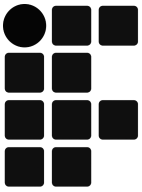


INSTITUTO FEDERAL
Farroupilha
Campus Uruguaiana

CURSO TÉCNICO EM INFORMÁTICA INTEGRADO AO ENSINO MÉDIO

SAMUEL ALVARENGA PRATES

OMBO-MOTORISTA

APLICATIVO DE MOBILIDADE URBANA PARA APOIO AO MOTORISTA NO MONITORAMENTO DO TRANSPORTE PÚBLICO URBANO

URUGUAIANA
2025

SAMUEL ALVARENGA PRATES

OMBO-MOTORISTA

APLICATIVO DE MOBILIDADE URBANA PARA APOIO AO MOTORISTA NO MONITORAMENTO DO TRANSPORTE PÚBLICO URBANO

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

Alvarenga Prates, Samuel.

OMBO-MOTORISTA : APLICATIVO DE MOBILIDADE URBANA PARA APOIO AO MOTORISTA NO MONITORAMENTO DO TRANSPORTE PÚBLICO URBANO / Samuel Alvarenga Prates. — 2025.
[59] f.

Trabalho de Conclusão de Curso Técnico – Instituto Federal de Educação, Ciência e Tecnologia Farroupilha, Uruguaiana, 2025.

1. Mobilidade Urbana. 2. Transporte Público. 3. Geolocalização 4. Aplicativo Mobile. 5. Kotlin. 6. Firebase.

I.OMBO-MOTORISTA: Aplicativo de mobilidade Urbana para Apoio ao Motorista no Monitoramento do Transporte Público Urbano.

CDD [número da CDD].

SAMUEL ALVARENGA PRATES

OMBO-MOTORISTA

APLICATIVO DE MOBILIDADE URBANA PARA APOIO AO MOTORISTA NO MONITORAMENTO DO TRANSPORTE PÚBLICO URBANO

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 DD/MM/AAAA.

BANCA EXAMINADORA

_____

Prof. Ms. Eduardo Henrique Spies
Orientador

_____

Prof. Ms. Michel Michelon
Orientador

_____

Prof.ª Dr.ª Melina Mörschbächer
Avaliadora

_____

Prof. Ms. Rafael Ávila Sides
Avaliador

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

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Silvia e Neil, pelo esforço incansável em proporcionar as condições necessárias para a minha permanência na escola e pelos recursos que garantiram que eu não passasse necessidade nos materiais de estudo. A vocês, que sempre acreditaram em mim, expresso meu amor e minha gratidão. Espero, um dia, ser capaz de retribuir tudo o que fizeram por mim e proporcionar-lhes uma vida confortável e feliz.

Aos meus melhores amigos Daniel Dorneles Façanha, Pedro Gonzalez, Anna Laura Arruda Brasil, Pedro Mena Barreto Rossa, Murillo Machado, Matheo Martins Bessow e Érico da Costa Broglio, que sempre estiveram ao meu lado ao longo desta jornada. Agradeço pela paciência em compreender meus sentimentos tortuosos e minhas escolhas nem sempre acertadas, pelas conversas profundas nos momentos difíceis e pela simples, mas valiosa, companhia nos momentos de descontração. Vocês tornaram essa caminhada mais leve e significativa.

Aos meus orientadores, Eduardo Henrique Spies, Michel Michelon e Stéphane Rodrigues Dias, pelo apoio constante na idealização e realização deste trabalho. Sem a dedicação, orientação e conhecimento compartilhado por vocês, este trabalho não existiria. Agradeço pela paciência, pelos ensinamentos e por acreditarem no potencial deste projeto.

A todos vocês, meu sincero muito obrigado.

Se você vai ter que conviver com você mesmo até o fim, se você vai ter que se aguentar até o fim, se você vai ser espectador de você mesmo até o fim, é melhor que se encante com o que faz.

CLÓVIS DE BARROS FILHO

Sem amor, no entanto, a vida é arrastada, cinza, em preto-e-branco, e é preciso esperar a semana inteira para que um novo e entediante final de semana separe a desgraça da semana anterior da desgraça da próxima semana.

ALBERT EINSTEIN

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo mobile para motoristas de transporte público urbano de Uruguaiana/RS, originado da Prática Profissional Integrada "Design Thinking – Arquitetos do Amanhã". O problema identificado foi a carência de informações acessíveis sobre o transporte coletivo municipal, gerando insegurança e atrasos. O objetivo consistiu em desenvolver um aplicativo que permite o envio de dados de localização em tempo real, gerenciamento do status da linha e comunicação de ocorrências operacionais. A metodologia incluiu levantamento bibliográfico, análise de sistemas similares, especificação de requisitos através de diagramas de casos de uso e implementação utilizando Kotlin. As principais tecnologias empregadas foram Android SDK, Jetpack Compose, Firebase (Authentication, Firestore e Realtime Database), OSMdroid para mapas e Google Play Services Location para GPS. O sistema permite ao motorista selecionar rotas, alterar status do ônibus, realizar rastreamento GPS e acessar ouvidoria. Inclui também painel administrativo para gerenciamento de usuários e moderação de ouvidorias. O aplicativo demonstra como soluções tecnológicas podem melhorar a qualidade do transporte público, promovendo transparência e eficiência no serviço à população.

Palavras-chave: mobilidade urbana; transporte público; geolocalização; aplicativo mobile; Kotlin; Firebase.

ABSTRACT

This work presents the development of a mobile application for urban public transport drivers in Uruguaiana/RS, originated from the Integrated Professional Practice "Design Thinking – Architects of Tomorrow". The identified problem was the lack of accessible information about the municipal public transport system, generating insecurity and delays. The objective was to develop an application that allows real-time location data transmission, route status management, and operational incident communication. The methodology included bibliographic research, analysis of similar systems, requirements specification through use case diagrams, and implementation using Kotlin. The main technologies employed were Android SDK, Jetpack Compose, Firebase (Authentication, Firestore, and Realtime Database), OSMdroid for maps, and Google Play Services Location for GPS. The system allows drivers to select routes, change bus status, perform GPS tracking, and access an ombudsman service. It also includes an administrative panel for user management and ombudsman moderation. The application demonstrates how technological solutions can improve public transport quality, promoting transparency and efficiency in the service provided to the population.

Keywords: urban mobility; public transportation; geolocation; mobile application; Kotlin; Firebase.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS E QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface (Interface de Programação de Aplicações)

CDD – Classificação Decimal de Dewey

CSS – Cascading Style Sheets (Folhas de Estilo em Cascata)

ETEC – Escola Técnica Estadual

FAB – Floating Action Button (Botão de Ação Flutuante)

GPS – Global Positioning System (Sistema de Posicionamento Global)

HTML – HyperText Markup Language (Linguagem de Marcação de Hipertexto)

I/O – Input/Output (Entrada/Saída)

IFFar – Instituto Federal de Educação, Ciência e Tecnologia Farroupilha

JS – JavaScript

JSON – JavaScript Object Notation (Notação de Objetos JavaScript)

JWT – JSON Web Token (Token Web JSON)

NoSQL – Not Only SQL (Não Apenas SQL)

OSM – OpenStreetMap (Mapa Aberto de Ruas)

PPI – Prática Profissional Integrada

SDK – Software Development Kit (Kit de Desenvolvimento de Software)

TCC – Trabalho de Conclusão de Curso

UI – User Interface (Interface do Usuário)

UID – User Identifier (Identificador de Usuário)

UUID – Universally Unique Identifier (Identificador Único Universal)

XML – eXtensible Markup Language (Linguagem de Marcação Extensível)

SUMÁRIO

1 INTRODUÇÃO

O transporte público é um direito social, previsto no artigo sexto da Constituição Federal (Brasil, 1988), sendo essencial para garantir a mobilidade urbana, o acesso ao trabalho, à educação, à saúde e ao lazer. Sua ausência ou má qualidade afeta diretamente o exercício da cidadania e a inclusão social.

A mobilidade urbana está diretamente relacionada à forma como as pessoas se deslocam na cidade e à organização do espaço urbano, influenciando a qualidade de vida da população. Um sistema de transporte público eficiente contribui para a redução das desigualdades sociais, do uso excessivo de veículos particulares e dos impactos ambientais, além de favorecer a integração entre diferentes regiões da cidade. Quando bem planejada, a mobilidade urbana promove inclusão social, permitindo que diferentes grupos tenham acesso aos serviços urbanos de maneira mais justa e equilibrada (Aráujo, 2011).

Em muitos municípios brasileiros, observa-se uma carência significativa na oferta de informações acessíveis sobre o sistema de transporte coletivo. As cidades não possuem um sistema digital eficiente para consulta de rotas, horários e localização dos veículos em tempo real, o que compromete o uso efetivo do serviço por parte da população. Isso ocorre pois essa falta de informações afeta diretamente os usuários, pois gera insegurança, atrasos, perda de tempo e desmotivação no uso do transporte público, dificultando o planejamento dos deslocamentos diários e prejudicando o acesso a compromissos como trabalho, estudo e serviços essenciais.

Um desses municípios é Uruguaiana. Observa-se que, em nossa cidade, mesmo quando o transporte público está disponível, existem carências comunicacionais grandes, dentre as quais pode-se citar dificuldades em garantir que informações básicas, como trajetos das linhas, horários de ônibus, itinerários e eventuais problemas de fornecimento de serviço sejam comunicados de forma clara e atualizada à população.

Nesse sentido, este trabalho tem como objetivo compor uma proposta de solução computacional na forma de um sistema de mobilidade urbana para Uruguaiana, por meio do desenvolvimento de um aplicativo *mobile* que forneça as informações de posição e estado das linhas, na visão do motorista, a fim de garantir que sua contraparte, exiba as rotas, paradas e a localização dos ônibus em tempo real. A proposta busca suprir a carência de ferramentas confiáveis que integrem

essas informações de forma acessível, intuitiva e eficiente, promovendo mais autonomia ao usuário e maior controle sobre o seu tempo.

1.1 JUSTIFICATIVA

Essa proposta surge como um desdobramento do trabalho realizado na PPI (Prática Profissional Integrada) de 2024, chamada Design Thinking, Arquitetos do Amanhã, na qual a Câmara de Vereadores de Uruguaiana, em parceria com o Instituto Federal Farroupilha (IFFar), propôs que estudantes encontrassem soluções para problemas locais. Neste trabalho, foi desenvolvido um site que exibia rotas, paradas e horários dos ônibus do município de Uruguaiana, a fim de melhor publicizar as rotas para a população.

No primeiro momento, os dados foram coletados manualmente, a partir da vivência no transporte público da cidade. No entanto, ao longo do processo, o projeto foi repensado e decidiu-se pela criação de um aplicativo *mobile*, por ser uma alternativa mais prática e acessível à população, além de garantir funcionalidades únicas, como o acompanhamento do tempo até o ônibus chegar na parada mais próxima ou a possibilidade de comunicar prontamente qualquer problema com a linha, para evitar tempo gasto na parada. Para isso, a proposta foi dividida em duas frentes: a visão do motorista, desenvolvida por Samuel Alvarenga Prates, e a visão dos passageiros, que ficou sob responsabilidade de Eduardo Antunes da Silva.

1.2 OBJETIVOS

Nesta seção são descritos os objetivos do trabalho, construídos a fim de garantir a efetivação da proposta.

1.2.1 Objetivo Geral

Desenvolver um aplicativo *mobile* voltado ao uso do motorista de transporte público, integrado a aplicativo de mobilidade urbana, que permita o envio de dados de localização em tempo real, o gerenciamento do status da linha e a comunicação de ocorrências operacionais, possibilitando que essas informações sejam

disponibilizadas ao aplicativo do usuário final e contribuam para um transporte público mais eficiente, transparente e acessível.

1.2.2 Objetivos Específicos

- Implementar um sistema de ativação de viagem, no qual o motorista informe o início da operação ao entrar no ônibus, habilitando o envio contínuo de dados.
- Integrar o uso do GPS do dispositivo móvel para coleta e transmissão da localização do ônibus em tempo real ao sistema central, possibilitando o acompanhamento da rota pelo aplicativo do usuário.
- Permitir ao motorista alterar o status da linha, sinalizando situações como atrasos, interrupções, desvios de rota ou problemas operacionais.
- Assegurar a comunicação entre o aplicativo do motorista e o aplicativo do usuário, garantindo a atualização constante das informações exibidas aos passageiros.
- Desenvolver uma interface simples e intuitiva, adequada ao contexto de uso do motorista, minimizando distrações e facilitando a interação durante a operação do veículo.

1.3 METODOLOGIA

Para a realização deste trabalho que as seguintes etapas foram concluídas:

- **Definir o tema:** Este trabalho teve início dentro da Prática Profissional Integrada (PPI) **Design Thinking – Arquitetos do Amanhã**. A partir dessa atividade, o grupo decidiu seguir com o projeto e dividiu as tarefas entre os integrantes: **Eduardo Antunes da Silva**, que ficou responsável por desenvolver o aplicativo que apresenta visão do passageiro, enquanto que **Samuel Alvarenga Prates** assumiu o aplicativo voltado para a operação dos motoristas.
- **Levantamento de requisitos:** para o cumprimento dessa etapa, foram realizadas:
    - Pesquisas sobre trabalhos acadêmicos correlatos, por meio de buscas em plataformas como o Google Acadêmico, a própria pesquisa do Google e o repositório Arandu do IFFar. Foram selecionados os

trabalhos mais parecidos com este projeto, e os resumos estão apresentados na seção 2.1 do TCC.

- Pesquisas de sistemas implementados semelhantes, a partir da análise de aplicativos e sistemas já existentes que oferecem recursos parecidos com os que se pretende implementar. Essa etapa serviu para entender boas práticas de usabilidade, funções mais utilizadas e problemas que os usuários enfrentam, ajudando a construir um sistema mais eficiente.

- **Especificação de requisitos**
  - Criar wireframes: com base na análise dos requisitos definidos, foram elaborados os wireframes das principais telas do sistema. Os wireframes tem como objetivo representar a estrutura visual do sistema e orientar o desenvolvimento da interface do usuário, garantindo usabilidade e coerência no design.
  - Documentação de casos de uso: os casos de uso foram feitos de forma simples, usando uma tabela para cada função do sistema. Em cada tabela foi descrito quem está usando (motorista ou passageiro), o que ele quer fazer, quais passos ele segue para realizar essa ação e o que precisa acontecer antes e depois disso.

- **Criar o banco de dados:** o banco de dados foi planejado com base nos requisitos do sistema. Primeiro, foram definidos os dados que precisavam ser armazenados e como eles se relacionavam entre si. Em seguida, foi criado um **diagrama do banco de dados**, que mostra as tabelas, os campos de cada tabela (como "Coleção usuários", "Coleção Rotas", "Nó ônibus" e “Coleção ouvidoria”) e como essas tabelas estão ligadas.

- **Desenvolver o sistema: p**ara o desenvolvimento deste aplicativo mobile, foi escolhida a linguagem **Kotlin**, por ser uma linguagem moderna, oficial para o desenvolvimento de aplicativos Android. O Kotlin é simples de escrever, tem uma sintaxe clara e facilita a criação de aplicativos mais seguros e com menos chances de erros.

- **Escrever o trabalho:** o trabalho foi escrito de forma paralela às etapas de pesquisa e planejamento.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os principais conceitos e referências que ajudam a compreender os fundamentos deste trabalho. O conteúdo foi organizado em duas partes: **trabalhos relacionados** e **sistemas semelhantes**. O objetivo é mostrar como outras iniciativas buscaram resolver problemas parecidos no transporte público e quais soluções já existentes podem servir de inspiração para o desenvolvimento deste projeto.

2.1 TRABALHOS RELACIONADOS

Como resultado das pesquisas realizadas nas plataformas Google Acadêmico, Pesquisa Google e no repositório Arandu do IFFar, foram identificados alguns trabalhos com propostas similares. Entre eles, destaca-se o projeto desenvolvido por Gabriela de Sousa Abreu, Giovanna Gabriela de Oliveira Araújo, Letícia de Oliveira Barbosa, Samuel de Andrade Ribeiro e Tatiana Aparecida Debolete Ferrari, estudantes do curso técnico em Administração da ETEC Antônio Devisate, na cidade de Marília-SP.

Abreu et al. (2022) desenvolveram um aplicativo com o objetivo de melhorar o transporte coletivo da cidade, respondendo às principais reclamações da população, como atrasos frequentes e a falta de informações atualizadas sobre horários e rotas. O aplicativo oferece funcionalidades como visualização da grade de horários, localização em tempo real dos ônibus, número de passageiros, valor da tarifa, alertas sobre as rotas, tempo estimado dos percursos e até recursos de acessibilidade, como o *TalkBack*.

Esse projeto serviu de referência, principalmente na forma como centraliza diferentes informações úteis ao usuário em um só lugar. Apesar deste trabalho focar na perspectiva do motorista, a proposta de melhorar a experiência geral no uso do transporte público foi uma inspiração importante.

Outro exemplo analisado foi o artigo Santos (2019) *CrowdBus*, que apresenta um aplicativo colaborativo criado com base na metodologia do ***crowdsourcing*** (ambiente online que utiliza a contribuição coletiva para realizar tarefas, gerar ideias e criar projetos). A proposta busca melhorar a mobilidade urbana a partir da

participação direta dos usuários, que podem compartilhar informações em tempo real sobre os ônibus e o transporte coletivo. O estudo de caso foi feito nas cidades de Maceió e Recife, e mostrou que os usuários estavam insatisfeitos com o sistema atual, revelando a necessidade de soluções inovadoras e participativas.

Esses trabalhos reforçam a importância de desenvolver soluções voltadas às necessidades reais da população. A ideia de facilitar o acesso à informação em tempo real, por exemplo, é algo que este projeto também busca aplicar, mesmo que por meio de um modelo diferente.

2.2 SISTEMAS SEMELHANTES

Cadê o Ônibus? é um aplicativo brasileiro de monitoramento em tempo real do transporte coletivo urbano, com foco na visualização da frota de ônibus em circulação. A plataforma utiliza dados de rastreamento via GPS integrados aos sistemas das operadoras de transporte e disponibilizados por prefeituras, permitindo ao usuário acompanhar a localização dos veículos, consultar horários estimados de chegada e planejar rotas com maior precisão. O sistema opera com base em informações oficiais e atualizadas dinamicamente, oferecendo uma visualização georreferenciada dos pontos de parada e do deslocamento dos ônibus no mapa. Com interface leve e funcional, o aplicativo visa otimizar o tempo de espera do usuário e apoiar a tomada de decisão no deslocamento diário, contribuindo para uma mobilidade urbana mais eficiente e orientada por dados.

A **Uber** é uma plataforma digital de transporte sob demanda que utiliza tecnologias de geolocalização e roteamento em tempo real para intermediar corridas entre motoristas parceiros e usuários. Por meio do aplicativo, o passageiro pode solicitar uma corrida, visualizar a estimativa de tempo e custo, acompanhar o deslocamento do veículo e realizar o pagamento de forma integrada. O sistema opera com base em algoritmos de correspondência entre oferta e demanda, levando em conta a localização dos usuários, tráfego em tempo real e disponibilidade de motoristas. A plataforma também coleta dados de navegação e avaliações para melhorar a eficiência do serviço, oferecer diferentes categorias de transporte e promover segurança na experiência de mobilidade urbana.

Quadro 1 – Sistemas Semelhantes

Sistemas	Pontos Positivos	Pontos Negativos
Cadê o Ônibus?	- Localização em tempo real da frota com dados oficiais das prefeituras - Interface leve, simples e funcional - Ajuda a planejar rotas e reduzir o tempo de espera	- Disponível apenas onde as prefeituras disponibilizam os dados de GPS - Algumas cidades podem ter dados desatualizados
Uber	- Geolocalização precisa e tempo estimado de chegada do veículo - Interface organizada, intuitiva e segura - Excelente gestão de dados de trânsito e navegação	- Foca em transporte individual, não em coletivo - Algoritmos de roteamento não lidam com rotas de ônibus ou paradas
Aplicativo da Equipe GLTS – Marília-SP	- Mostra localização em tempo real da frota - Informa o tempo estimado de percurso e quantidade de passageiros - Alerta sobre mudanças de rota e problemas	- Projeto ainda em desenvolvimento, pode carecer de integração com todos os sistemas oficiais

Fonte: Autoria Própria (2025).

3 DESENVOLVIMENTO

Nesta seção, apresenta as etapas do desenvolvimento do sistema proposto por este trabalho: requisitos funcionais, diagrama de casos de uso, casos de uso, tecnologias utilizadas, figuras de interface, figura de funções e figuras do banco de dados.

3.1 REQUISITOS FUNCIONAIS

Nessa seção são apresentados os requisitos funcionais do sistema, em quadros que facilitam a sua visualização e compreensão.

[RF001] Autenticar Motorista	
Descrição:	O motorista realiza *login* com email e senha. O sistema verifica se o usuário tem acesso nível 2 (motorista).
Prioridade:	□ Essencial ■ Importante □ Desejável

[RF002] Manter Rastreamento	
Descrição:	O motorista seleciona a rota, o status do ônibus e inicia ou para o rastreamento.
Prioridade:	■ Essencial □ Importante □ Desejável

[RF003] Visualizar Rotas	
Descrição:	O motorista visualiza a rota selecionada.
Prioridade:	□ Essencial □Importante ■ Desejável

[RF004] Acessar Ouvidoria	
Descrição:	O motorista pode registrar uma reclamação e ver as que já fez anteriormente, tanto as respondidas quanto as não respondidas.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF005] Visualizar Histórico de Ouvidorias

Descrição:	O motorista visualiza suas ouvidorias enviadas, separadas entre pendentes e resolvidas.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF006] Acessar Perfil	
Descrição:	O motorista acessa o perfil e visualiza suas informações.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF007] Editar Senha	
Descrição:	O motorista pode editar sua senha.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF008] Autenticar Administrador	
Descrição:	O administrador realiza login com email e senha. O sistema verifica se ele é de nível 3 (Administrador).
Prioridade:	□ Essencial ■ Importante □ Desejável

[RF009] Gerenciar Motoristas	
Descrição:	O administrador pode listar, criar, editar e deletar Motoristas.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF010] Gerenciar Administradores	
Descrição:	O administrador pode listar, criar, editar e deletar administradores.
Prioridade:	□ Essencial □ Importante ■ Desejável

[RF011] Gerenciar Ouvidoria	
Descrição:	O administrador pode listar e responder às reclamações.
Prioridade:	□ Essencial □ Importante ■ Desejável

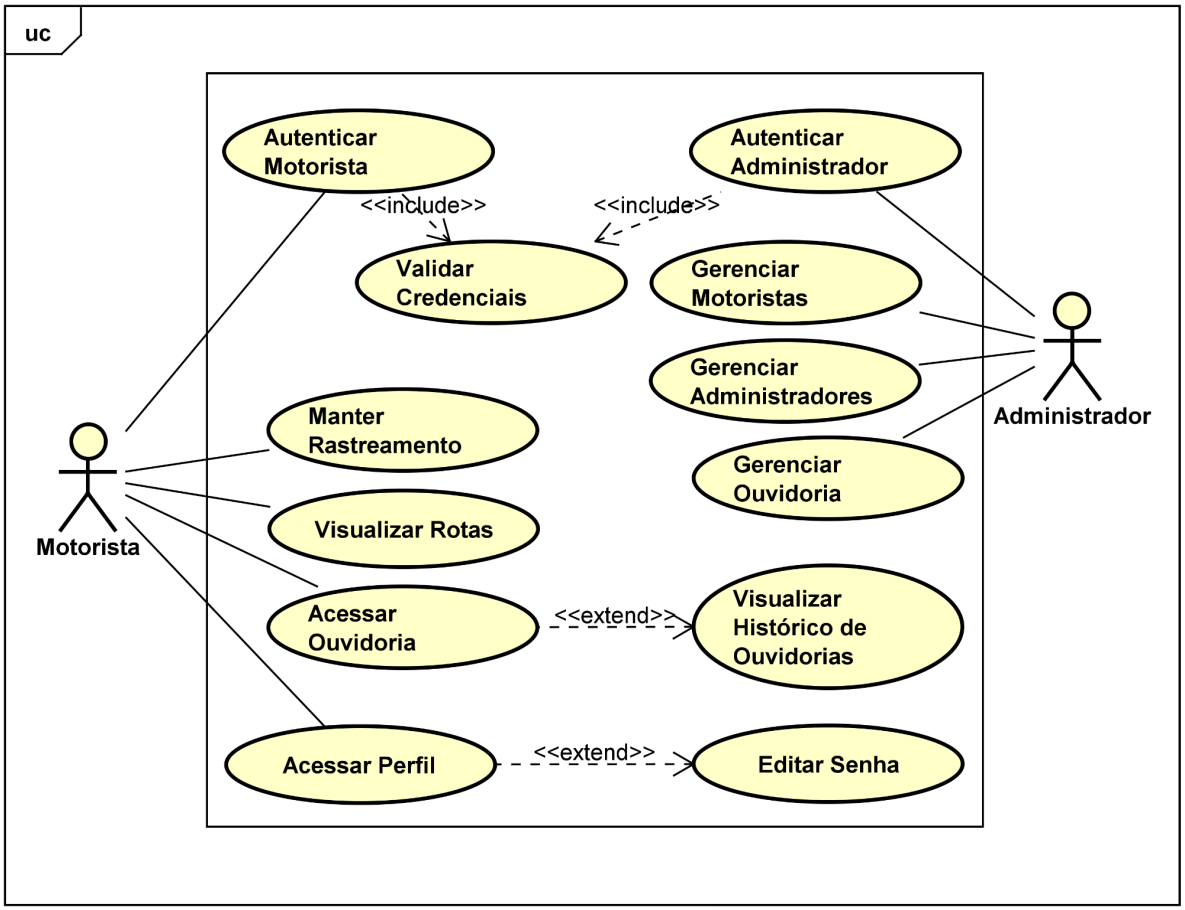
[RF012] Validar Credenciais

Descrição:	O sistema verifica se as credenciais informadas são de acesso nível 2 (Motorista) ou 3 (Administrador).
Prioridade:	□ Essencial ■ Importante □ Desejável

3.2 DIAGRAMA DE CASOS DE USO

A Figura 1 representa o Diagrama de Casos de Uso do sistema, demonstrando as ações que cada ator pode realizar. O motorista pode autenticar-se, manter rastreamento, visualizar rotas, acessar ouvidoria (com extensão para visualizar histórico) e acessar perfil (com extensão para editar senha). O administrador pode autenticar-se, gerenciar motoristas, gerenciar administradores e gerenciar ouvidoria. Ambos os casos de autenticação incluem validação de credenciais.

Figura 1 – Diagrama de Casos de Uso



Fonte: Autoria Própria(2025).

3.3 CASOS DE USO

A seguir os casos de uso:

Quadro 2 - Autenticar Motorista

[UC01] Autenticar Motorista	
Atores:	Motorista
Pré-Condições:	O motorista possui cadastro no sistema com acesso nível 2.
Pós-Condições:	O motorista está autenticado e acessa a tela inicial do aplicativo.
FLUXO PRINCIPAL	
1. O sistema exibe a tela de login. 2. O motorista informa email e senha. 3. O motorista solicita o login. 4. O sistema executa o caso de uso [UC12] Validar Credenciais. 5. O sistema verifica que o email está verificado. 6. O sistema redireciona o motorista para a tela inicial.	
FLUXOS ALTERNATIVOS	
A2. O motorista esqueceu a senha. A2.1. O motorista clica em "Esqueci a senha". A2.2. O sistema exibe o formulário de recuperação. A2.3. O motorista informa seu email. A2.4. O sistema envia email de recuperação de senha. A2.5. O caso de uso retorna ao passo 1.	
FLUXOS DE EXCEÇÃO	
E4. Credenciais inválidas. E4.1. O sistema exibe mensagem de erro. E4.2. O caso de uso retorna ao passo 2. E5. Email não verificado. E5.1. O sistema envia um novo email de verificação. E5.2. O sistema exibe mensagem informando que o email precisa ser verificado. E5.3. O caso de uso termina.	

Fonte: Autoria Própria(2025).

Quadro 3 - Manter Rastreamento

[UC02] Manter Rastreamento	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado e há rotas cadastradas no sistema.
Pós-Condições:	O rastreamento está ativo ou foi encerrado, atualizando o Realtime Database.
FLUXO PRINCIPAL	
Iniciar Rastreamento: 1. O motorista seleciona uma rota através do caso de uso [UC03] Visualizar Rotas. 2. O motorista define o status do ônibus (Em operação, Fora da rota, Parado, Manutenção, Fora de serviço). 3. O motorista clica no botão de iniciar rastreamento. 4. O sistema verifica as permissões de localização. 5. O sistema inicia o LocationTrackingService. 6. O sistema exibe notificação persistente informando que o rastreamento está ativo. 7. O sistema atualiza a localização, velocidade e status no Realtime Database a cada 2 segundos. **Parar Rastreamento:** 8. O motorista clica no botão de parar rastreamento. 9. O sistema solicita confirmação. 10. O motorista confirma. 11. O sistema encerra o LocationTrackingService. 12. O sistema remove os dados do ônibus do Realtime Database. 13. O sistema remove a notificação persistente.	
FLUXOS ALTERNATIVOS	
1. Nenhuma rota selecionada. 1.1. O sistema exibe mensagem solicitando seleção de rota. 1.2. O caso de uso retorna ao passo 1.	

10. O motorista cancela a operação. 10.1. O sistema mantém o rastreamento ativo. 10.2. O caso de uso termina.
FLUXOS DE EXCEÇÃO
E4. Permissões de localização não concedidas. E4.1. O sistema exibe diálogo explicando a necessidade das permissões. E4.2. O sistema solicita as permissões necessárias. E4.3. Se as permissões forem negadas, o caso de uso termina. E4.4. Se as permissões forem concedidas, o caso de uso retorna ao passo 5. E7. Erro ao atualizar Realtime Database. E7.1. O sistema registra o erro no log. E7.2. O sistema tenta novamente na próxima atualização.

Fonte: Autoria Própria(2025).

Quadro 4 - Visualizar Rotas

[UC03] Visualizar Rotas	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado e há rotas cadastradas no sistema.
Pós-Condições:	A rota selecionada é exibida no mapa com seus pontos e paradas.
FLUXO PRINCIPAL	
1. O motorista acessa a tela inicial. 2. O sistema carrega todas as rotas do Firestore. 3. O motorista clica no botão de seleção de rota. 4. O sistema exibe diálogo com a lista de rotas (nome, código e cor). 5. O motorista seleciona uma rota. 6. O sistema desenha a polyline da rota no mapa com a cor correspondente. 7. O sistema exibe os marcadores das paradas. 8. O sistema exibe card superior com o nome da rota selecionada.	
FLUXOS ALTERNATIVOS	
A2. Nenhuma rota cadastrada.	

A2.1. O sistema exibe a mensagem "Nenhuma rota disponível". A2.2. O caso de uso termina. A5. O motorista cancela a seleção. A5.1. O sistema fecha o diálogo. A5.2. A rota anterior permanece selecionada (se houver). A5.3. O caso de uso termina.
FLUXOS DE EXCEÇÃO
E2. Erro ao carregar rotas do Firestore. E2.1. O sistema exibe mensagem de erro. E2.2. O sistema oferece a opção de tentar novamente.

Fonte: Autoria Própria(2025).

Quadro 5 - Acessar Ouvidoria

[UC04] Acessar Ouvidoria	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado.
Pós-Condições:	Uma nova ouvidoria foi enviada ou o motorista visualiza ouvidorias existentes.
FLUXO PRINCIPAL	
1. O motorista acessa o menu e seleciona "Ouvidoria". 2. O sistema exibe a tela de ouvidoria com três abas: Pendentes, Resolvidas, Nova. 3. O motorista seleciona a aba "Nova". 4. O motorista seleciona a categoria do problema. 5. O motorista preenche o título do problema. 6. O motorista preenche a descrição detalhada. 7. O motorista pode opcionalmente informar localização e telefone. 8. O motorista clica em "Enviar Relatório". 9. O sistema valida os campos obrigatórios. 10. O sistema gera um ID único (UUID). 11. O sistema salva a ouvidoria no Firestore com status "Pendente". 12. O sistema exibe mensagem de sucesso com o número do protocolo. 13. O sistema limpa os campos do formulário.	

FLUXOS ALTERNATIVOS
A3. O motorista seleciona a aba "Pendentes". A3.1. O sistema executa [UC05] Visualizar Histórico de Ouvidorias para ouvidorias pendentes. A3. O motorista seleciona a aba "Resolvidas". A3.1. O sistema executa [UC05] Visualizar Histórico de Ouvidorias para ouvidorias resolvidas.
FLUXOS DE EXCEÇÃO
E9. Campos obrigatórios não preenchidos. E9.1. O sistema exibe mensagem indicando os campos faltantes. E9.2. O caso de uso retorna ao passo 4. E11. Erro ao salvar no Firestore. E11.1. O sistema exibe mensagem de erro. E11.2. O sistema mantém os dados preenchidos. E11.3. O motorista pode tentar enviar novamente.

Fonte: Autoria Própria(2025).

Quadro 6 - Visualizar Histórico de Ouvidorias

[UC05] Visualizar Histórico de Ouvidorias	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado e acessou a ouvidoria.
Pós-Condições:	O motorista visualiza o histórico de ouvidorias (pendentes ou resolvidas).
FLUXO PRINCIPAL	
1. O sistema carrega as ouvidorias do motorista do Firestore. 2. O sistema filtra as ouvidorias conforme a aba selecionada (Pendentes ou Resolvidas). 3. O sistema ordena as ouvidorias por data (mais recentes primeiro). 4. O sistema exibe lista de cards com: categoria, título, descrição resumida e data. 5. O motorista pode clicar em um card para ver detalhes. 6. O sistema exibe diálogo com informações completas da ouvidoria.	

7. Se a ouvidoria estiver resolvida, o sistema também exibe a resposta do administrador.
FLUXOS ALTERNATIVOS
A3. O motorista seleciona a aba "Pendentes". A3.1. O sistema executa [UC05] Visualizar Histórico de Ouvidorias para ouvidorias pendentes. A3. O motorista seleciona a aba "Resolvidas". A3.1. O sistema executa [UC05] Visualizar Histórico de Ouvidorias para ouvidorias resolvidas.
FLUXOS DE EXCEÇÃO
E1. Erro ao carregar ouvidorias. E1.1. O sistema exibe mensagem de erro. E1.2. O sistema oferece a opção de tentar novamente.

Fonte: Autoria Própria(2025).

Quadro 7 - Acessar Perfil

[UC06] Acessar Perfil	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado.
Pós-Condições:	As informações do perfil são exibidas.
FLUXO PRINCIPAL	
1. O motorista acessa o menu e seleciona "Perfil". 2. O sistema carrega os dados do usuário do Firebase Authentication. 3. O sistema exibe card com as informações: nome e email. 4. O sistema exibe o botão "Alterar Senha". 5. O motorista pode clicar em "Alterar Senha" para executar [UC07] Editar Senha.	
FLUXOS ALTERNATIVOS	
A5. O motorista volta para a tela anterior sem alterar senha. A5.1. O caso de uso termina.	
FLUXOS DE EXCEÇÃO	
E2. Erro ao carregar dados do usuário.	

E2.1. O sistema exibe mensagem de erro. E2.2. O sistema exibe dados parciais disponíveis.

Fonte: Autoria Própria(2025).

Quadro 8 - Editar Senha

[UC07] Editar Senha	
Atores:	Motorista
Pré-Condições:	O motorista está autenticado e acessou o perfil.
Pós-Condições:	A senha do motorista foi alterada
FLUXO PRINCIPAL	
1. O motorista clica em "Alterar Senha". 2. O sistema exibe diálogo com campos: Senha atual, Nova senha, Confirmar nova senha. 3. O motorista preenche os três campos. 4. O motorista clica em "Alterar". 5. O sistema valida se a nova senha tem pelo menos 6 caracteres. 6. O sistema valida se a nova senha e confirmação são iguais. 7. O sistema reautentica o usuário com a senha atual. 8. O sistema atualiza a senha no Firebase Authentication. 9. O sistema exibe mensagem de sucesso. 10. O sistema fecha o diálogo. 11. O sistema limpa os campos.	
FLUXOS ALTERNATIVOS	
A4. O motorista clica em "Cancelar". A4.1. O sistema fecha o diálogo. A4.2. O caso de uso termina.	
FLUXOS DE EXCEÇÃO	
E5. Nova senha com menos de 6 caracteres. E5.1. O sistema exibe a mensagem "A senha deve ter pelo menos 6 caracteres". E5.2. O caso de uso retorna ao passo 3. E6. Nova senha e confirmação não coincidem. E6.1. O sistema exibe a mensagem "As senhas não coincidem".	

E6.2. O caso de uso retorna ao passo 3. E7. Senha atual incorreta. E7.1. O sistema exibe a mensagem "Senha atual incorreta". E7.2. O caso de uso retorna ao passo 3. E8. Erro ao atualizar senha. E8.1. O sistema exibe mensagem com o erro. E8.2. O caso de uso retorna ao passo 3.

Fonte: Autoria Própria(2025).

Quadro 9 – Autenticar Administrador

[UC08] Autenticar Administrador	
Atores:	Administrador
Pré-Condições:	O administrador possui cadastro no sistema com acesso nível 3.
Pós-Condições:	O administrador está autenticado e acessa o painel administrativo.
FLUXO PRINCIPAL	
1. O sistema exibe a tela de login. 2. O administrador informa email e senha. 3. O administrador solicita o login. 4. O sistema executa o caso de uso [UC12] Validar Credenciais. 5. O sistema verifica que o email está verificado. 6. O sistema redireciona o administrador para o painel administrativo.	
FLUXOS ALTERNATIVOS	
A2. O administrador esqueceu a senha. A2.1. O administrador clica em "Esqueci a senha". A2.2. O sistema exibe o formulário de recuperação. A2.3. O administrador informa seu email. A2.4. O sistema envia email de recuperação de senha. A2.5. O caso de uso retorna ao passo 1.	
FLUXOS DE EXCEÇÃO	

E4. Credenciais inválidas.

E4.1. O sistema exibe mensagem de erro.

E4.2. O caso de uso retorna ao passo 2.

E5. Email não verificado.

E5.1. O sistema exibe mensagem informando que o email precisa ser verificado.

E5.2. O caso de uso termina.

Fonte: Autoria Própria(2025).

Quadro 10 - Gerenciar Motoristas

[UC09] Gerenciar Motoristas	
Atores:	Administrador
Pré-Condições:	O administrador está autenticado.
Pós-Condições:	Um motorista foi criado, editado, ativado/desativado ou excluído.
FLUXO PRINCIPAL	

Listar Motoristas

1. O administrador acessa o painel administrativo.
2. O sistema exibe a aba "Motoristas" selecionada.
3. O sistema carrega motoristas (acesso = 2) do Firestore.
4. O sistema exibe lista de cards com: nome, email, UID e status (ativo/inativo).

Criar Motorista:

C1. O administrador clica no botão "+".

C2. O sistema exibe formulário: nome, email, senha, confirmar senha.

C3. O administrador preenche os campos.

C4. O administrador clica em "Criar Motorista".

C5. O sistema valida os campos.

C6. O sistema cria conta no Firebase Authentication com nível acesso = 2.

C7. O sistema salva dados no Firestore.

C8. O sistema envia email de verificação.

C9. O sistema exibe mensagem de sucesso.

C10. O caso de uso retorna ao passo 3.

Editar Motorista:

E1. O administrador clica em "Editar" em um card de motorista.

E2. O sistema exibe diálogo com campos: nome e email preenchidos.

E3. O administrador altera os dados.

E4. O administrador clica em "Salvar".

E5. O sistema valida os campos.

E6. O sistema atualiza os dados no Firestore.

E7. O sistema exibe mensagem de sucesso.

E8. O caso de uso retorna ao passo 3.

Ativar/Desativar Motorista:

D1. O administrador clica em "Ativar" ou "Desativar" em um card.

D2. O sistema solicita confirmação.

D3. O administrador confirma.

D4. O sistema atualiza o campo "ativo" no Firestore.

D5. O sistema exibe mensagem de sucesso.

D6. O caso de uso retorna ao passo 3.

Excluir Motorista:

X1. O administrador clica em "Excluir" em um card de motorista.

X2. O sistema solicita confirmação com aviso que a ação é irreversível.

X3. O administrador confirma.

X4. O sistema remove o documento do Firestore.

X5. O sistema exibe mensagem de sucesso.

X6. O caso de uso retorna ao passo 3.

FLUXOS ALTERNATIVOS

A3. Nenhum motorista cadastrado.

A3.1. O sistema exibe a mensagem "Nenhum motorista cadastrado" com ícone.

A3.2. O sistema oferece botão "+" para adicionar.

FLUXOS DE EXCEÇÃO

E3. Erro ao carregar motoristas.

E3.1. O sistema exibe mensagem de erro.

E3.2. O sistema oferece opção de atualizar.

C5, E5. Campos inválidos ou vazios.

C5.1, E5.1. O sistema exibe mensagem indicando o problema. C5.2, E5.2. O caso de uso retorna ao passo anterior. C6. Email já cadastrado. C6.1. O sistema exibe a mensagem "Este e-mail já está cadastrado". C6.2. O caso de uso retorna ao passo C3.

Fonte: Autoria Própria(2025).

Quadro 11 - Autenticar Administrador

[UC10] Gerenciar Administradores	
Atores:	Administrador
Pré-Condições:	O administrador está autenticado.
Pós-Condições:	Um administrador foi criado, editado, ativado/desativado ou excluído.
FLUXO PRINCIPAL	
Listar Administradores: 1. O administrador acessa o painel administrativo. 2. O administrador seleciona a aba "Administradores". 3. O sistema carrega administradores (acesso = 3) do Firestore. 4. O sistema exibe lista de cards com: nome, email, UID e status (ativo/inativo). **Criar Administrador:** C1. O administrador clica no botão "+". C2. O sistema exibe formulário: nome, email, senha, confirmar senha. C3. O administrador preenche os campos. C4. O administrador clica em "Criar Administrador". C5. O sistema valida os campos. C6. O sistema cria conta no Firebase Authentication com nível acesso = 3. C7. O sistema salva dados no Firestore. C8. O sistema envia email de verificação. C9. O sistema exibe mensagem de sucesso. C10. O caso de uso retorna ao passo 3. **Editar Administrador:**	

E1. O administrador clica em "Editar" em um card. E2. O sistema exibe diálogo com campos: nome e email preenchidos. E3. O administrador altera os dados. E4. O administrador clica em "Salvar". E5. O sistema valida os campos. E6. O sistema atualiza os dados no Firestore. E7. O sistema exibe mensagem de sucesso. E8. O caso de uso retorna ao passo 3. **Ativar/Desativar Administrador:** D1. O administrador clica em "Ativar" ou "Desativar" em um card. D2. O sistema solicita confirmação. D3. O administrador confirma. D4. O sistema atualiza o campo "ativo" no Firestore. D5. O sistema exibe mensagem de sucesso. D6. O caso de uso retorna ao passo 3. **Excluir Administrador:** X1. O administrador clica em "Excluir" em um card. X2. O sistema solicita confirmação com aviso que a ação é irreversível. X3. O administrador confirma. X4. O sistema remove o documento do Firestore. X5. O sistema exibe mensagem de sucesso. X6. O caso de uso retorna ao passo 3.
FLUXOS ALTERNATIVOS
A3. Nenhum administrador cadastrado. A3.1. O sistema exibe a mensagem "Nenhum administrador cadastrado" com ícone. A3.2. O sistema oferece botão "+" para adicionar.
FLUXOS DE EXCEÇÃO
E3. Erro ao carregar administradores. E3.1. O sistema exibe mensagem de erro. E3.2. O sistema oferece opção de atualizar. C5, E5. Campos inválidos ou vazios.

C5.1, E5.1. O sistema exibe mensagem indicando o problema. C5.2, E5.2. O caso de uso retorna ao passo anterior. C6. Email já cadastrado. C6.1. O sistema exibe a mensagem "Este e-mail já está cadastrado". C6.2. O caso de uso retorna ao passo C3.

Fonte: Autoria Própria(2025).

Quadro 12 - Gerenciar Ouvidoria

[UC11] Gerenciar Ouvidoria	
Atores:	Administrador
Pré-Condições:	O administrador está autenticado e há ouvidorias cadastradas.
Pós-Condições:	Uma ouvidoria foi visualizada ou respondida.
FLUXO PRINCIPAL	
Listar Ouvidorias: 1. O administrador clica no ícone "Ouvidoria" no menu superior. 2. O sistema carrega todas as ouvidorias do Firestore ordenadas por data. 3. O sistema exibe duas abas: "Pendentes" e "Resolvidas". 4. O sistema filtra as ouvidorias conforme a aba selecionada. 5. O sistema exibe lista de cards com: categoria, título, nome do usuário, descrição resumida e data. 6. O sistema exibe badges com a quantidade de ouvidorias em cada aba. **Visualizar Detalhes:** V1. O administrador clica em um card de ouvidoria. V2. O sistema exibe diálogo com informações completas: categoria, título, descrição, localização, telefone, data, protocolo. V3. Se a ouvidoria estiver resolvida, exibe também: resposta, nome do administrador que respondeu e data da resposta. **Responder Ouvidoria:** R1. O administrador clica em uma ouvidoria pendente. R2. O sistema exibe diálogo com campo de texto "Sua Resposta". R3. O administrador escreve a resposta. R4. O administrador clica em "Marcar como Resolvida".	

R5. O sistema valida se a resposta não está vazia. R6. O sistema atualiza a ouvidoria no Firestore: resolvido=true, resposta, nomeAdministrador, dataResposta, status="Resolvido". R7. O sistema exibe mensagem de sucesso. R8. O sistema fecha o diálogo. R9. O caso de uso retorna ao passo 2.
FLUXOS ALTERNATIVOS
A2. Nenhuma ouvidoria cadastrada. A2.1. O sistema exibe a mensagem "Nenhuma ouvidoria encontrada" com ícone. A2.2. O sistema oferece botão para atualizar. R4. O administrador clica em "Fechar" sem responder. R4.1. O sistema fecha o diálogo sem salvar. R4.2. A ouvidoria permanece pendente.
FLUXOS DE EXCEÇÃO
E2. Erro ao carregar ouvidorias. E2.1. O sistema exibe mensagem de erro. E2.2. O sistema oferece opção de atualizar. R5. Campo de resposta vazio. R5.1. O sistema exibe a mensagem "Por favor, escreva uma resposta". R5.2. O caso de uso retorna ao passo R3. R6. Erro ao salvar no Firestore. R6.1. O sistema exibe mensagem de erro. R6.2. O sistema mantém a resposta digitada. R6.3. O administrador pode tentar novamente.

Fonte: Autoria Própria(2025).

Quadro 13 - Validar Credenciais

[UC12] Validar Credenciais	
Atores:	Administrador

Pré-Condições:	O administrador possui cadastro no sistema com acesso nível 3.
Pós-Condições:	O administrador está autenticado e acessa o painel administrativo.

FLUXO PRINCIPAL

1. O sistema recebe email e senha.
2. O sistema valida se os campos não estão vazios.
3. O sistema tenta autenticar com Firebase Authentication.
4. O sistema obtém o UID do usuário autenticado.
5. O sistema busca o documento do usuário no Firestore usando o UID.
6. O sistema obtém o campo "acesso" do documento.
7. O sistema verifica se o acesso é 2 (Motorista) ou 3 (Administrador).
8. O sistema retorna o nível de acesso para o caso de uso chamador.

FLUXOS ALTERNATIVOS

Não há.

FLUXOS DE EXCEÇÃO

E2. Campos vazios.

E2.1. O sistema retorna erro "Preencha todos os campos".

E2.2. O caso de uso termina.

E3. Credenciais inválidas.

E3.1. O sistema retorna erro "Email ou senha incorretos".

E3.2. O caso de uso termina.

E5. Documento do usuário não encontrado no Firestore.

E5.1. O sistema retorna erro "Usuário não encontrado no sistema".

E5.2. O caso de uso termina.

E7. Nível de acesso diferente de 2 ou 3.

E7.1. O sistema retorna erro "Acesso negado".

E7.2. O sistema realiza logout do Firebase Authentication.

E7.3. O caso de uso termina.

Fonte: Autoria Própria(2025).

3.4 TECNOLOGIAS UTILIZADAS

Esta seção descreve as principais tecnologias empregadas no desenvolvimento do aplicativo para motoristas, explicando as razões técnicas de cada escolha e como contribuíram para resolver os desafios específicos do projeto.

3.4.1 Android e Kotlin

O desenvolvimento foi realizado para a plataforma Android utilizando Kotlin como linguagem principal. Android foi escolhido por representar aproximadamente 70% do mercado brasileiro de smartphones, garantindo maior alcance entre os motoristas de transporte público. Kotlin, por sua vez, tornou-se a linguagem oficial do Google para Android em 2017 e oferece vantagens significativas sobre o Java, como null safety (prevenção de erros de ponteiro nulo), sintaxe mais enxuta e suporte nativo a corrotinas para programação assíncrona. No contexto deste projeto, Kotlin permitiu reduzir significativamente o código necessário para manipular callbacks assíncronos do Firebase e gerenciar o ciclo de vida dos componentes Android.

3.4.2 Firebase Authentication

Responsável pelo gerenciamento completo de autenticação e autorização dos usuários. O serviço oferece recursos prontos como criação de contas, login com email/senha, verificação de email, recuperação de senha e tokens de sessão seguros. No sistema desenvolvido, cada usuário recebe um UID[1] único no momento do cadastro, que serve como chave primária para vincular dados em outras coleções. A principal vantagem foi não precisar implementar lógicas complexas de hash de senhas, gestão de tokens JWT ou mecanismos de segurança, pois o Firebase Authentication já implementa as melhores práticas da indústria.

[1]**UID (User Identifier):** Identificador único atribuído a cada usuário no momento do cadastro. É uma string alfanumérica gerada automaticamente que garante que cada conta seja única no sistema, funcionando como chave primária para relacionar dados.

3.4.3 Cloude Firestore

Selecionado como banco de dados principal por sua natureza NoSQL[2] orientada a documentos, que se adequa bem à estrutura hierárquica dos dados do projeto (rotas contendo arrays de pontos e paradas). O Firestore organiza dados em coleções (equivalente a tabelas) e documentos (equivalente a registros), permitindo estruturas flexíveis sem necessidade de migrações de schema. No aplicativo, armazena dados permanentes como usuários, rotas configuradas e histórico de ouvidorias. Uma característica importante é a capacidade de realizar consultas com filtros (whereEqualTo) e ordenação (orderBy), utilizada para buscar rotas específicas e listar ouvidorias por status.

3.4.4 Realtime Database

Diferente do Firestore, o Realtime Database foi escolhido especificamente para o rastreamento em tempo real por sua capacidade de sincronização instantânea e estrutura otimizada para atualizações frequentes. Enquanto o Firestore cobra por operação de leitura/escrita, o Realtime Database cobra por bandwidth e armazenamento, tornando-o mais econômico para dados que mudam constantemente (como posição GPS atualizada a cada 2 segundos). No sistema, cada ônibus em rastreamento possui um nó temporário identificado pelo UID do motorista, contendo latitude, longitude, velocidade, status e timestamp. Esses dados são consumidos em tempo real pelo aplicativo dos passageiros (desenvolvido por Eduardo Antunes), que observa mudanças no banco via listeners.

3.4.5 Jetpack Compose

Framework moderno do Google para construção de interfaces declarativas no Android, lançado em 2021 como substituto do sistema tradicional baseado em XML. Em Jetpack Compose, a interface é construída através de funções composable que descrevem como a UI deve aparecer dado um determinado estado, eliminando a

---

[2]**NoSQL:** Tipo de banco de dados que não utiliza o modelo relacional tradicional de tabelas com linhas e colunas. Organiza dados em formatos flexíveis como documentos (JSON), grafos ou chave-valor, permitindo estruturas mais dinâmicas sem necessidade de schemas rígidos.

necessidade de manipular views manualmente. Quando o estado muda (por exemplo, isTracking passa de false para true), o Compose automaticamente reconstrói apenas as partes da interface afetadas. Essa abordagem reativa simplificou significativamente o desenvolvimento das telas do motorista, especialmente a tela principal que precisa reagir a múltiplas mudanças de estado (rota selecionada, status do ônibus, permissões concedidas, rastreamento ativo). O Compose também oferece pré visualização em tempo real durante o desenvolvimento, acelerando o ciclo de design-implementação.

3.4.6 Osmdroid (Openstreetmap)

Biblioteca Android que permite integrar mapas do OpenStreetMap, projeto colaborativo de dados geográficos abertos. Foi escolhida em detrimento do Google Maps principalmente por dois fatores: ausência de custos (Google Maps cobra após 28.000 carregamentos mensais) e liberdade de personalização. O OSMdroid oferece classes como MapView (componente visual do mapa), Marker (marcadores no mapa) e Polyline[3] (linhas conectando pontos), que foram utilizadas para desenhar as rotas das linhas de ônibus. Uma limitação identificada foi o desempenho ligeiramente inferior ao Google Maps em dispositivos de baixo custo, porém aceitável para o caso de uso do projeto. A biblioteca requer configuração de user-agent e diretórios de cache, realizados na classe Application (MyApp.kt).

3.4.7 Google Play Services Location

Biblioteca oficial do Google para acesso aos serviços de localização do Android. Oferece abstração de alto nível sobre os provedores de localização do sistema (GPS, Wi-Fi, torres de celular), permitindo solicitar updates periódicos através do FusedLocationProviderClient. No projeto, foi configurado um LocationRequest com prioridade PRIORITY_HIGH_ACCURACY (máxima precisão via GPS), intervalo de 2000ms entre atualizações e distância mínima de 1 metro para disparar novos eventos. O LocationCallback recebe objetos Location contendo

---

[3] ***Polyline*:** Linha formada pela conexão sequencial de múltiplos pontos geográficos no mapa. No contexto do projeto, representa visualmente o trajeto completo de uma linha de ônibus, conectando todos os pontos da rota.

latitude, longitude, velocidade (em m/s), altitude e precisão estimada. Essa biblioteca foi essencial para implementar o rastreamento em tempo real, pois lida automaticamente com otimizações de bateria e escolha do melhor provedor de localização disponível.

3.4.8 Android Foreground Service

Serviços Android normais podem ser finalizados pelo sistema operacional para economizar recursos quando o aplicativo está em segundo plano. Para tarefas críticas que precisam continuar executando (como o rastreamento GPS do motorista), o Android oferece Foreground Services, que mantêm uma notificação persistente visível ao usuário e recebem prioridade de processamento. No projeto, a classe LocationTrackingService estende Service e utiliza startForeground() para garantir que o rastreamento continue mesmo com o aplicativo minimizado. A notificação exibida mostra o nome da rota sendo percorrida e o status atual do ônibus, permitindo que o motorista tenha visibilidade da operação. A partir do Android 8.0, é obrigatório declarar o tipo de foreground service no Manifest (android:foregroundServiceType="location").

3.4.9 Androidx Lifecycle

Conjunto de componentes que ajudam a gerenciar o ciclo de vida de Activities e Fragments no Android, onde componentes passam por estados como Created, Started, Resumed, Paused, Stopped e Destroyed. O Lifecycle-aware components observam automaticamente essas mudanças e reagem adequadamente. No projeto, *ViewModel*[4] e LiveData (parte da arquitetura Lifecycle) foram utilizados para separar lógica de negócio (ViewModels) da interface (Composables). ViewModels sobrevivem a mudanças de configuração como rotação de tela, preservando o estado. O LaunchedEffect do Compose foi usado para iniciar corrotinas que observam mudanças no ViewModel e atualizam a UI, garantindo que operações sejam canceladas quando a tela é destruída, prevenindo travamentos de memória.

---

[4] ***ViewModel***: Componente da arquitetura Android responsável por armazenar e gerenciar dados relacionados à interface. Sobrevive a mudanças de configuração (como rotação de tela) e mantém a separação entre lógica de negócio e interface visual.

3.4.10 Kotlin Coroutines

Sistema de concorrência nativo do Kotlin que permite escrever código assíncrono de forma sequencial, sem callbacks aninhados. Corrotinas são "*threads*[5] leves" que podem ser suspensas e retomadas sem bloquear a thread principal. No contexto do projeto, todas as operações de rede (Firebase) e I/O (Geocoder) foram executadas em corrotinas para não travar a interface. Foram utilizados escopos diferentes: viewModelScope (vinculado ao ciclo de vida do ViewModel) e CoroutineScope customizado com SupervisorJob no Service (permite que falhas em uma coroutina não cancelem outras). O Dispatchers.IO foi usado para operações de I/O, enquanto Dispatchers.Main para atualizações de UI.

3.4.11 Permissões de localização

O Android possui um sistema granular de permissões que o usuário deve conceder explicitamente. Para este projeto, foram necessárias três permissões: ACCESS_FINE_LOCATION (localização precisa via GPS), ACCESS_COARSE_LOCATION (localização aproximada via rede) e ACCESS_BACKGROUND_LOCATION (localização em segundo plano, obrigatória a partir do Android 10). O código implementa verificação de permissões com ContextCompat.checkSelfPermission(), solicitação via ActivityResultContracts.RequestMultiplePermissions() e então o sistema direciona para as configurações do aplicativo. O HomeViewModel mantém estados hasLocationPermission e hasBackgroundPermission para controlar quando o rastreamento pode ser iniciado.

3.5 FIGURAS DO SISTEMA

[5] ***Thread***: Sequência independente de execução dentro de um programa. A thread principal (main thread) é responsável pela interface do usuário, enquanto threads secundárias executam operações pesadas em segundo plano para não travar a aplicação.

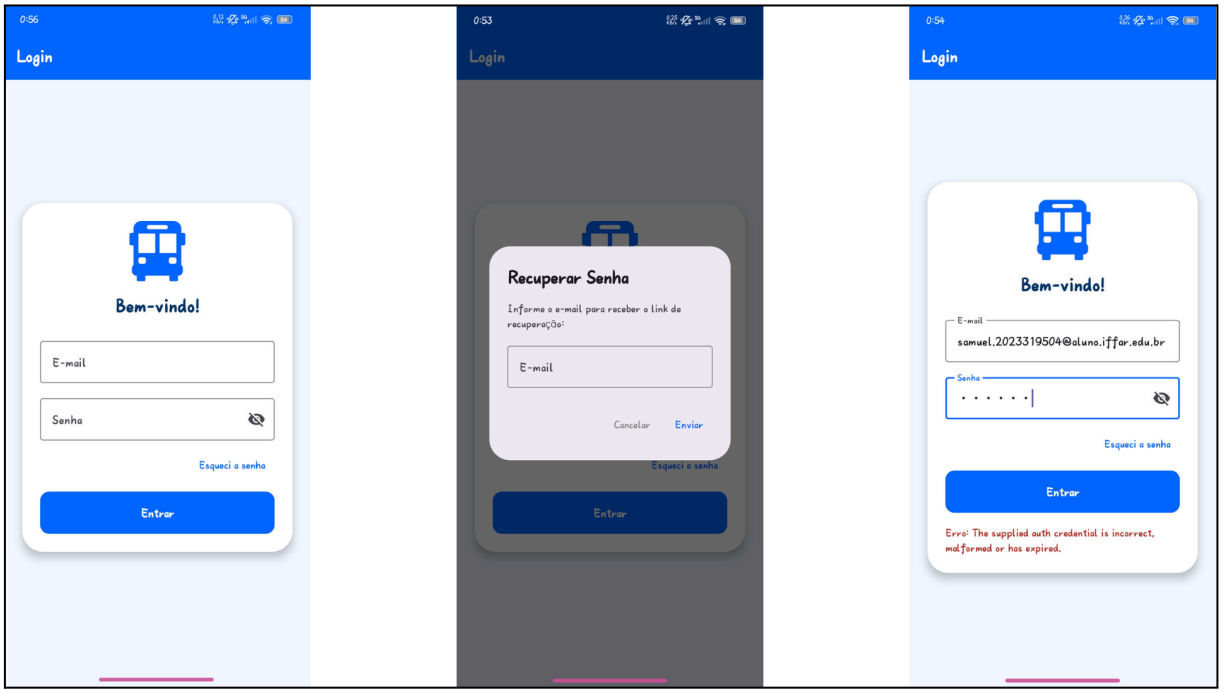
A seguir as figuras das Figuras da interface, Figuras do código e Figuras do banco de dados:

3.5.1 Figuras da interface

A seguir as figuras referentes às telas do sistema:

A Figura 2 apresenta a tela de autenticação do sistema. Um card branco centralizado exibe o ícone de ônibus em azul no topo, seguido pelo texto "Bem-vindo!" e campos para e-mail e senha. O campo de senha possui ícone de visibilidade para alternar entre mostrar e ocultar caracteres. Abaixo dos campos, o link "Esqueci a senha" permite recuperação de acesso através de diálogo modal. O botão "Entrar" em azul (#0066FF) ocupa toda a largura do card. Mensagens de erro ou sucesso aparecem abaixo do botão em cores apropriadas (vermelho para erro, azul para sucesso). O fundo da tela utiliza cinza claro (#F0F7FF) para melhor contraste.

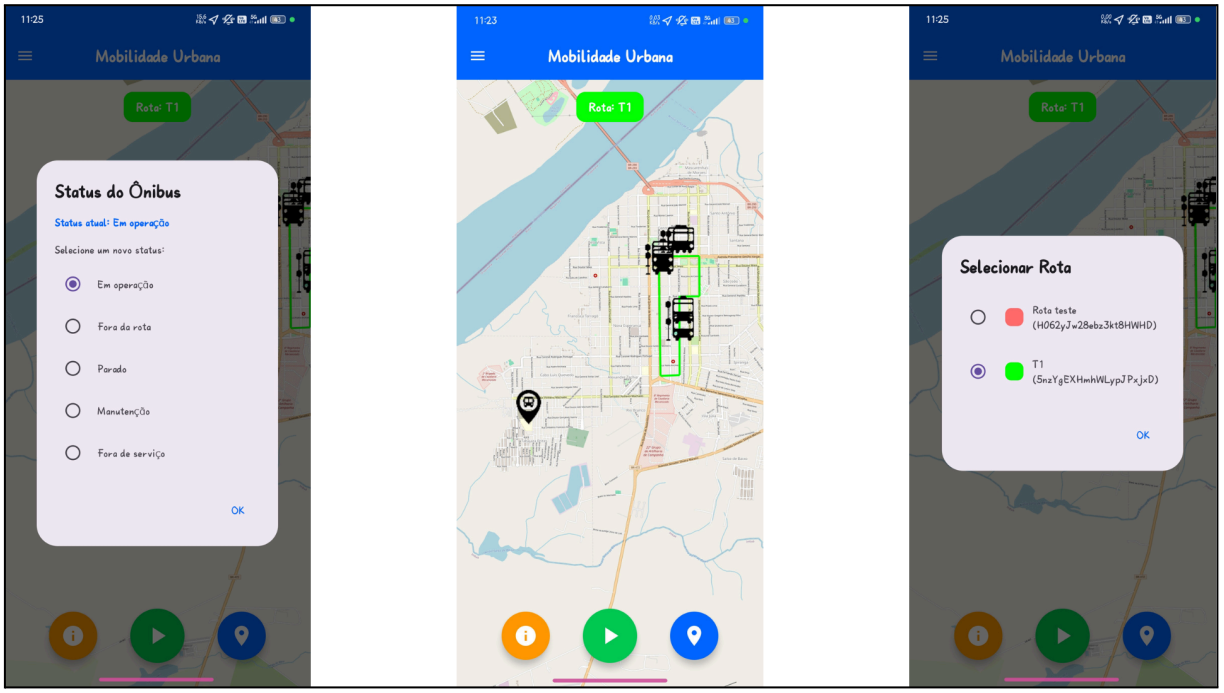
Figura 2 – Tela de Login



Fonte: Autoria Própria(2025).

A Figura 3 mostra a interface principal do motorista com mapa OpenStreetMap em tela cheia. A localização atual do motorista é indicada por marcador personalizado, enquanto a rota selecionada aparece como polyline colorida. Marcadores de paradas de ônibus são exibidos ao longo do trajeto. No topo, um card flutuante mostra o nome da rota e status do rastreamento. Três botões de ação flutuante (FABs) circulares ficam na parte inferior: botão laranja para alterar status do ônibus (com diálogo de radiobuttons), botão verde/vermelho central maior para iniciar/parar rastreamento, e botão azul para selecionar rota (exibindo lista com cores visuais). Menu lateral (drawer) oferece acesso a Perfil e Ouvidoria.

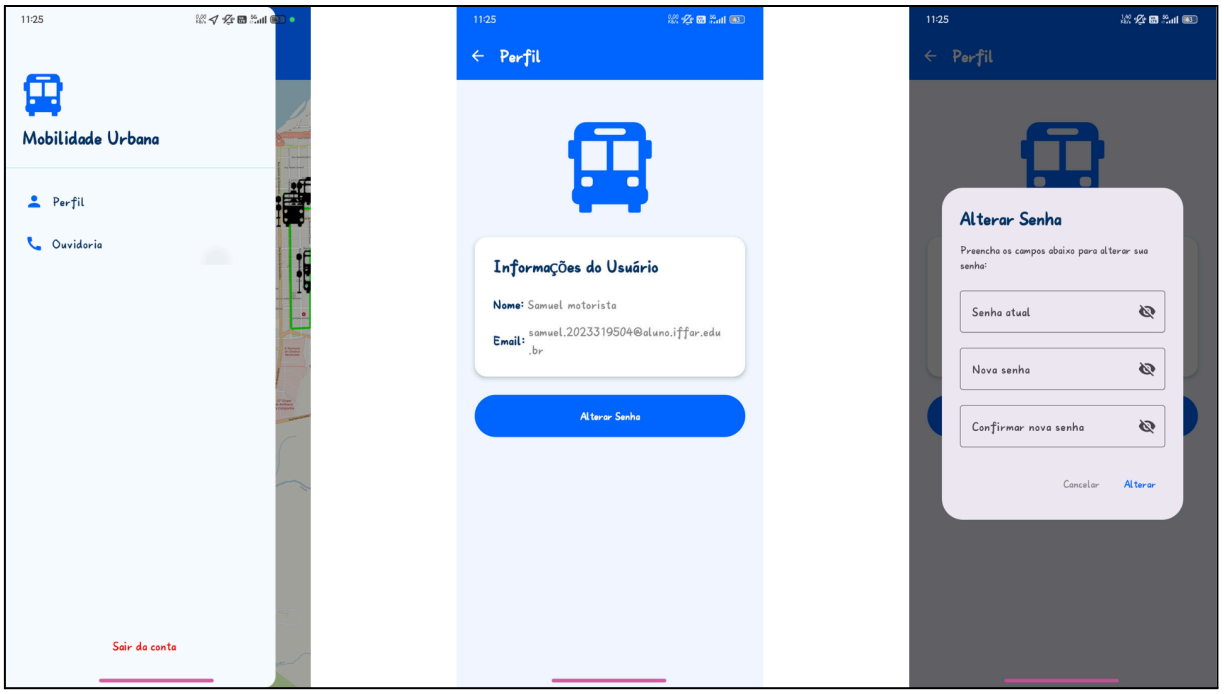
Figura 3 – Tela Principal (Home)



Fonte: Autoria Própria(2025).

A Figura 4 apresenta a tela de perfil do motorista. Um ícone de ônibus azul de 120dp aparece no topo, seguido por card branco com informações do usuário: nome completo e e-mail obtidos do Firebase Authentication. O botão "Alterar Senha" em azul ocupa toda a largura do card. Ao clicar, abre-se diálogo modal com três campos: senha atual, nova senha e confirmar nova senha, todos com ícones de visibilidade. O sistema valida comprimento mínimo de 6 caracteres e verifica se as senhas coincidem antes de permitir a alteração. Mensagens de sucesso (verde) ou erro (vermelho) aparecem abaixo do botão.

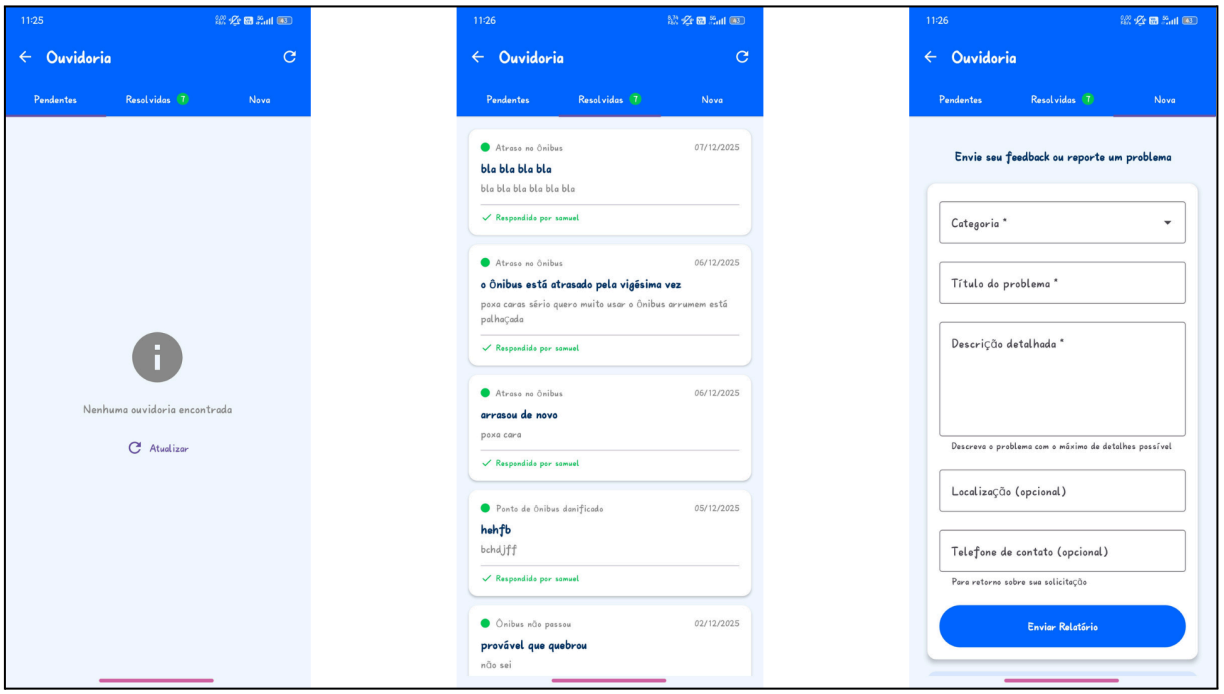
Figura 4 – Tela de Perfil



Fonte: Autoria Própria(2025).

A Figura 5 exibe a tela de ouvidoria organizada em três abas: "Pendentes" (badge laranja), "Resolvidas" (badge verde) e "Nova". A aba Pendentes lista cards brancos com indicador colorido, categoria, título, data e descrição resumida. A aba Resolvidas exibe estrutura similar com informação adicional do administrador responsável. A aba Nova apresenta formulário completo com dropdown de 10 categorias, campos de título e descrição detalhada (textarea de 180dp), além de campos opcionais para localização e telefone. Card informativo azul claro explica políticas de privacidade e prazo de resposta. Diálogo de detalhes mostra informações completas incluindo protocolo (8 caracteres do ID) e, quando resolvido, resposta do administrador em card verde claro.

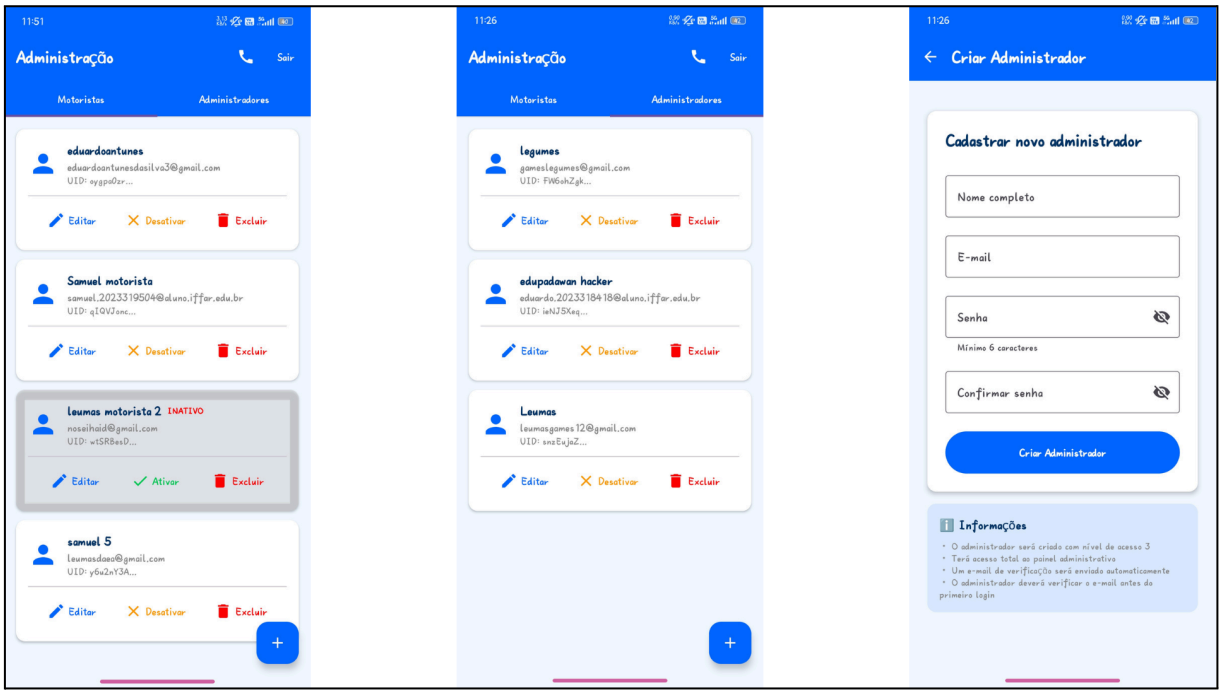
Figura 5 – Tela de Ouvidoria do Motorista



Fonte: Autoria Própria(2025).

A Figura 6 demonstra as telas de gerenciamento de usuários do sistema. A tela principal (esquerda) possui TopAppBar azul com ícone de ouvidoria e botão "Sair", além de TabRow alternando entre "Motoristas" e "Administradores" com badges indicando quantidades. Cards de usuários exibem ícone, nome, e-mail, UID parcial e três botões de ação: Editar (azul, ícone lápis), Ativar/Desativar (verde/laranja, ícone check/close) e Excluir (vermelho, ícone lixeira). Usuários inativos aparecem em fundo cinza claro. FAB azul com ícone "+" no canto inferior direito permite adicionar novos usuários. As telas de criação de motorista e administrador compartilham layout idêntico: card branco centralizado com campos de nome completo, e-mail, senha e confirmar senha (com ícones de visibilidade), botão principal azul "Criar Motorista/Administrador", e card informativo azul claro explicando nível de acesso (2 para motorista, 3 para administrador) e processo de verificação de e-mail obrigatória.

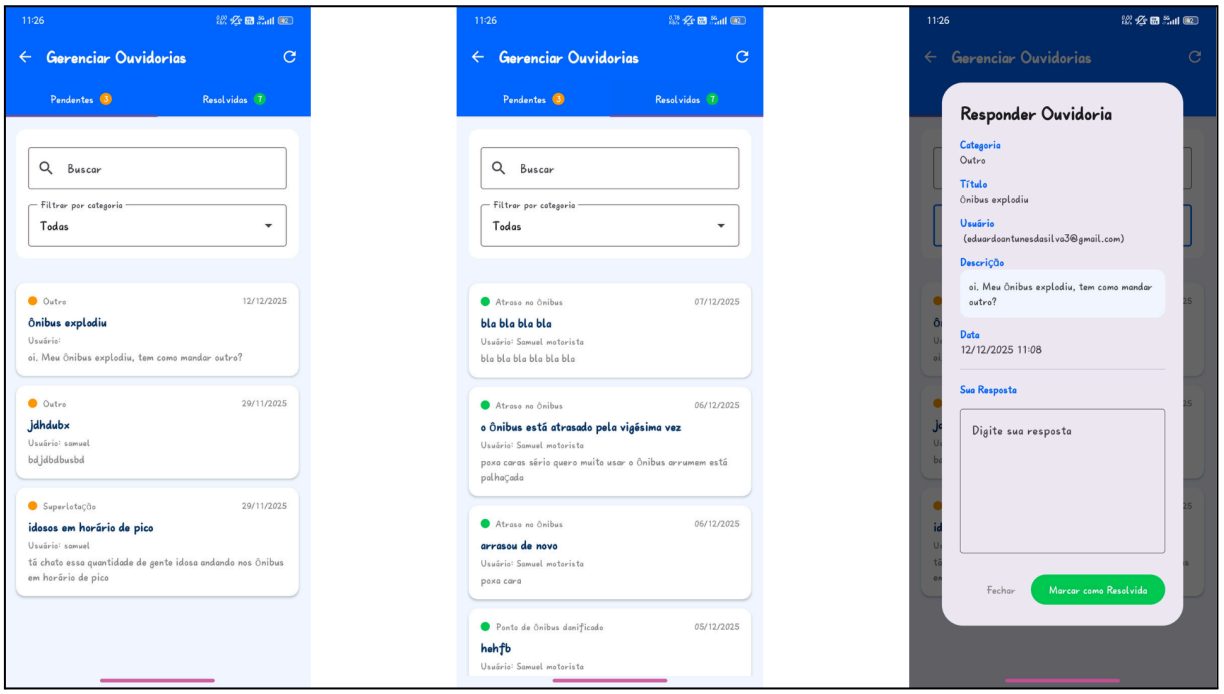
Figura 6 – Painel Administrativo - Gerenciamento de Usuários



Fonte: Autoria Própria(2025).

A Figura 7 apresenta a tela de gerenciamento de ouvidorias pelo administrador. TopAppBar azul contém botão voltar, título "Gerenciar Ouvidorias" e ícone de atualização. TabRow alterna entre abas "Pendentes" (badge laranja) e "Resolvidas" (badge verde) mostrando quantidades. Card branco de filtros oferece campo de busca com ícone de lupa e dropdown de categorias (11 opções incluindo "Todas"). Lista exibe cards de ouvidorias com indicador colorido de status, categoria, título, nome do usuário, descrição resumida e data. Cards são clicáveis para abrir diálogo de resposta. O diálogo apresenta todas as informações da ouvidoria em LazyColumn: categoria, título, usuário com e-mail, descrição em card azul claro (#F0F7FF), localização e telefone (quando disponíveis), data/hora completa, divisor horizontal e textarea para resposta do administrador. Para ouvidorias pendentes, botão verde "Marcar como Resolvida" salva a resposta com timestamp e nome do administrador no Firestore.

Figura 7 – Gerenciamento de Ouvidorias - Administrador



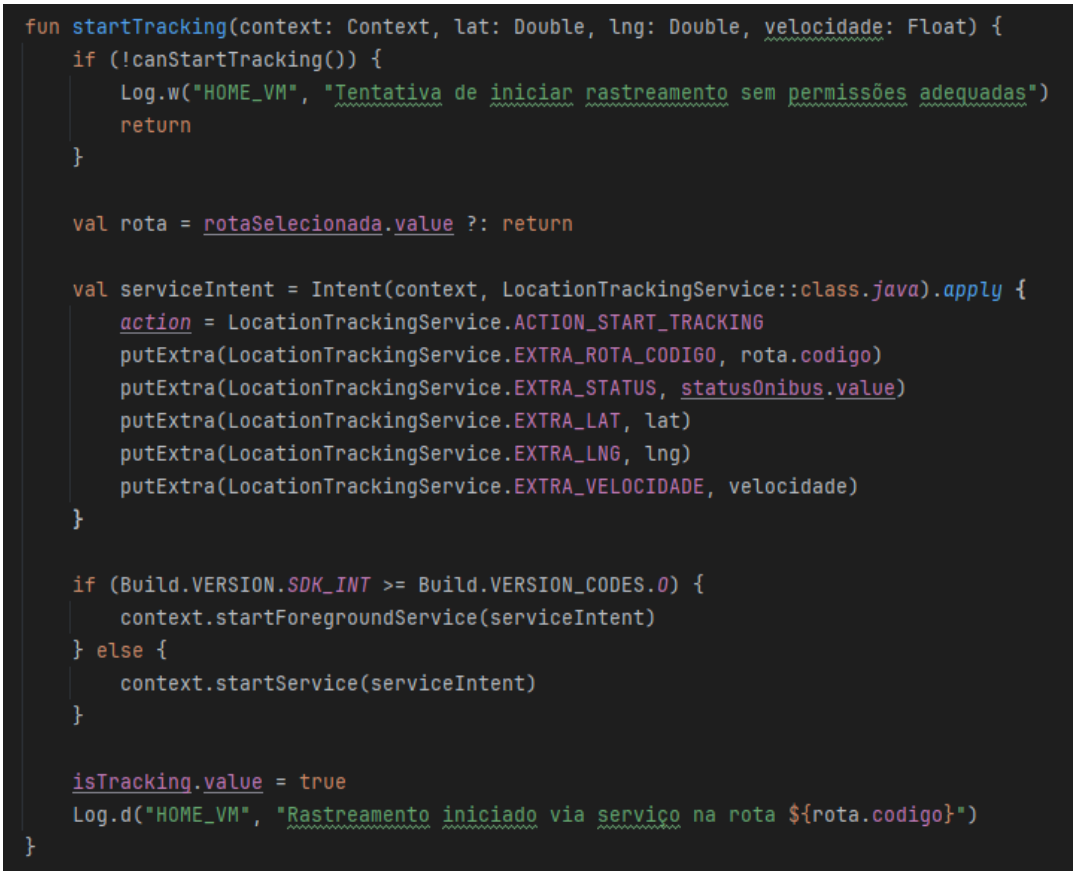
Fonte: Autoria Própria(2025).

3.5.2 Figuras do código

A seguir as figuras referentes a partes fundamentais do código:

A Figura 8 apresenta a função startTracking do HomeViewModel, responsável por iniciar o rastreamento em tempo real. A função primeiro verifica se o motorista concedeu todas as permissões necessárias através de canStartTracking(). Em seguida, cria uma Intent com os dados iniciais (código da rota, status, localização e velocidade) e inicia o LocationTrackingService. A partir da versão Android O (API 26), o serviço é iniciado como Foreground Service, garantindo que continue executando mesmo em segundo plano.

Figura 8 – Função startTracking (HomeViewModel)



```
fun startTracking(context: Context, lat: Double, lng: Double, velocidade: Float) {
    if (!canStartTracking()) {
        Log.w("HOME_VM", "Tentativa de iniciar rastreamento sem permissões adequadas")
        return
    }

    val rota = rotaSelecionada.value ?: return

    val serviceIntent = Intent(context, LocationTrackingService::class.java).apply {
        action = LocationTrackingService.ACTION_START_TRACKING
        putExtra(LocationTrackingService.EXTRA_ROTA_CODIGO, rota.codigo)
        putExtra(LocationTrackingService.EXTRA_STATUS, statusOnibus.value)
        putExtra(LocationTrackingService.EXTRA_LAT, lat)
        putExtra(LocationTrackingService.EXTRA_LNG, lng)
        putExtra(LocationTrackingService.EXTRA_VELOCIDADE, velocidade)
    }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        context.startForegroundService(serviceIntent)
    } else {
        context.startService(serviceIntent)
    }

    isTracking.value = true
    Log.d("HOME_VM", "Rastreamento iniciado via serviço na rota ${rota.codigo}")
}
```

Fonte: Autoria Própria(2025).

A Figura 9 mostra updateLocationAndVehicleInfo, função que atualiza as informações do ônibus no Realtime Database. É chamada a cada 2 segundos pelo LocationCallback. Primeiro obtém o endereço através do Geocoder, depois cria um HashMap com todos os dados (localização, velocidade, status, endereço) e envia para o nó "onibus" no Realtime Database. Isso permite que o aplicativo dos passageiros visualize a posição do ônibus em tempo real.

Figura 9 – Função updateLocationAndVehicleInfo (LocationTrackingService)

```
private fun updateLocationAndVehicleInfo(location: Location) {
    val user = auth.currentUser
    if (user == null || !isTracking || rotaCodigo == null) {
        Log.w("LocationService", "Não pode atualizar: user=$user, tracking=$isTracking, rota=$rotaCodigo")
        return
    }

    // Obtém o endereço em uma coroutine
    serviceScope.launch {
        val endereco = getAddressFromLocation(location.latitude, location.longitude)

        // Atualiza TUDO na coleção onibus no Realtime Database
        val onibusData = hashMapOf<String, Any>(
            "uid" to user.uid,
            "status" to statusOnibus,
            "rotaCodigo" to rotaCodigo!!,
            "lat" to location.latitude,
            "lng" to location.longitude,
            "localizacao" to endereco,
            "velocidade" to location.speed,
            "timestamp" to ServerValue.TIMESTAMP
        )

        realtimeDatabase.getReference("onibus")
            .child(user.uid)
            .setValue(onibusData)
            .addOnSuccessListener {
                Log.d("LocationService", "Informações completas atualizadas no Realtime Database: $endereco")
            }
            .addOnFailureListener { e ->
                Log.e("LocationService", "Erro ao atualizar informações no Realtime Database", e)
            }
    }
}
```

Fonte: Autoria Própria(2025).

A Figura 10 apresenta checkLocationPermissions, função que verifica o status das permissões de localização. Verifica tanto localização precisa (FINE_LOCATION) quanto aproximada (COARSE_LOCATION). Para Android 10 (API 29) ou superior, também verifica permissão de localização em segundo plano (BACKGROUND_LOCATION), essencial para manter o rastreamento ativo quando o aplicativo não está em primeiro plano.

Figura 10 – Função checkLocationPermissions (HomeViewModel)

```
fun checkLocationPermissions(context: Context) {
    val fineLocation = ContextCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_FINE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED

    val coarseLocation = ContextCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_COARSE_LOCATION
    ) == PackageManager.PERMISSION_GRANTED

    hasLocationPermission.value = fineLocation || coarseLocation

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        hasBackgroundPermission.value = ContextCompat.checkSelfPermission(
            context,
            Manifest.permission.ACCESS_BACKGROUND_LOCATION
        ) == PackageManager.PERMISSION_GRANTED
    } else {
        hasBackgroundPermission.value = true
    }

    Log.d("HOME_VM", "Permissões: location=$fineLocation, background=${hasBackgroundPermission.value}")
}
```

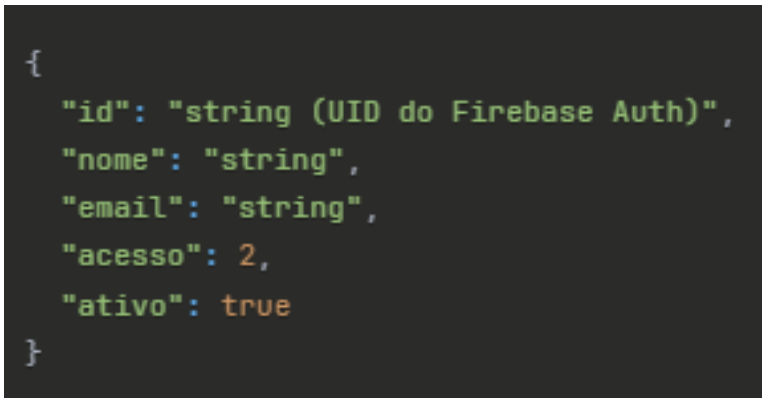
Fonte: Autoria Própria(2025).

3.5.3 Figuras do banco de dados

O sistema utiliza dois bancos de dados Firebase: Firestore para dados permanentes (usuários, rotas, paradas) e Realtime Database para rastreamento em tempo real.

A Figura 11 apresenta a coleção "usuarios", que armazena informações dos motoristas. O campo "acesso" define o nível de permissão (2 = motorista, 3 = administrador). O campo "ativo" permite desativar usuários sem excluí-los do banco. O ID do documento é o mesmo UID gerado pelo Firebase Authentication.

Figura 11 – Estrutura da Coleção usuários



Fonte: Autoria Própria(2025).

A Figura 12 apresenta a coleção “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 12 – Estrutura da Coleção rotas

```
{
  "codigo": "string (ID do documento)",
  "nome": "string",
  "cor": "string (hexadecimal)",
  "codigo": [
    {
      "lat": number,
      "lng": number
    }
  ],
  "paradas": [
    {
      "id": "string",
      "nome": "string",
      "lat": number,
      "lng": number
    }
  ],
  "horarios": {
    "Segunda": ["07:00-08:00", "12:00-13:00"],
    "Terça": ["07:00-08:00", "12:00-13:00"]
  }
}
```

Fonte: Autoria Própria(2025).

A Figura 13 apresenta o nó "ônibus" no Realtime Database, que armazena temporariamente a localização de cada ônibus em rastreamento. Cada motorista tem um subnó identificado por seu UID. Os dados são atualizados a cada 2 segundos enquanto o rastreamento está ativo e removidos quando o motorista para o rastreamento.

Figura 13 – Estrutura do Nó ônibus

```
{
  "onibus": {
    "UID_DO_MOTORISTA": {
      "uid": "string",
      "status": "string",
      "rotaCodigo": "string",
      "lat": number,
      "lng": number,
      "localizacao": "string (endereço)",
      "velocidade": number,
      "timestamp": number (ServerValue.TIMESTAMP)
    }
  }
}
```

Fonte: Autoria Própria(2025).

A Figura 14 apresenta a coleção "ouvidoria", que armazena reclamações e sugestões dos motoristas. O campo "categoria" classifica o tipo de problema. Os campos "resposta", "nomeAdministrador" e "dataResposta" são preenchidos quando um administrador resolve a ouvidoria.

Figura 14 – Estrutura da Coleção ouvidoria

Fonte: Autoria Própria(2025).

4 CONSIDERAÇÕES FINAIS

O desenvolvimento deste Trabalho de Conclusão de Curso representou uma oportunidade de aplicar os conhecimentos adquiridos ao longo do Curso Técnico em Informática Integrado ao Ensino Médio em uma solução prática e relevante para a mobilidade urbana de Uruguaiana. O aplicativo desenvolvido para motoristas complementa o sistema voltado aos passageiros (desenvolvido por Eduardo Antunes da Silva), formando um ecossistema integrado de informações sobre o transporte público municipal.

A escolha das tecnologias — Kotlin como linguagem principal, Jetpack Compose para construção declarativa de interfaces, Firebase (Authentication, Firestore e Realtime Database) para backend, OSMdroid para visualização de mapas e Google Play Services Location para captura de coordenadas GPS — possibilitou criar um sistema funcional, responsivo, escalável e de fácil manutenção. Destaca-se especialmente a implementação do LocationTrackingService como Foreground Service, que garante o rastreamento contínuo da localização do ônibus mesmo quando o aplicativo está em segundo plano, respeitando as boas práticas do Android em termos de consumo de bateria e privacidade do usuário.

Os objetivos propostos foram plenamente alcançados: o motorista pode visualizar e selecionar rotas cadastradas, alterar o status operacional do ônibus (em operação, fora da rota, parado, em manutenção ou fora de serviço), iniciar e parar o rastreamento GPS, acessar seu perfil com possibilidade de alteração de senha e utilizar o sistema de ouvidoria para reportar problemas ou sugestões. A integração com o Realtime Database permite que essas informações sejam sincronizadas automaticamente com o aplicativo dos passageiros, com atualizações a cada 2 segundos durante o rastreamento ativo.

O painel administrativo desenvolvido possibilita o gerenciamento completo de usuários (motoristas e administradores), incluindo funcionalidades de criação, edição, ativação/desativação e exclusão de contas, além da moderação do sistema de ouvidoria com respostas às solicitações dos motoristas. Essa estrutura administrativa garante controle e governança sobre o sistema.

Como trabalhos futuros, sugere-se: implementação de notificações push para alertar motoristas sobre mudanças de rota ou comunicados da administração; integração com sistemas de controle de combustível e manutenção preventiva dos

veículos; desenvolvimento de dashboards analíticos para gestores visualizarem métricas de desempenho das linhas (pontualidade, velocidade média, tempo de paradas); implementação de histórico de viagens para auditoria e planejamento; e otimização de rotas baseada em dados históricos de tráfego e demanda.

Este projeto demonstra como a tecnologia mobile pode ser aplicada para melhorar serviços públicos essenciais, contribuindo para uma mobilidade urbana mais eficiente, transparente e centrada nas necessidades dos cidadãos. A experiência adquirida no desenvolvimento deste sistema fornece uma base sólida para futuros projetos na área de desenvolvimento de software e sistemas de informação geográfica.

REFERÊNCIAS

ABREU, Gabriela de Sousa; ARAÚJO, Giovanna Gabriela de Oliveira; BARBOSA, Letícia de Oliveira; RIBEIRO, Samuel de Andrade; FERRARI, Tatiana Aparecida Debolete. **Tecbus**, 2022. Trabalho de Conclusão de Curso (Técnico em administração) - Escola Técnica Estadual Etec Antonio Devisate, Marília, 2022. Disponível em: https://ric.cps.sp.gov.br/handle/123456789/14460.

SANTOS, Ramon; OLIVEIRA, Amanda. **CrowdBus: Mobilidade urbana colaborativa no Brasil**. 2019. Disponível em: https://www.aedb.br/seget/arquivos/artigos14/41620481.pdf

GOOGLE PLAY STORE. **Cadê o Ônibus?**. Disponível em: https://play.google.com/store/apps/details?id=br.nanoit.viewbus. Acesso em: 20 jul. 2025.

UBER. **Site oficial da Uber Brasil.** Disponível em: https://www.uber.com/br/pt-br. Acesso em: 20 jul. 2025.

ARAÚJO, M. R. M. de et al. *Transporte público coletivo: discutindo acessibilidade, mobilidade e qualidade de vida.* Psicologia & Sociedade, Florianópolis, v. 23, n. 3, p. 574–582, dez. 2011. Disponível em: https://www.scielo.br/j/psoc/a/XWXTQXKJ44BtT5Qw7dLWgvF/?format=html&lang=pt. Acesso em: 15 dez. 2025.