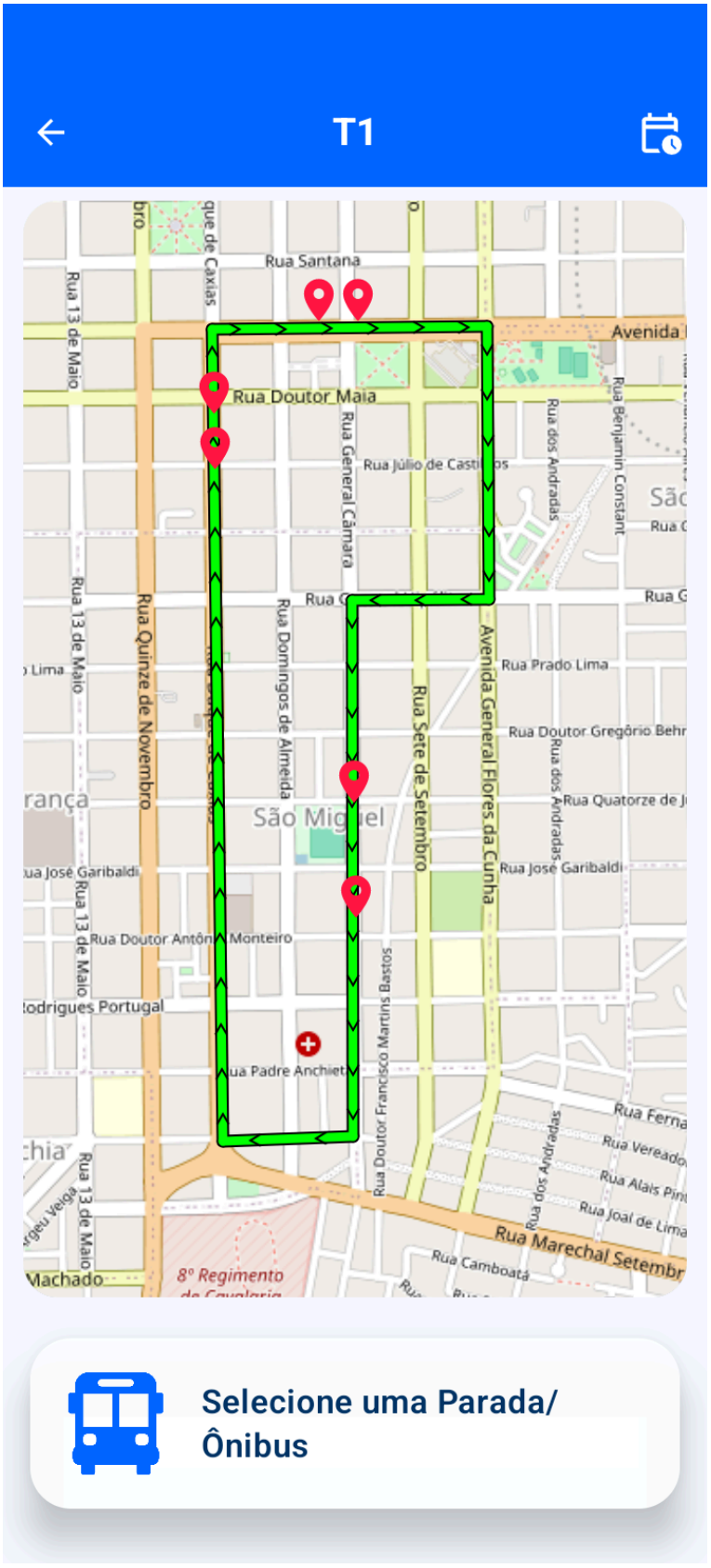
Fonte: Autoria Própria (2025).

A Figura 4, que apresenta a RouteScreen, é a principal visualização detalhada de uma linha de ônibus no aplicativo, combinando Jetpack Compose e o mapa OSMDroid para criar uma experiência interativa e atualizada em tempo real. O RouteViewModel centraliza toda a comunicação com o Firebase, usando Firestore para rotas e paradas e Realtime Database para acompanhar a posição dos ônibus, realizando atualizações a cada 2 segundos e aplicando projeções geométricas para alinhar com precisão os marcadores dos veículos à polyline da rota.

O mapa apresenta a trajetória com duas camadas de linha (uma base preta mais larga e outra mais fina na cor da rota), setas direcionais distribuídas ao longo do percurso e ícones personalizados para paradas (vermelhos, ampliados quando selecionados) e ônibus (coloridos conforme seu status). Ao tocar em uma parada, o sistema calcula e exibe, na barra inferior, o tempo estimado de chegada do próximo ônibus, atualizado a cada 3 segundos com base em velocidade, distância restante e até possíveis voltas completas no trajeto. Um card flutuante animado revela os horários por dia da semana ao tocar no ícone superior.

Com uma interface moderna, TopBar personalizada, estados de carregamento, tratamento de erros e navegação fluida via NavController, a tela permite explorar rotas, acompanhar ônibus em movimento e acessar informações práticas de maneira intuitiva e responsiva. O resultado é uma experiência robusta que une renderização avançada no mapa, cálculos geométricos de precisão e atualizações em tempo real do Firebase para um monitoramento confiável do transporte público.

Figura 4 – Tela de Linha.



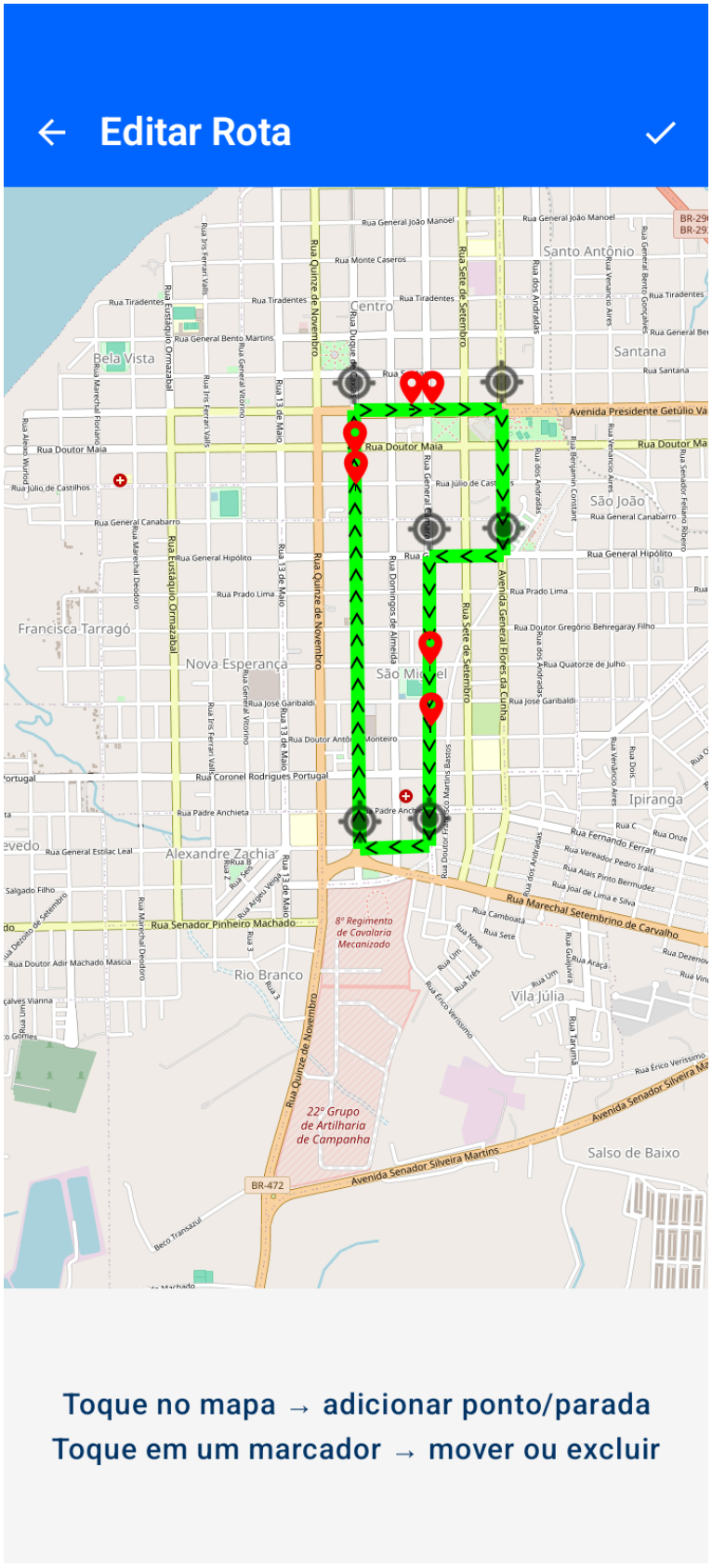
Fonte: Autoria Própria (2025).

A Figura 5, que apresenta tela RouteEditScreenAdm, é construída com Jetpack Compose, oferece uma interface declarativa, responsiva e totalmente orientada a estados para que administradores possam criar ou editar rotas de transporte de forma intuitiva. Seu comportamento é guiado pelo RouteEditViewModel, que centraliza toda a lógica essencial do editor e garante que qualquer alteração realizada na interface reflita imediatamente no mapa e nos componentes visuais.

No gerenciamento da geometria da rota, o ViewModel controla operações como adicionar, mover e remover pontos do traçado, sempre mantendo a poligonal fechada, sincronizando automaticamente o primeiro e o último ponto. Da mesma forma, supervisiona todo o ciclo de vida das paradas, permitindo adicioná-las, reposicioná-las ou removê-las. Cada parada passa por um cálculo de distância acumulada ao longo da linha, garantindo a sua ordenação automática conforme a posição real no percurso. Para manter a precisão geométrica, a tela utiliza funções que asseguram projeções corretas, alinhamento sobre a linha e orientação consistente das setas direcionais.

Ao ser inicializada, a tela carrega do Firebase Firestore os dados já existentes da rota, e todas as modificações, incluindo pontos, paradas, nome, cor e horários, são salvas de forma assíncrona, acompanhadas das validações adequadas. A visualização da rota ocorre sobre um mapa interativo baseado em OSMdroid, que exibe a polyline colorida, marcadores, setas de direção e demais elementos gráficos. Além disso, diálogos construídos em Compose permitem definir configurações antes da gravação dos dados, enquanto a navegação é gerenciada pelo Navigation Compose, oferecendo uma experiência fluida, consistente e eficiente para o gerenciamento administrativo de rotas.

Figura 5 – Tela de Edição de Linha – Funcionário.



Fonte: Autoria Própria (2025).

3.4.2 Figuras das funções

Abaixo serão apresentadas as figuras e as explicações das funções mais importantes do sistema.

A Figura 6 apresenta o RouteEditViewModel, dentro dele o processo de adicionar pontos na rota funciona de forma organizada para manter sempre a poligonal fechada e consistente. Quando um novo ponto é inserido, o sistema identifica que o último ponto da lista é sempre uma cópia do primeiro e, por isso, coloca o novo ponto imediatamente antes dele. Caso seja o primeiro ponto criado, o ViewModel gera automaticamente um ponto duplicado para garantir o fechamento da rota desde o início. Sempre que um ponto é adicionado, a lista é reorganizada, os IDs são atualizados e a polyline é reconstruída, mantendo a rota visualmente correta no mapa.

Figura 6 – Função adicionarPonto – RouteEditViewModel.

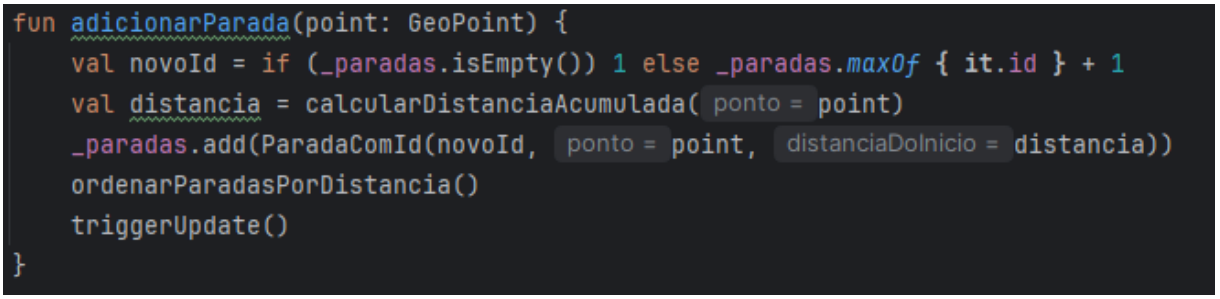
```
fun adicionarPonto(point: GeoPoint) {
    when {
        _pontos.isEmpty() -> {
            _pontos.add(PontoComId( id = 1,  ponto = point))
            _pontos.add(PontoComId( id = 2,  ponto = point))
        }
        else -> {
            val novoId = _pontos.maxOf { it.id } + 1
            if (_pontos.size >= 2 && _pontos.last().ponto == _pontos.first().ponto) {
                _pontos.add( index = _pontos.size - 1,  element = PontoComId(novoId,  ponto = point))
            } else {
                _pontos.add(PontoComId(novoId,  ponto = point))
            }
        }
    }
    reordenarIdsPontos()
    triggerUpdate()
}
```

Fonte: Autoria Própria (2025).

A Figura 7 apresenta a função adicionarParada, cujo a adição de paradas segue um procedimento criterioso, baseado em cálculos geométricos. Ao tocar no mapa, o ViewModel não cria a parada exatamente na posição clicada; primeiro ele identifica o trecho da rota mais próximo e calcula a distância até ele. Caso essa

distância seja superior a 50 metros, a parada é descartada por estar fora da rota. Se estiver dentro do limite, o sistema projeta a parada exatamente sobre a polyline, garantindo alinhamento perfeito. Em seguida, calcula a distância acumulada ao longo da rota para determinar a posição correta dessa parada na ordem. Assim, todas as paradas permanecem alinhadas, organizadas e coerentes com o percurso real.

Figura 7 – Função adicionarParada – RouteEditViewModel.



```
fun adicionarParada(point: GeoPoint) {
    val novoId = if (_paradas.isEmpty()) 1 else _paradas.maxOf { it.id } + 1
    val distancia = calcularDistanciaAcumulada( ponto = point)
    _paradas.add(ParadaComId(novoId,  ponto = point,  distanciaDoInicio = distancia))
    ordenarParadasPorDistancia()
    triggerUpdate()
}
```

Fonte: Autoria Própria (2025).

A Figura 8 apresenta a função calcularTempoParaParada, essa função calcula, de forma inteligente, o tempo estimado para um ônibus chegar até uma parada específica, analisando a posição e a velocidade de todos os veículos que estão circulando naquela rota. Primeiro, ela determina a posição da parada ao longo da rota e obtém também o comprimento total da linha. Em seguida, para cada ônibus, calcula sua distância acumulada na polyline, compara com a posição da parada e verifica se o veículo já passou dela, caso tenha passado mais de 50 metros, o sistema considera automaticamente que o ônibus chegará apenas na próxima volta, somando o comprimento total da rota à distância restante. Depois disso, calcula o quanto falta para o ônibus atingir a parada e usa sua velocidade (com um valor mínimo aplicado para evitar estimativas irrealistas) para converter essa distância em minutos. Entre todos os veículos avaliados, o algoritmo escolhe o menor tempo e o transforma em uma mensagem amigável, como “Chegando”, “1 min”, ou “>2h”, retornando sempre a previsão mais favorável para o passageiro. Dessa forma, a função oferece uma estimativa precisa e adaptada ao comportamento real dos ônibus na rota.

Figura 8 – Função calcularTempoParaParada – RouteViewModel.

```
fun calcularTempoParaParada(paradaPosition: GeoPoint, onibusList: List<OnibusInfo>, pontos: List<GeoPoint>): Stri
    if (onibusList.isEmpty() || pontos.size < 2) return "Indisponível"

    val distParada = calcularDistanciaAcumuladaDoOnibusAtual( busPosition = paradaPosition, pontos)
    val comprimentoTotal = calcularVertexCumDists(pontos).last()
    var menorTempo = Double.MAX_VALUE
    var melhorTexto = "Indisponível"

    onibusList.forEach { onibus ->
        val distOnibus = calcularDistanciaAcumuladaDoOnibusAtual( busPosition = onibus.position, pontos)
        var diferenca = distParada - distOnibus

        // Se o ônibus já passou da parada → próxima volta
        if (diferenca < -50) {
            diferenca += comprimentoTotal
        }

        val distanciaRestante = diferenca.coerceAtLeast( minimumValue = 0.0)
        var velocidade = onibus.velocity
        if (velocidade < 5.55) velocidade = 7.0 // 25 km/h mínimo

        val tempoMinutos = if (velocidade > 0) (distanciaRestante / velocidade) / 60.0 else Double.MAX_VALUE

        if (tempoMinutos < menorTempo) {
            menorTempo = tempoMinutos
            melhorTexto = when {
                tempoMinutos < 0.5 -> "Chegando"
                tempoMinutos < 1   -> "<1 min"
                tempoMinutos < 2   -> "1 min"
                tempoMinutos > 120 -> ">2h"
                tempoMinutos > 90  -> "Próxima volta: >90 min"
                tempoMinutos > 60  -> "Próxima volta: ${tempoMinutos.toInt()} min"
                else               -> "${tempoMinutos.toInt()} min"
            }
        }
    }
    return melhorTexto
}
```

Fonte: Autoria Própria (2025).

3.5 MODELAGEM DO BANCO DE DADOS

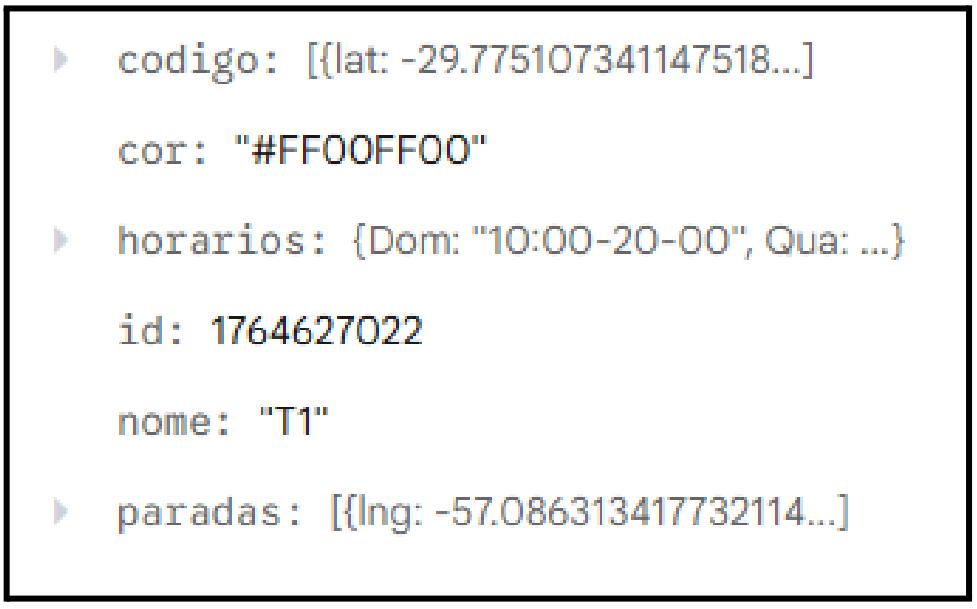
Nesta subseção são apresentadas as tabelas do Banco de Dados do sistema em JSON[7], no formato de árvore de dados, acompanhadas de suas devidas explicações.

[7] JSON é um jeito simples de escrever dados em formato de texto. Ele funciona como uma lista organizada de informações usando chaves e valores, por exemplo: `"nome": "Ana"`. Também permite criar listas e grupos de dados. É fácil de ler, fácil de entender e usado por aplicativos e sites para guardar e enviar informações de forma organizada.

3.5.1 Figuras do banco de dados

A Figura 9 apresenta a tabela de rotas, que representa um documento do Firestore no formato JSON, onde cada coluna mostra um campo armazenado dentro do registro. O campo "codigo" é uma lista de objetos contendo pontos da rota (lat/lng). A “cor” é uma string no formato hexadecimal. O campo "horarios" é um objeto JSON com dias da semana como chaves e faixas de horário como valores. O “id” é um identificador numérico da rota, enquanto "nome" é o nome da linha. Já "paradas" é outra lista de objetos contendo a localização e informações das paradas. Tudo é estruturado como listas e objetos JSON dentro de um único documento no Firestore.

Figura 9 – Tabela de Rotas

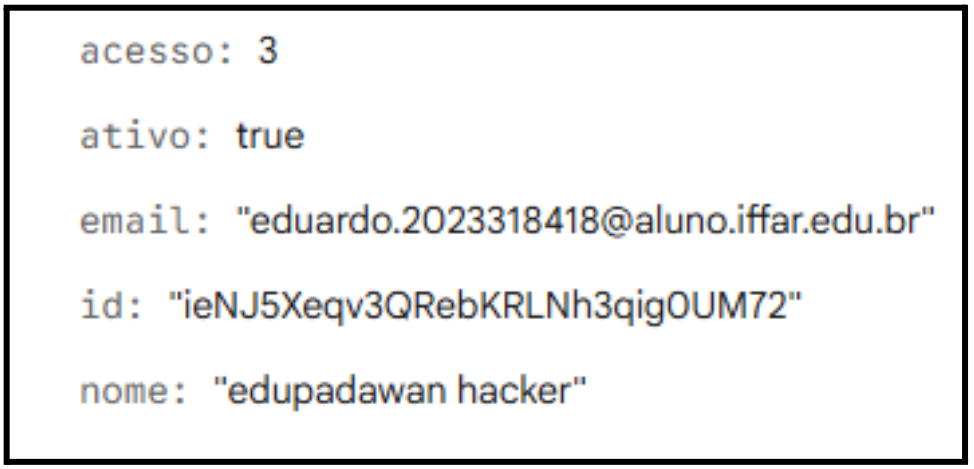


Fonte: Autoria Própria (2025).

A Figura 10 apresenta a tabela usuarios do banco de dados, esse documento do Firestore está estruturado em formato JSON, onde cada campo representa uma informação do usuário armazenada no sistema. O campo "acesso" indica o nível de permissão do usuário, enquanto "ativo" é um valor booleano que mostra se a conta está habilitada. O "email" registra o endereço utilizado no cadastro pelo Firebase Authentication. O "id" corresponde ao identificador único do usuário no Firebase. Já

o campo "nome" armazena o nome associado à conta. Todos esses dados formam um objeto JSON simples que representa um único usuário dentro da coleção.

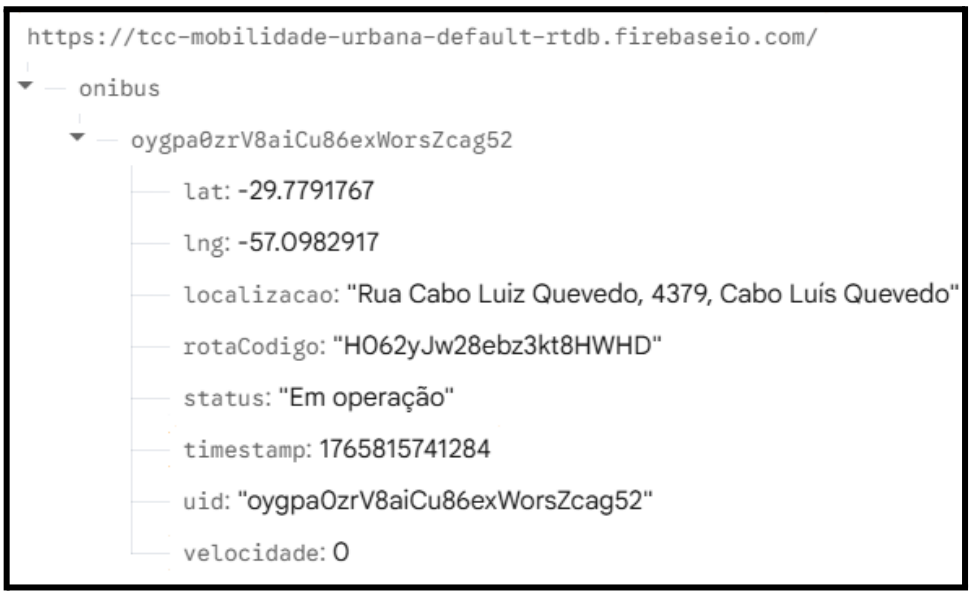
Figura 10 – Tabela de Usuários



Fonte: Autoria Própria (2025).

A Figura 11 apresenta a estrutura da tabela onibus armazenada no Firebase Realtime Database. Esse registro é organizado em formato JSON, no qual cada nó representa uma informação relacionada ao ônibus monitorado pelo sistema. O campo “lat” armazena a latitude e “lng” a longitude, permitindo identificar a posição geográfica atual do veículo no mapa. O campo “localizacao” descreve o endereço aproximado correspondente às coordenadas registradas. Já “rotaCodigo” indica a rota à qual o ônibus está vinculado no sistema. O campo “*status*” informa a situação operacional do veículo, enquanto “velocidade” registra a velocidade atual. O timestamp representa o momento da última atualização dos dados, e o uid corresponde ao identificador único do ônibus no banco. Esses dados, em conjunto, possibilitam o acompanhamento em tempo real da localização e do estado dos ônibus pelos usuários do aplicativo.

Figura 11 – Tabela de Ônibus



Fonte: Autoria Própria (2025).

4 CONSIDERAÇÕES FINAIS

O desenvolvimento deste Trabalho de Conclusão de Curso representou uma grande oportunidade de aplicar, na prática, os conhecimentos adquiridos ao longo do Curso Técnico em Informática, transformando uma demanda real da comunidade de Uruguaiana em uma solução tecnológica funcional. O aplicativo de mobilidade urbana, voltado tanto para o passageiro quanto para a administração de rotas, busca suprir a carência de informações acessíveis sobre o transporte público, oferecendo mapas interativos, horários e visualização de linhas de forma intuitiva e gratuita.

A escolha de tecnologias como Kotlin, Jetpack Compose, Firebase Authentication e Firestore, OSMdroid e Navigation Compose possibilitou a criação de um sistema robusto, reativo e de fácil manutenção. Entre os destaques do desenvolvimento, encontra-se a tela de edição de rotas, que utiliza lógica geométrica avançada, para garantir projeções precisas, alinhamento correto de paradas e orientação fiel das setas direcionais. Apesar dos desafios enfrentados, como cálculos de distância e gerenciamento de estados reativos, os objetivos propostos foram plenamente alcançados, resultando em uma ferramenta que contribui para a acessibilidade, a sustentabilidade urbana e a digitalização de serviços públicos.

Este projeto não apenas consolidou competências técnicas em desenvolvimento Android, como também reforçou a relevância de soluções locais aplicadas a problemas sociais, incentivando o uso do transporte coletivo e facilitando o cotidiano dos cidadãos.

Agradeço pela última vez aos meus orientadores e à instituição por essa jornada enriquecedora, que simboliza o encerramento de uma etapa acadêmica e o início de novas possibilidades profissionais na área da informática. O resultado final demonstra como a tecnologia, quando bem aplicada, é capaz de gerar impactos positivos e significativos na sociedade.

REFERÊNCIAS

BRASIL. Lei nº 12.587, de 3 de janeiro de 2012. Institui as diretrizes da Política Nacional de Mobilidade Urbana. Brasília, DF, 2012. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12587.htm. Acesso em: dez. 2025.

CAMPOS, Vânia Barcellos Gouvêa. Uma Visão da Mobilidade Urbana Sustentável. 2006. Disponível em: Uma visão da mobilidade urbana sustentável. Acesso em: jul. 2025.

RUBIM, Bárbara; LEITÃO, Sérgio. O Plano de Mobilidade Urbana e o Futuro das Cidades. Disponível em: O plano de mobilidade urbana e o futuro das cidades. 2013.Acesso em: jul. 2025.

VACCARI, Lorreine Santos; FANINI, Valter. Mobilidade urbana. Publicações temáticas da Agenda Parlamentar do Conselho Regional de Engenharia, Arquitetura e Agronomia do Paraná–CREA-PR. Curitiba, 2011. Disponível em: Mobilidade Urbana - Crea-PR. Acesso em: jul. 2025.