



UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Um Sistema Computacional para Transporte PÚBLICO com Acesso via NFC e Rastreamento com GPS

Lucas Prado Ribeiro

Boa Vista - RR

Março de 2025

Lucas Prado Ribeiro

Um Sistema Computacional para Transporte Públco com Acesso via NFC e Rastreamento com GPS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Roraima como requisito para a obtenção do grau de bacharel em Ciência da Computação.

Orientador(a)

Dr. Herbert Oliveira Rocha

Universidade Federal de Roraima

Departamento de Ciência da Computação

Boa Vista - RR

Março de 2025

DECLARAÇÃO DE AUTORIA

Eu, **Lucas Prado Ribeiro** (código de matrícula **2020018187**), autor da(o) monografia/TCC (Trabalho de Conclusão de Curso), declaro que o trabalho em referência é de minha total autoria e de minha inteira responsabilidade o texto apresentado. Declaro, ainda, que as citações e paráfrases dos autores estão indicadas com as respectivas obras e anos de publicação. Declaro, para os devidos fins que estou ciente:

- dos Artigos 297 a 299 do Código Penal, Decreto-Lei n. 2.848 de 7 de dezembro de 1940;
- da Lei n. 9.610, de 19 de fevereiro de 1998, sobre os Direitos Autorais; e
- que plágio consiste na reprodução de obra alheia e submissão da mesma como trabalho próprio ou na inclusão, em trabalho próprio, de ideias, textos, tabelas ou ilustrações (quadros, figuras, gráficos, fotografias, retratos, lâminas, desenhos, organogramas, fluxogramas, plantas, mapas e outros) transcritos de obras de terceiros sem a devida e correta citação da referência.

O corpo docente responsável pela avaliação deste trabalho poderá não aceitar o referido trabalho caso os pontos mencionados acima sejam descumpridos, por conseguinte, considerar-me reprovado.

Assinatura do acadêmico(a)
Boa Vista - RR, data (por extenso).

FOLHA DE APROVAÇÃO

Monografia de Graduação sob o título **Um Sistema Computacional para Transporte Público com Acesso via NFC e Rastreamento com GPS** apresentada por <**Lucas Prado Ribeiro**> e aceita pelo Departamento de Ciência da Computação da Universidade Federal de Roraima, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Titulação e nome do(a) orientador(a)

Orientador(a)

Departamento

Universidade

Titulação e nome do(a) membro da banca examinadora

Co-orientador(a), se houver

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Boa Vista - RR, data de aprovação (por extenso).

Dedico com muito amor e felicidade, às duas pessoas que são a razão do meu viver, que sempre estiveram ao meu lado, que me criaram e cuidaram de mim. Minha mãe, Leilany, e minha avó, Rosimar.

Agradecimentos

Primeiramente, agradeço a Deus por me dar forças ao longo dessa jornada. Agradeço também à minha família por estar sempre ao meu lado em todos os momentos, pelos conselhos e pelo apoio incondicional. Agradeço aos meus amigos e colegas pela companhia, pela ajuda durante esses anos de faculdade, e aos meus queridos professores do DCC pelos ensinamentos e por contribuírem para o meu desenvolvimento.

A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.

Arthur Schopenhauer

Um Sistema Computacional para Transporte Públco com Acesso via NFC e Rastreamento com GPS

Autor: Lucas Prado Ribeiro

Orientador: Dr. Herbert Oliveira Rocha

Resumo

O transporte público de Boa Vista/RR utiliza cartões com tecnologia RFID para o pagamento das passagens, mas apresenta limitações, como a ausência de um sistema eficiente de recarga e a falta de rastreamento dos ônibus. Este estudo propõe o desenvolvimento de um sistema baseado em uma aplicação móvel construída com Flutter e tecnologia NFC, que permitirá realizar recargas, consultar o saldo e o histórico de transações, além de oferecer a funcionalidade de rastrear a localização dos ônibus em tempo real através do GPS. De forma complementar, será implementado um sistema com leitor RFID e módulo GPS para simular o funcionamento do transporte público. Essa solução visa melhorar a eficiência e a conveniência para os usuários no sistema de recarga de cartões de ônibus.

Palavras-chave: Aplicativo móvel, RFID, Flutter.

Listas de figuras

Figura 1 – Exemplo de código Flutter - Hello World	18
Figura 2 – Exemplo de código Dart Class	20
Figura 3 – Exemplo de estrutura banco de dados MySQL.	22
Figura 4 – Exemplo de estrutura banco de dados Firebase.	24
Figura 5 – Exemplo de código consulta Firebase	25
Figura 6 – Exemplo de interação NFC.	26
Figura 7 – LoRaWAN Architecture.	29
Figura 8 – Elementos do Microcontrolador.	30
Figura 9 – ESP32-DevKitC V4 com ESP-WROOM-32.	32
Figura 10 – Ciclo de vida do sistema.	40
Figura 11 – Diagrama de fluxo do app.	41
Figura 12 – Diagrama de fluxo do sistema do ônibus.	42
Figura 13 – Modelo de dados Usuários.	43
Figura 14 – Modelo de dados Cartão.	43
Figura 15 – Modelo de dados Ônibus.	44
Figura 16 – Telas de cadastro e login.	45
Figura 17 – Código NFC Service	47
Figura 18 – Telas de recarga Pix.	48
Figura 19 – Telas de recarga cartão de crédito.	49
Figura 20 – Telas de gps.	51
Figura 21 – Esquema de conexões do sistema do ônibus.	53
Figura 22 – Protótipo do sistema do ônibus.	57
Figura 23 – Class EmailValidator	59
Figura 24 – Caso de uso 1 cadastro NFC.	62
Figura 25 – Caso de uso 1 cadastro S/ NFC.	63
Figura 26 – Caso de uso 1 cadastro inválido.	64
Figura 27 – Resultado do caso de uso 1.	64
Figura 28 – Rota do experimento	66

Figura 29 – Resultado do caso de uso 2 67

Lista de tabelas

Tabela 1 – Classificação dos artigos por técnicas.	37
--	----

Sumário

1	INTRODUÇÃO	13
1.1	Motivação	14
1.2	Definição do Problema	14
1.3	Objetivos	15
1.3.1	Objetivos Específicos	15
1.4	Organização do Trabalho	15
2	FUNDAMENTOS TEÓRICOS	17
2.1	Flutter	17
2.1.1	Dart	19
2.2	Tipos de banco de dados	21
2.2.1	MySQL	21
2.2.2	Firebase Database	23
2.3	NFC - <i>Near Field Communication</i>	25
2.4	Geolocalização	27
2.4.1	GPS	27
2.4.2	LoRaWAN	28
2.5	Microcontroladores	30
2.5.1	ESP32	31
3	TRABALHOS CORRELATOS	33
3.1	IOT based smart bus system	33
3.2	Smart Bus Ticketing System	34
3.3	RFID Based Bus Ticket Generation System	34
3.4	Análise do sistema de Mobile Payment implementado no transporte público na cidade de São Paulo	35
3.5	Correlações entre os trabalhos e a pesquisa	36
4	SOLUÇÃO PROPOSTA	39

4.1	Arquitetura	39
4.2	Ferramentas e Implementações	42
4.2.1	Modelagem de Dados	43
4.2.2	Aplicativo móvel em Flutter	44
4.2.2.1	Telas de Gerenciamento do Cartão de Acesso	45
4.2.2.2	Telas de Rastreamento do Ônibus	50
4.2.3	Sistema de Controle de Acesso e Comunicação do Ônibus . . .	52
5	AVALIAÇÃO EXPERIMENTAL	56
5.1	Projeto da Avaliação Experimental	56
5.2	Execução e Resultado da Avaliação Experimental	58
5.2.1	Caso de Uso 1: Cadastro de usuários e cartões RFID no sistema	61
5.2.2	Caso de Uso 2: Simulação de Rastreamento de Ônibus na UFRR	65
6	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	69
	Referências	71

1 Introdução

Nota-se que o transporte público é amplamente utilizado por milhões de pessoas para se deslocarem aos ambientes de trabalho, de estudo e de lazer, especialmente em áreas metropolitanas densamente povoadas. Dentro desse contexto, os passageiros fazem uso de cartões de ônibus (equipado com tecnologia de radiofrequência) para pagamento de tarifas, prática que tem se destacado como eficiente e conveniente. No entanto, as filas são um problema constante para quem necessita comprar ou recarregar seu bilhete, não sendo raros os dias em que o sistema está lento ou, muitas vezes, inoperante. Assim, o usuário perde muito do seu tempo nas filas (RICARDO; FREITAS, 2017).

Esperar por um ônibus pode ser extremamente demorado e ineficiente, podendo superar 30 minutos até a chegada do transporte, como observado na cidade de Boa Vista-RR. De acordo Shah et al. (2020), a falta de sistemas precisos de rastreamento contribui para esse problema, tornando impossível saber a localização exata dos ônibus. Além disso, a necessidade de comprar passagens com dinheiro em espécie e lidar com troco ao entrar no ônibus gera desconforto adicional para os passageiros.

A eficiência de uma aplicação móvel para gerenciamento de recarga de cartões de ônibus e localização oferece uma solução prática e acessível para os usuários do transporte público. Neste sentido, este trabalho apresenta um sistema de bilhetagem, baseado em Shah et al. (2020), sem o uso do dinheiro em espécie e sem complicações, sendo a tarifa deduzida diretamente na carteira do aplicativo. Além disso, a solução proposta apresenta aos passageiros a localização em tempo real do ônibus via um microcontrolador. Os resultados preliminares apresentados a seguir, apontam a solução proposta como promissora e capaz de lidar com os desafios apontados.

1.1 Motivação

A implementação de soluções tecnológicas, como o uso de NFC e geolocalização, podem oferecer um avanço considerável na eficiência do transporte, tornando-o mais conveniente e acessível para os usuários. Além disso, soluções como a eliminação de pontos físicos de recarga, conforme proposto por Shah et al. (2020), podem reduzir custos operacionais e melhorar a experiência do usuário, motivando a busca por uma solução prática e inovadora.

1.2 Definição do Problema

A qualidade de vida nas cidades urbanas está diretamente ligada à eficiência do transporte público. Atualmente, a falta de informações sobre a localização dos ônibus e a necessidade de recarregar os cartões em pontos físicos geram desconforto e perda de tempo para os passageiros, impactando negativamente a experiência de deslocamento. A modernização do sistema de transporte público, com foco na implementação de plataformas digitais para recarga de cartões e na disponibilização de informações em tempo real sobre a localização dos ônibus, é crucial para melhorar a mobilidade urbana e garantir um deslocamento mais ágil e tranquilo para os cidadãos. Como destacado por Ricardo e Freitas (2017), a ausência de informações precisas sobre o transporte público causa frustração e impacta negativamente a qualidade de vida nas cidades.

Diante dos desafios enfrentados pela mobilidade urbana, este trabalho se propõe a investigar a seguinte questão: **Como desenvolver um sistema computacional que otimize o gerenciamento de pagamentos de ônibus e permita o rastreamento dos veículos em tempo real, com estimativa de tempo de chegada, visando a melhoria da experiência do usuário e a otimização da mobilidade urbana?**

1.3 Objetivos

Este trabalho tem como objetivo principal otimizar a experiência do usuário e a mobilidade urbana por meio da prototipação e avaliação de um sistema computacional para gerenciamento de acesso e rastreamento de ônibus. O sistema proposto fornecerá informações precisas e em tempo real sobre a localização dos ônibus e a estimativa do tempo de chegada em cada ponto de parada, tornando o transporte público mais eficiente, confiável e acessível para todos.

1.3.1 Objetivos Específicos

Os objetivos específicos são:

1. Implementar um aplicativo móvel para permitir a recarga e consulta de saldo dos cartões de transporte público;
2. Projetar e avaliar um sistema computacional para pagamento e rastreamento de ônibus;
3. Integrar o aplicativo móvel e um sistema embarcado com acesso sem fio e geolocalização com um banco de dados central;
4. Validar o método proposto por meio de testes funcionais e casos de uso simulado.

1.4 Organização do Trabalho

A introdução deste trabalho apresentou: o contexto, definição do problema, motivação, objetivos e contribuições deste trabalho. Os capítulos restantes são organizados da seguinte forma:

- No Capítulo 2, **Fundamentos Teóricos**, são apresentados os conceitos abordados neste trabalho, especificamente: Desenvolvimento de aplicati-

vos via Flutter, tipos de banco de dados, tecnologias de comunicação com NFC e geolocalização.

- No Capítulo 3, **Trabalhos Correlatos**, são analisados os trabalhos correlatos a solução proposta.
- No Capítulo 4, **Método da Solução Proposta**, é descrito as etapas de execução do método da solução proposta para projetar e desenvolver um sistema computacional que simplifique o processo de recarga de crédito e a consulta de saldo do cartão de transporte público.
- No Capítulo 5, **Avaliação Experimental**, é apresentado o planejamento e projeto para execução da avaliação da solução proposta;
- E, por fim, no Capítulo 6, **Considerações Finais e trabalhos futuros**, apresenta-se as considerações finais, análise dos experimentos e trabalhos futuros.

2 Fundamentos Teóricos

Este capítulo tem como objetivo apresentar os conceitos fundamentais para o entendimento da solução proposta neste trabalho. Serão abordados, ao longo deste capítulo, os seguintes tópicos: a plataforma *Flutter*, os diferentes tipos de banco de dados, a tecnologia *NFC (Near Field Communication)* e os sistemas de geolocalização, os quais são essenciais para a compreensão e implementação da solução.

2.1 Flutter

O *Flutter* é o kit de desenvolvimento de software (SDK) de interface do usuário portátil do Google para criar aplicações móveis, web e de *desktop* nativamente compiladas. O *Flutter* oferece um ambiente completo com *framework*, *widgets* e ferramentas. *Flutter* também é de código aberto e gratuito, o que significa que pode ser utilizado de maneira simples (BHAGAT et al., 2022).

O conceito central do framework *Flutter* são os *widgets* que representam a estrutura básica da interface do usuário, armazenando configurações como propriedades, layout e comportamento visual. O *Flutter* classifica os *widgets* em *StatelessWidget* e *StatefulWidget*, sendo que o primeiro não possui estado mutável, enquanto o segundo pode atualizar dinamicamente sua interface por meio da função `setState()`. Esse modelo reduz o consumo de recursos, melhora a eficiência da renderização e torna o desenvolvimento mais modular e reutilizável (WANG, 2022).

Uma característica destacada do *Flutter* é o recurso de *hot reload*, que permite a visualização imediata de alterações no código, facilitando o desenvolvimento e a implementação de melhorias na aplicação, tornando-o mais próximo do código de máquina e permitindo que funcione de maneira mais ágil (BHAGAT et al., 2022).

Em comparação com o React Native, o Flutter apresenta uma menor dependência de software de terceiros. Enquanto no React Native, os desenvolvedores frequentemente dependem de bibliotecas externas para utilizar módulos nativos (como NFC, Wi-Fi, GPS, etc.), no Flutter, essa dependência é significativamente reduzida, especialmente no que diz respeito aos widgets que acessam APIs do dispositivo, navegação e outros recursos essenciais (GÜLCÜOĞLU et al., 2021).

Outro diferencial é o fato de seu mecanismo de renderização ser próprio, eliminando a necessidade de camadas intermediárias, como ocorre no React Native, reduzindo perdas de desempenho e proporcionando uma experiência mais próxima do nativo (BHAGAT et al., 2022).

Figura 1 – Exemplo de código Flutter - Hello World

```
1 void main() {  
2     runApp(const MyApp());  
3 }  
4  
5 class MyApp extends StatelessWidget {  
6     const MyApp({Key? key}) : super(key: key);  
7  
8     @override  
9     Widget build(BuildContext context) {  
10         return MaterialApp(  
11             home: Scaffold(  
12                 appBar: AppBar(  
13                     title: Text('Flutter Demo'),  
14                 ),  
15                 body: Text('Hello world :)'),  
16             ),  
17         );  
18     }  
19 }
```

Fonte: Própria do autor.

Por exemplo, a hierarquia de widgets do aplicativo Hello World, apresentado na Figura 1, segue assim:

1. MyApp é o widget criado pelo usuário e é construído usando o widget nativo do Flutter, MaterialApp.
2. MaterialApp possui uma propriedade denominada de home para especificar a interface do usuário da página inicial.

3. Scaffold é o widget nativo do Flutter e tem duas propriedades — body e appBar.
4. Body é usado para especificar sua interface principal, e appBar é usado para especificar sua interface de cabeçalho.

Como exemplo de utilização do framework, temos o trabalho proposto por Weber e Cantarelli (2020), que desenvolveu uma aplicação móvel capaz de realizar agendamentos e controle de horários de atendimento para clientes e prestadores de serviços, atingindo todos objetivos traçados e entregando uma aplicação que atendeu aos requisitos funcionais traçados, onde a escolha do Flutter como framework de desenvolvimento possibilitou uma integração com mais plataformas, entregando uma solução com interfaces intuitivas com a utilização de widgets.

2.1.1 Dart

Dart é uma linguagem de programação que é desenvolvida e mantida pelo Google. É amplamente usada dentro do Google e foi comprovado que tem a capacidade de desenvolver aplicativos da web massivos, como o AdWords (WU, 2018). De acordo com o índice TIOBE, que mede a popularidade das linguagens de programação, Dart atingiu sua posição mais alta em janeiro de 2017, ocupando o 17º lugar. Embora não esteja atualmente entre as linguagens mais populares, sua adoção tem crescido, especialmente devido ao sucesso do framework Flutter, que utiliza Dart como linguagem principal.

A linguagem de programação Dart é utilizada no desenvolvimento de aplicações com Flutter, é otimizada para a criação de aplicativos rápidos em qualquer plataforma. Com uma sintaxe semelhante a outras linguagens populares, sua curva de aprendizado é relativamente baixa para desenvolvedores familiarizados com essas tecnologias. Como uma linguagem orientada a objetos, Dart suporta conceitos fundamentais como classes, herança, interfaces e tipagem opcional. Isso assegura que as aplicações desenvolvidas com Dart

sejam indistinguíveis de aplicações nativas em nível de máquina (BHAGAT et al., 2022).

A capacidade de executar código em várias plataformas, como Windows, Linux e macOS, por meio da Dart Virtual Machine, amplia ainda mais seu uso em diferentes ambientes. Outra característica importante são os recursos de compilação JIT (*Just in Time*) e AOT (*Ahead of Time*), que aumentam o desempenho do Dart. A compilação JIT permite recargas rápidas de código, enquanto a compilação AOT proporciona um tempo de inicialização rápido e um desempenho superior — fatores cruciais para aplicações que exigem respostas ágeis (WANG, 2022).

Figura 2 – Exemplo de código Dart Class

```
1 class Spacecraft {  
2     String name;  
3     DateTime? launchDate;  
4     int? get launchYear => launchDate?.year;  
5     Spacecraft(this.name, this.launchDate) {  
6     }  
7     Spacecraft.unlaunched(String name) : this(name,  
8         null);  
9     void describe() {  
10        print('Spacecraft: $name');  
11        var launchDate = this.launchDate;  
12        if (launchDate != null) {  
13            int years = DateTime.now().difference(  
14                launchDate).inDays ~/ 365;  
15            print('Launched: $launchYear ($years years  
16                ago)');  
17        } else {  
18            print('Unlaunched');  
19        }  
    }
```

Fonte: Dart (2024).

O código da Figura 2 apresenta um exemplo de uma classe com três propriedades, dois construtores e um método. Uma das propriedades não pode ser definida diretamente, sendo acessada por meio de um método `getter` (em vez de uma variável). O método na linha 9 utiliza interpolação de strings para imprimir equivalentes de string de variáveis dentro de literais de string. Entre as vantagens do Dart destaca-se:

- Estrutura de Programação Orientada a Objetos: A estrutura da classe `Spacecraft` exemplifica a programação orientada a objetos, onde propriedades e métodos são encapsulados em um único objeto, promovendo uma melhor organização e reutilização do código. Isso também facilita a manutenção, já que as funcionalidades relacionadas à espaçonave estão agrupadas.
- Null Safety: A utilização do tipo `DateTime?` para a propriedade `launchDate` e `int?` para o getter `launchYear`, nas linhas 3 e 4 do código na Figura 2, demonstra a funcionalidade de `null safety` do Dart. Essa característica garante que as variáveis sejam tratadas de forma segura, evitando erros em tempo de execução relacionados a valores nulos.

2.2 Tipos de banco de dados

Os bancos de dados são fundamentais para a gestão e armazenamento de informações em diversos contextos, e sua escolha pode impactar significativamente o desempenho e a escalabilidade de uma aplicação. Nesta seção, exploraremos dois tipos principais de bancos de dados: relacionais e não relacionais, com foco nas suas características e adequação a diferentes cenários de desenvolvimento.

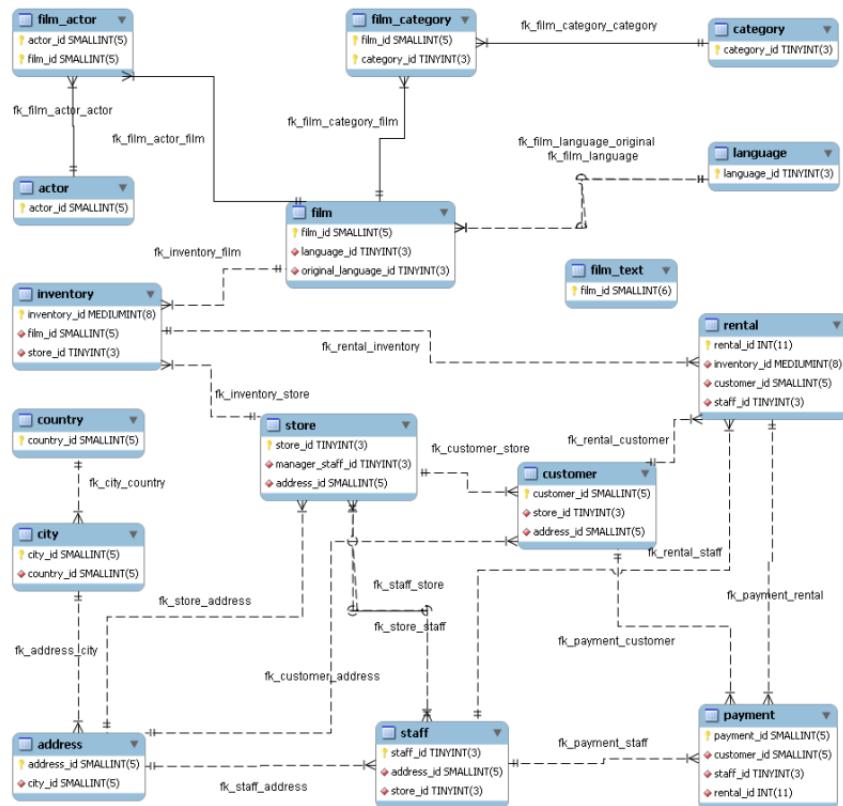
2.2.1 MySQL

Os bancos de dados relacionais, como o MySQL, organizam dados em tabelas estruturadas, onde as informações são armazenadas em linhas e colunas. Cada linha representa um registro, e cada coluna um atributo específico do registro. A estrutura relacional permite a definição clara de entidades e seus relacionamentos, facilitando a manipulação e consulta dos dados através da linguagem SQL (*Structured Query Language*) (MySQL, 2024).

Os comandos SQL podem ser divididos em duas categorias principais: Data Definition Language (DDL), utilizada para criação das estruturas de arma-

zenamento dos dados, como tabelas e índices, e a Data Manipulation Language (DML), responsável pelas operações de manipulação dos dados armazenados, como criação de novos registros, modificação dos dados já existentes e recuperação dos dados salvos. Cada comando SQL possui uma sintaxe própria e bem definida. Todos possuem a especificação de uma ação, por exemplo, criar uma tabela (CREATE TABLE) ou atualizar dados (UPDATE). Em seguida, são informados demais parâmetros que afetam o tipo de estrutura que será criado ou os dados que serão atingidos (COCOLETE et al., 2024).

Figura 3 – Exemplo de estrutura banco de dados MySQL.



Fonte: Internet em (MYSQL, 2024)

A Figura 3 apresenta a visualização e manipulação das propriedades das tabelas em um diagrama de entidade-relacionamento estendido (EER) no MySQL Workbench, utilizando o banco de dados de exemplo **sakila**. As tabelas do banco estão interconectadas por chaves estrangeiras, com exceção da tabela **film_text**. Algumas tabelas, como a **film**, possuem múltiplas relações complexas entre si e com outras tabelas.

tiplas chaves estrangeiras que se relacionam com a mesma tabela, como as chaves `fk_film_language_original` e `fk_film_language`, que se conectam à `language` tabela. Quando há múltiplos relacionamentos entre duas tabelas, as linhas de conexão são executadas simultaneamente.

No diagrama, os relacionamentos de identificação e não identificação são representados por linhas sólidas e quebradas, respectivamente. Por exemplo, a chave estrangeira `category_id` na tabela `film_category` tabela, que faz parte da chave primária, utiliza uma linha sólida para se relacionar com a `category` tabela. Já a chave `country_id` na tabela `city`, que não é parte da chave primária, é representada por uma linha quebrada, indicando um relacionamento de não identificação.

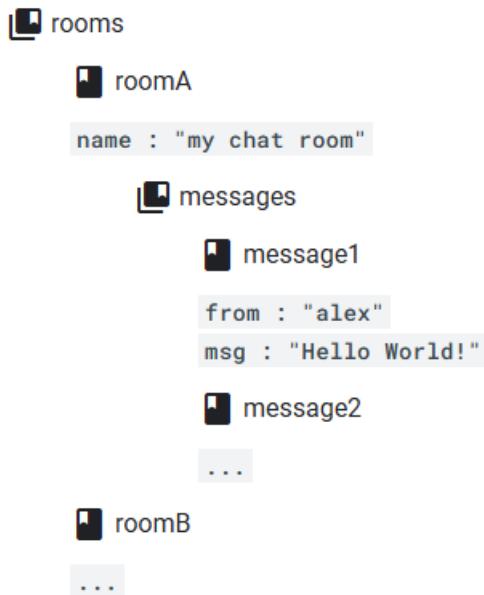
2.2.2 Firebase Database

O Firebase Database é um exemplo de bancos de dados não relacional que oferecem uma abordagem diferente para o armazenamento de dados. No Firebase Cloud Firestore, os dados são armazenados como JSON e sincronizados em tempo real entre todos os clientes conectados. Estes dados são hospedados e armazenados na nuvem. Este modelo é ideal para aplicações que requerem atualizações instantâneas e interatividade, especialmente em ambientes móveis (SUDIARTHA et al., 2020).

Firebase oferece uma variedade de serviços, incluindo autenticação de usuários, banco de dados em tempo real, armazenamento em nuvem e muito mais. Uma das principais vantagens do Firebase é possuir um sistema próprio de autenticação com segurança pré-configurado: o Firebase Authentication, que é utilizado para gerenciar o login e cadastro dos usuários, e o Firestore, que é o banco de dados NoSQL do Firebase, utilizado neste trabalho para armazenar dados de usuários e cartões de acesso. O Firestore sincroniza automaticamente os clientes com os dados mais recentes, é escalável e ideal para aplicações que requerem uma resposta rápida e capacidade de lidar com grandes volumes de

dados (FIREBASE, 2024).

Figura 4 – Exemplo de estrutura banco de dados Firebase.



Fonte: Internet em (FIREBASE, 2024).

O Cloud Firestore do Firebase organiza os dados em documentos e coleções, permitindo uma estrutura hierárquica onde documentos podem conter subcoleções, conforme ilustrado na Figura 4. A coleção `rooms` é a coleção principal que contém as salas de chat, enquanto a subcoleção `messages` contém as mensagens associadas a cada sala. Essa estrutura é especialmente útil para aplicativos que requerem relações complexas entre dados.

O Firebase, como banco de dados NoSQL, oferece uma abordagem otimizada para armazenar e recuperar dados em tempo real, utilizando uma estrutura de documentos e coleções. Essa estrutura facilita a consulta e sincronização dos dados de forma eficiente. Por exemplo, para consultar todas as mensagens de uma sala de chat no Firestore, pode-se utilizar o seguinte código:

Figura 5 – Exemplo de código consulta Firebase

```
1 const roomRef = db.collection('rooms').doc('roomA')
  .collection('messages');
2
3 roomRef.get().then((querySnapshot) => {
4   querySnapshot.forEach((doc) => {
5     console.log(doc.id, ' => ', doc.data());
6   });
7 }).catch((error) => {
8   console.log('Erro: ', error);
9 }) ;
```

Fonte: Própria do autor.

Neste exemplo de código da Figura 5, o Firestore permite que as mensagens da sala A sejam acessadas facilmente por meio de sua hierarquia de dados. A grande vantagem dessa abordagem, como destacado por Sudiartha et al. (2020), é a sincronização automática dos dados em tempo real, o que permite que todas as instâncias do aplicativo sejam atualizadas simultaneamente, sem a necessidade de operações complexas, como em bancos de dados relacionais.

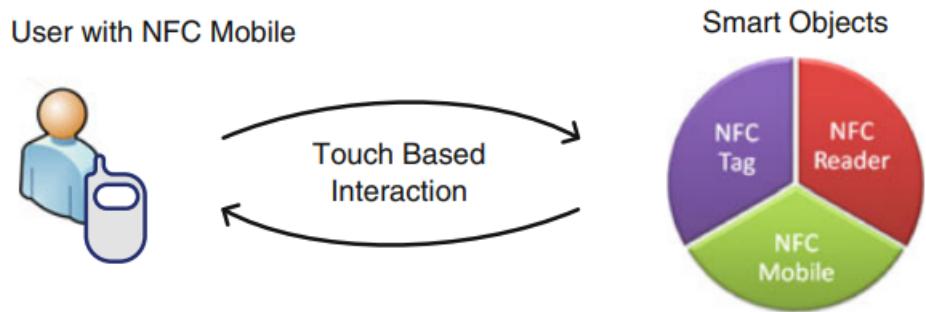
2.3 NFC - *Near Field Communication*

O NFC é uma tecnologia de comunicação sem fio de curto alcance que permite a troca de dados entre dispositivos quando estão próximos um do outro. Essa tecnologia é amplamente utilizada em pagamentos móveis, cartões de transporte público e sistemas de controle de acesso devido à sua conveniência e segurança. Os desenvolvedores de aplicativos móveis podem projetar soluções personalizadas que interagem com tags NFC e outros dispositivos, permitindo a leitura e gravação de dados por meio da conexão baseada em toque (COSKUN et al., 2013).

As tags NFC são cartões de memória sem contato que armazenam dados em um formato especial chamado NDEF (NFC Data Exchange Format). Essas tags vêm em uma variedade de formatos, permitindo diversas implementações em termos de tamanhos, capacidades, custos e funcionalidades. O NFC definiu cinco tipos diferentes de tags que variam no protocolo de comunicação

subjacente e na estrutura de dados para armazenar mensagens NDEF, mas o comportamento geral dessas tags é similar (COSKUN et al., 2013).

Figura 6 – Exemplo de interação NFC.



Fonte: (COSKUN et al., 2013).

De acordo com a Figura 6, o usuário primeiro interage com um objeto inteligente (uma etiqueta NFC, um leitor NFC ou outro telefone celular habilitado para NFC) usando seu telefone celular habilitado para NFC (em resumo: NFC móvel). Após o toque, o NFC móvel pode fazer uso adicional dos dados recebidos ou, alternativamente, usar serviços móveis fornecidos, como abrir uma página da web, fazer uma conexão de serviço da web etc.

A implementação do NFC em aplicativos Flutter é exemplificada através da utilização de funções do pacote `NFC_manager`, onde ao interagir com a interface do usuário, o aplicativo pode verificar se a funcionalidade NFC está disponível no dispositivo e, em seguida, iniciar uma sessão de leitura de tag. Este processo não apenas melhora a experiência do usuário, mas também facilita a gestão e a recarga de cartões de transporte, promovendo a inovação no setor de mobilidade urbana (WANG, 2022).

No trabalho de Ricardo e Freitas (2017) é apresentado um sistema para recarga de cartão para transporte público dos usuários do transporte público de São Paulo, dá um exemplo de como a tecnologia NFC se encaixa neste tipo de problema. Onde utiliza a tecnologia NFC para realizar a transferência de dados de um cartão para o dispositivo, bastando encostar o cartão na parte de trás do aparelho para fazer o pareamento das informações.

2.4 Geolocalização

A geolocalização é a técnica que permite determinar a localização de um dispositivo com base em suas coordenadas geográficas. Este recurso é cada vez mais utilizado em diversas aplicações, especialmente devido à crescente demanda por soluções que integrem dados de localização em contextos econômicos, como a maximização de vendas por meio do processamento de informações geográficas. Tecnologias modernas, como o desenvolvimento de constelações de satélites, exigem precisão elevada na medição de distâncias e separações entre pontos coordenados (JAPARA et al., 2023).

No contexto do Flutter, a implementação da geolocalização é facilitada pelo uso de bibliotecas como `geolocator` e `location`, que fornecem acesso fácil às informações de localização do dispositivo. Estas bibliotecas permitem que os desenvolvedores implementem funcionalidades como rastreamento de localização em tempo real e cálculo de distâncias entre coordenadas. O `geolocator`, por exemplo, oferece métodos para obter a posição atual do dispositivo, além de calcular a distância entre diferentes pontos geográficos (JAPARA et al., 2023).

Esses recursos são especialmente úteis em aplicativos que exigem localização precisa, como serviços de entrega e navegação. A integração da geolocalização em aplicativos Flutter não apenas melhora a experiência do usuário, mas também possibilita a criação de soluções inovadoras que atendem às necessidades do mercado atual, tornando a geolocalização uma competência essencial para desenvolvedores que desejam se destacar na criação de software moderno (JAPARA et al., 2023).

2.4.1 GPS

O Sistema de Posicionamento Global (GPS) é uma tecnologia fundamental que permite a determinação precisa da localização de objetos e pessoas em escala global. Operando por meio de uma constelação de satélites em órbita terres-

tre, o GPS transmite sinais que são captados por receptores em dispositivos como *smartphones*, veículos e relógios. Ao processar esses sinais, o receptor calcula com exatidão a posição geográfica do usuário em qualquer lugar do planeta. Além disso, o GPS fornece informações precisas de tempo, essenciais para diversas aplicações, como sincronização de redes de telecomunicações e sistemas bancários (ZANOTTA et al., 2011).

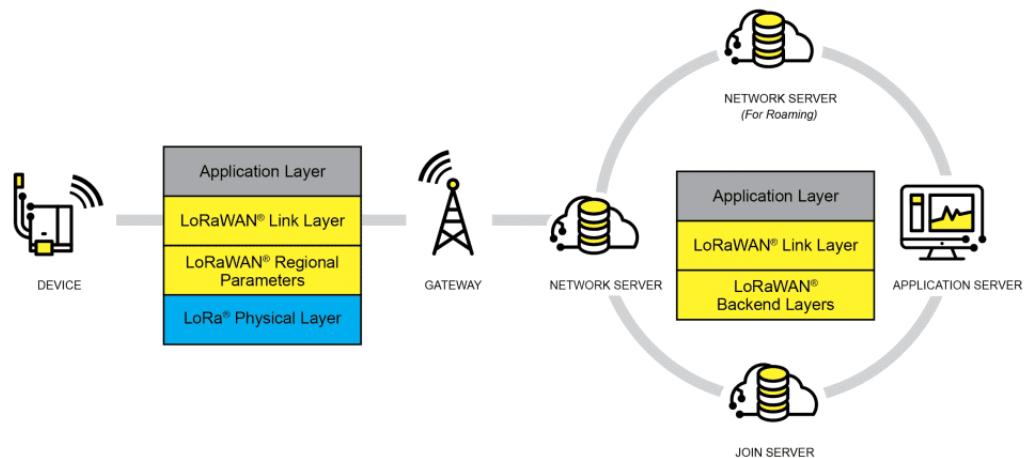
Um dos maiores desafios do sistema de ônibus atual é a falta de um sistema eficiente de rastreamento. Sem um monitoramento preciso, os passageiros ficam sem informações sobre a chegada do próximo ônibus (SHAH et al., 2020). Com o GPS, é possível obter informações em tempo real sobre a localização dos ônibus, permitindo que os passageiros planejem melhor suas viagens e diminuam o tempo de espera nas paradas. Além disso, um sistema de rastreamento pode melhorar a eficiência operacional das empresas de transporte, otimizando rotas e horários dos ônibus com base nos dados coletados.

2.4.2 LoRaWAN

LoRaWAN é uma tecnologia de comunicação sem fio projetada para conectar dispositivos IoT em grandes áreas com baixo consumo de energia. Essa rede permite que sensores, atuadores e dispositivos diversos enviem dados por longas distâncias sem precisar de grandes infraestruturas, opera com baixo consumo de energia e a capacidade de conectar um grande número de dispositivos a uma única rede. A capacidade de longo alcance da camada física LoRa permite links de salto único entre dispositivos finais e gateways. O sistema suporta comunicação bidirecional e endereçamento multicast para uso eficiente do espectro durante tarefas como atualizações de Firmware Over-The-Air (FOTA). A LoRaWAN tem sido amplamente utilizada em projetos de cidades inteligentes, agricultura de precisão e monitoramento ambiental (ALLIANCE, 2022).

Um exemplo de aplicação prática de LoRaWAN pode ser observado no estudo de Jabbar (2024), que desenvolveu um sistema IoT baseado em LoRaWAN para monitoramento da qualidade da água em áreas rurais. O sistema utiliza sensores instalados em corpos d'água para medir parâmetros como pH, temperatura e turbidez. Esses sensores transmitem os dados por meio de LoRaWAN para um servidor central, permitindo o monitoramento em tempo real da qualidade da água. A solução mostrou-se eficaz em fornecer dados confiáveis, mesmo em áreas com pouca infraestrutura de telecomunicações, além de oferecer baixo custo de manutenção e longo tempo de operação, graças ao consumo reduzido de energia dos dispositivos.

Figura 7 – LoRaWAN Architecture.



Fonte: (ALLIANCE, 2022).

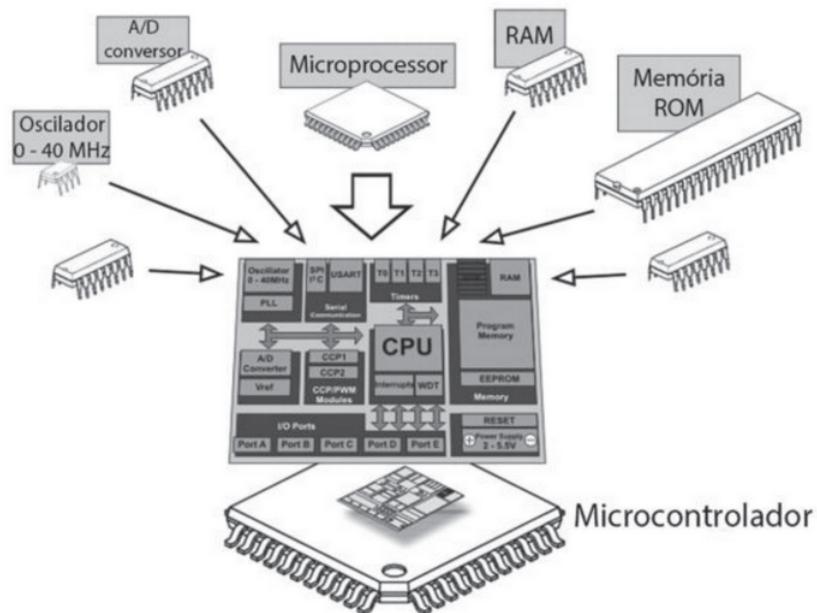
A arquitetura de rede LoRaWAN conforme mostra a Figura 7, usa uma topologia estrela-de-estrelas, onde é composta por várias camadas que colaboram para a comunicação eficiente em redes de IoT. A camada de aplicação interage com os usuários, enquanto a camada LoRaWAN gerencia a comunicação e autenticação entre dispositivos e a rede. O servidor de rede centraliza a gestão do tráfego de dados, enquanto os parâmetros regionais definem especificações operacionais conforme regulamentações locais. Os gateways atuam como intermediários que conectam dispositivos à rede, convertendo sinais RF em pacotes

IP, e a camada física utiliza a tecnologia LoRa para permitir comunicação de longo alcance.

2.5 Microcontroladores

O microcontrolador é um dispositivo considerado um computador completo em miniatura, integrado em um único circuito. Ele contém uma unidade de processamento e os componentes necessários para realizar tarefas de forma autônoma, como memórias de leitura e gravação de dados, armazenamento de *firmware*, conversores analógico-digitais (A/D) e portas programáveis de entrada e saída, conforme ilustrado na Figura 8. Essas portas são essenciais para funções como o controle de outros dispositivos e a interação com o usuário, entre outras (SILVA et al., 2019).

Figura 8 – Elementos do Microcontrolador.



Fonte: (COSTA, 2018).

A utilização de microcontroladores está intimamente ligada à sua programação, uma vez que, sem ela, não seria possível desenvolver sistemas capazes de executar tarefas específicas. A programação é realizada por meio

de linguagens que permitem ao programador interagir com o dispositivo de forma intuitiva. O código-fonte escrito é processado por um compilador, que o converte para uma linguagem de máquina comprehensível pelo microcontrolador (OLIVEIRA, 2012).

A linguagem C é amplamente adotada na programação de microcontroladores devido à sua eficiência e portabilidade. Por ser uma linguagem de alto nível, permite que o programador se concentre no desenvolvimento da aplicação, sem se preocupar com os detalhes de baixo nível do hardware. Além disso, a linguagem C facilita a adaptação de programas para diferentes sistemas, o que a torna uma escolha versátil para o desenvolvimento de sistemas embarcados (OLIVEIRA, 2012).

A variedade de microcontroladores disponíveis no mercado abrange aspectos como capacidade de memória interna, velocidade de processamento, número de pinos de entrada e saída, forma de alimentação, consumo de energia, quantidade de periféricos, arquitetura e conjunto de instruções (MARTINS, 2005). Esses fatores influenciam diretamente a escolha do microcontrolador para aplicações específicas.

2.5.1 ESP32

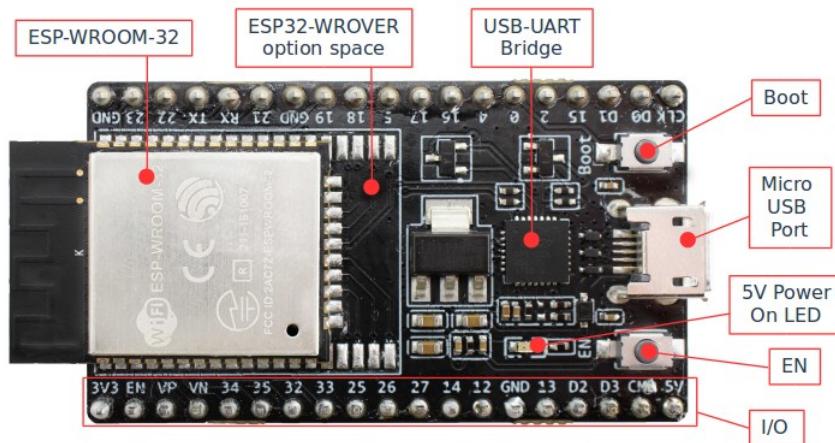
O ESP32 é um microcontrolador desenvolvido pela Espressif Systems, uma empresa chinesa com sede em Xangai. Ele se destaca como uma solução completa para redes Wi-Fi, funcionando tanto como uma ponte entre microcontroladores tradicionais e conexões sem fio quanto como um sistema autônomo capaz de executar aplicações independentes (KOLBAN, 2018).

Lançado comercialmente no final de 2016, o ESP32 é equipado com um processador dual-core Tensilica LX6, operando a uma frequência de até 160 MHz, e 520 KB de RAM. O microcontrolador oferece suporte a diversos periféricos, conversores analógico-digitais (ADC), e interfaces SPI, I2C, I2S e UART, o que o torna aplicável em uma variedade de projetos, desde automação

residencial até sistemas de monitoramento industrial (KOLBAN, 2018).

Para facilitar o uso do ESP32, diversos fabricantes disponibilizam placas de desenvolvimento, como o ESP32-DevKitC, ilustrado na Figura 9. Essa placa é compatível com *breadboards* e possui conectores laterais com 19 pinos de interface para GPIO, clock e alimentação. Também conta com uma porta mini USB para comunicação com o computador e fornecimento de energia, convertendo os 5V da USB para 3,3V. A placa inclui dois botões principais: EN, que reinicia o dispositivo, e BOOT, que ativa o modo de download de *firmware* quando pressionado simultaneamente ao botão EN (IBRAHIM, 2019).

Figura 9 – ESP32-DevKitC V4 com ESP-WROOM-32.



Fonte: (ESPRESSIF, 2023).

Para programar a placa, é necessário conectá-la a um computador via porta USB, utilizando o protocolo de comunicação serial. O ESP32 DevKitC já vem com *firmware* pré-carregado, que é ativado assim que a placa recebe energia. Para a comunicação com o dispositivo, pode-se utilizar softwares de emulação de terminal, como HyperTerm, Putty ou X-CTU, ou, alternativamente, o Arduino IDE, um dos ambientes de desenvolvimento mais comuns para microcontroladores, especialmente para a família Arduino (IBRAHIM, 2019).

3 Trabalhos Correlatos

Este capítulo visa apresentar os principais trabalhos relacionados com o sistema proposto, destacando as diferenças e contribuições ao método proposto. Os estudos abordados utilizam tecnologias de radiofrequência ou referenciam sistemas que fazem uso dessa ferramenta para criar soluções que resolvam problemas nas compras de passagens de ônibus e no rastreio dos ônibus.

3.1 *IOT based smart bus system*

O estudo de Shah et al. (2020) descreve o desenvolvimento de um sistema de reserva automatizada de passagens de ônibus. O sistema proposto calcula a tarifa pela distância percorrida e deduz o valor da tarifa da carteira eletrônica do passageiro, gerenciada no aplicativo móvel S-Bus. O sistema também rastreia a localização dos ônibus em tempo real.

O fluxo do sistema de ônibus inteligente baseado em IOT do Shah et al. (2020) envolve o uso de cartões RFID, que contêm um número exclusivo para cada passageiro. Esses cartões devem ser registrados no S-Bus, onde todos os detalhes do passageiro são associados a esse número e armazenados no banco de dados. Em cada ônibus, há duas unidades de leitores RFID, uma na porta de entrada e outra na de saída, que escaneiam os cartões dos passageiros e atualizam suas localizações no banco de dados. Esses leitores estão conectados ao NodeMCU (Módulo WiFi ESP8266) via Arduino Uno, garantindo conectividade ao banco de dados (SHAH et al., 2020).

No estudo de Shah et al. (2020), os passageiros podem acompanhar a localização atual do ônibus pelo aplicativo S-Bus, que acessa informações no banco de dados e exibe a posição do ônibus em um mapa do Google. O resumo da viagem, incluindo origem, destino e tarifa, também está disponível no aplicativo (SHAH et al., 2020). O sistema elimina o uso de dinheiro e papel,

substituindo os bilhetes físicos por digitais. As análises do sistema ajudam a gerenciar os horários e rotas dos ônibus.

3.2 Smart Bus Ticketing System

A pesquisa de Naila et al. (2020) propõe um sistema inteligente de bilhetagem de ônibus, melhorando a experiência tanto para usuários quanto para prestadores de serviço. A proposta inclui um aplicativo Android com leitor de QR Code e uma carteira digital. Passageiros podem se registrar no aplicativo, e a conta do condutor é controlada pelo administrador, que fornece suas credenciais. Uma vez logado, tanto o condutor quanto o passageiro devem vincular sua conta bancária ao aplicativo, para que possam transferir, adicionar e retirar dinheiro da carteira do aplicativo.

Na solução proposta por Naila et al. (2020), ao embarcar, o passageiro vê o destino e a tarifa exibidos com um QR Code no dispositivo. O condutor escaneia o código, e o valor é debitado da carteira do passageiro e creditado à do condutor. Uma confirmação da transação é enviada ao passageiro por mensagem de texto, e todas as atividades são registradas no servidor (NAILA et al., 2020). O sistema contribui para a sustentabilidade ao eliminar papel e resolver o problema de troco.

3.3 RFID Based Bus Ticket Generation System

O trabalho de Kaushik e Jain (2021) apresenta um sistema de bilhetes e identificação de passageiros no transporte público, utilizando IoT para solucionar problemas como divergência de tarifas e falta de autenticação. A solução é dividida em três partes principais: emissão de cartões RFID, geração de bilhetes usando esses cartões e verificação de bilhetes.

Na solução de Kaushik e Jain (2021), os cartões RFID podem ser emitidos para passageiros em balcões de pontos de ônibus, onde o gerente do balcão tem

um site no qual ele preencherá o formulário de registro com todos os detalhes do passageiro e emitirá um cartão RFID com uma etiqueta exclusiva. A geração do bilhete é automatizada, e o passageiro, após escanear o cartão, tem o valor da passagem deduzido automaticamente de sua conta. Portas de ônibus são abertas por 240 segundos após a validação do bilhete, mas, em caso de saldo insuficiente, a mensagem “Saldo insuficiente” é exibida (KAUSHIK; JAIN, 2021).

O protótipo do sistema utiliza tecnologias como cartões RFID, Arduino (ATmega328P), LCD para exibição de informações, módulo WiFi (ESP8266) para envio de dados, e servomotores para abertura de portas. O sistema facilita a geração de bilhetes e melhora a segurança e a eficiência do transporte público, além de reduzir o desperdício de papel associado aos bilhetes impressos manualmente (KAUSHIK; JAIN, 2021).

3.4 Análise do sistema de Mobile Payment implementado no transporte público na cidade de São Paulo

O trabalho de Ricardo e Freitas (2017) descreve um sistema para recarga de cartões de transporte público via celular utilizados por usuários do transporte público da cidade de São Paulo, inicialmente disponível para dispositivos com tecnologia NFC. Ao aproximar o cartão do celular, irá sincronizar as informações; o saldo e opções de recarga de crédito são exibidos. O pagamento é feito no próprio celular, que funciona como um posto de recarga portátil.

No trabalho de Ricardo e Freitas (2017), apesar das limitações, como a disponibilidade restrita de NFC, o sistema proporciona conveniência aos usuários e busca modernizar o transporte público. Contudo, a adoção do sistema foi limitada, beneficiando cerca de 2,5 milhões de pessoas em São Paulo, o que representa uma pequena parcela dos 10 milhões de passageiros transportados por dia na cidade. As limitações percebidas na elaboração desse trabalho foram as poucas informações disponíveis e também a falta de acessibilidade aos dados da empresa, ao processo de desenvolvimento e à implantação do projeto

(RICARDO; FREITAS, 2017).

3.5 Correlações entre os trabalhos e a pesquisa

A seguir, apresentamos uma análise dos trabalhos relacionados a este estudo, conforme a Tabela 1. Esta tabela apresenta as seguintes colunas:

- **Tecnologia:** Refere-se ao conjunto de ferramentas, dispositivos e sistemas aplicados para resolver problemas específicos ou melhorar processos. Em sistemas de transporte público, isso inclui soluções como IoT, RFID, e integração com dispositivos móveis, que facilitam o gerenciamento de usuários, pagamentos, rastreamento de veículos, entre outros.
- **Suporte a Rastreamento:** Refere-se às tecnologias e sistemas utilizados para monitorar e determinar a localização precisa de veículos, objetos ou pessoas.
- **Método de recarga:** Refere-se aos mecanismos disponibilizados para que os usuários adicionem crédito aos sistemas de pagamento digital de transporte. Esses métodos podem incluir recarga por aplicativos móveis, terminais físicos de autoatendimento ou integração com plataformas de pagamento online.
- **Notificações:** Verifica se o sistema envia notificações (como SMS ou no próprio aplicativo) para confirmar transações, informar saldo, ou fornecer atualizações sobre o status da conta do usuário.

Tabela 1 – Classificação dos artigos por técnicas.

Artigos	Técnicas			
	Tecnologia	Rastreamento	Método de recarga	Notificações
Trabalho 1	HF RFID	X	Carteira Digital	X
Trabalho 2	QR CODE		Carteira Digital	X
Trabalho 3	HF RFID		Ponto Fisico	
Trabalho 4	NFC ISO/IEC 14443		APP Mobile	X

Fonte: Própria do autor.

A Tabela 1 apresenta uma comparação das funcionalidades abordadas nos estudos revisados, destacando as diferenças entre as soluções de bilhetagem e rastreamento de ônibus. Cada um dos estudos revisados contribui de maneira distinta para o desenvolvimento do sistema proposto, incorporando tecnologias como RFID, NFC, carteira digital, rastreamento via GPS e aplicativos móveis para otimizar a experiência do usuário e melhorar a gestão do transporte público.

No estudo de Shah et al. (2020), observa-se o uso de tecnologias como RFID e monitoramento em tempo real, e com envio de notificações ao usuário pelo aplicativo S-Bus. Naila et al. (2020) diferencia-se por integrar a carteira digital com QR Code e incluir notificações de confirmação da transação por SMS, aprimorando a experiência do usuário. Kaushik e Jain (2021) focam exclusivamente em RFID para bilhetagem, sem oferecer funcionalidades como GPS ou notificações. Já o estudo de Ricardo e Freitas (2017) incorpora NFC para recarga de cartões utilizando o aplicativo *mobile* com diversas funções como a de notificações e visualização de saldo, mas apresenta limitações em termos de alcance de público.

Os estudos revisados desempenham um papel fundamental na construção do trabalho proposto, fornecendo uma base sólida de tecnologias e funcionalidades que foram aprimoradas e combinadas em uma solução integrada. Portanto, a contribuição deste trabalho é integrar essas tecnologias: rastreamento via GPS, recarga de crédito por NFC, notificações em tempo real e sincronização com um sistema RFID e carteira digital, de forma inovadora,

oferecendo uma solução mais completa, acessível e personalizada para os usuários do transporte público.

4 Solução Proposta

Este capítulo apresenta o projeto e o fluxo de execução da solução proposta, que utiliza tecnologias de radiofrequência para o gerenciamento dos cartões de ônibus, o banco de dados Firebase para o armazenamento e a sincronização de informações, e o framework Flutter para o desenvolvimento da interface e principais funcionalidades do sistema, tais como, recarga de crédito nos cartões de acesso por meio de métodos de pagamento online e a consulta da localização dos ônibus via GPS.

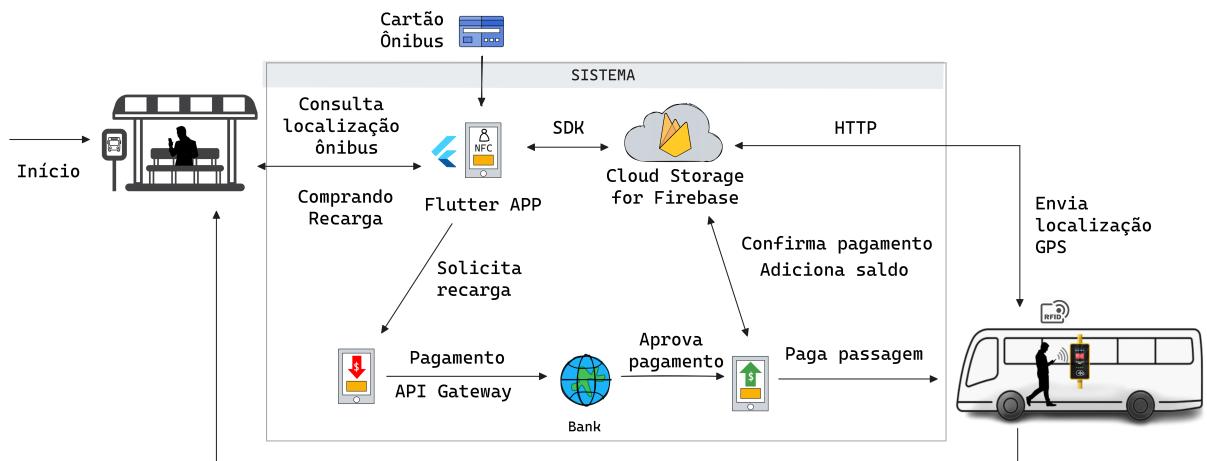
A solução foi desenvolvida com o objetivo de beneficiar os usuários do transporte público de Boa Vista, que atualmente conta com apenas dois pontos físicos de recarga de crédito no cartão espalhados pela cidade, um *chatbot* no *WhatsApp* que realiza recargas exclusivamente via PIX, e não possui um sistema de rastreamento de ônibus. A proposta visa melhorar a experiência dos usuários, oferecendo maior conveniência e acesso aos serviços. O código-fonte da solução está disponível no repositório online do GitHub, acessível pelo link: <<https://github.com/Lucasx10/FlutterAppBus.git>>.

4.1 Arquitetura

A Figura 10 ilustra o ciclo de vida do sistema, onde o usuário pode consultar os créditos do cartão de acesso no aplicativo e por meio da tecnologia NFC, aproximando o cartão no dispositivo de controle de acesso do ônibus. Assim, caso o saldo for insuficiente para a passagem, ele poderá recarregar os créditos no aplicativo, realizando o pagamento on-line do valor desejado. Logo, o sistema atualizará em tempo real o crédito do cartão de acesso do passageiro no banco de dados. Com isso, ele poderá utilizar o ônibus e, ao passar na catraca, o leitor RFID irá debitar o valor da passagem do crédito do cartão e atualizará o valor no banco de dados. Adicionalmente, conforme o ônibus muda a sua

posição nas rotas definidas, a sua respectiva localização (via GPS) é atualizada na nuvem para consulta pelo sistema e seus usuários.

Figura 10 – Ciclo de vida do sistema.



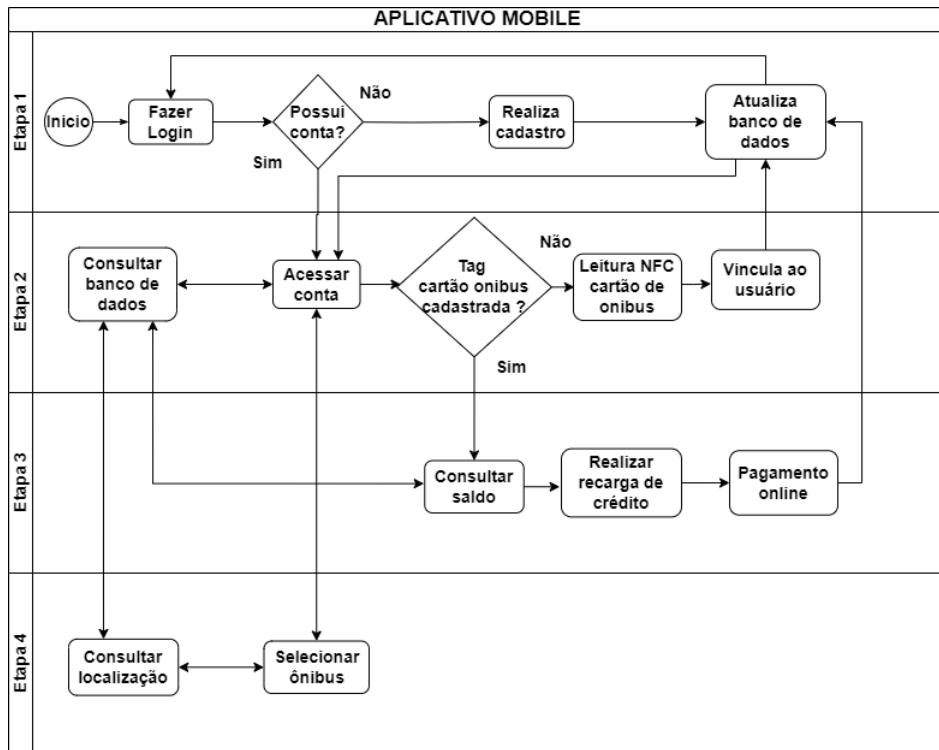
Fonte: Própria do autor

A Figura 11 apresenta o fluxo detalhado da solução, descrito no modelo BPMN (*Business Process Model and Notation*). Este fluxo ilustra as etapas do aplicativo, que são:

- na Etapa 1, onde o usuário pode realizar o login, caso já possua uma conta. Se não tiver uma conta, ele deverá se cadastrar, com os dados sendo registrados no banco de dados, e, em seguida, retornará à tela de login.
- na Etapa 2, após a autenticação, o usuário tem acesso às informações da sua conta, como nome e o cartão registrado, caso tenha um vinculado à sua conta. Caso não possua um cartão registrado, ele poderá realizar o cadastro e vincular o cartão à sua conta por meio da leitura NFC. Esses dados serão atualizados no banco de dados e refletirão as informações do usuário.
- na Etapa 3, o usuário poderá consultar o saldo e o histórico do cartão vinculado à sua conta. Além disso, terá a opção de realizar a recarga de crédito do cartão por meio de pagamento online.

- na Etapa 4, o usuário poderá selecionar o ônibus que deseja rastrear, visualizando sua localização em tempo real, conforme ilustrado no fluxo.

Figura 11 – Diagrama de fluxo do app.



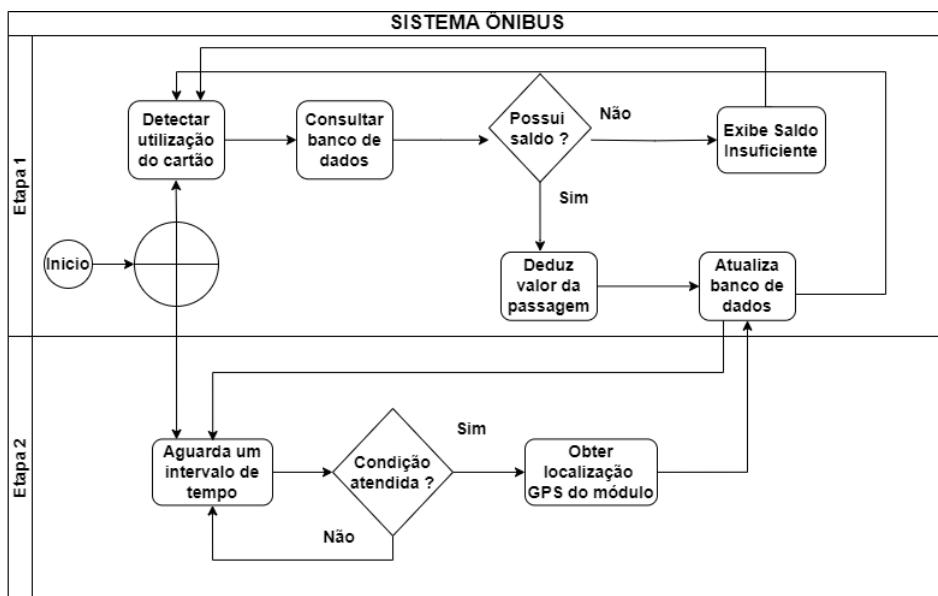
Fonte: Própria do autor

A Figura 12 detalha o fluxo do sistema de localização dos ônibus. Na Etapa 1, o sistema inicia e realiza duas funções simultâneas. Uma delas é monitorar as transações de débito no cartão e, ao detectar a aproximação do cartão, consulta o banco de dados para verificar o saldo. Se houver saldo suficiente, o valor da passagem é deduzido e o saldo atualizado. Caso o saldo seja insuficiente, o sistema informa o usuário.

A outra função simultânea é a coleta dos dados de GPS de um módulo instalado no sistema. Esses dados são utilizados para atualizar a localização do ônibus no banco de dados em um intervalo de tempo que foi definido como a cada 5 segundos, conforme ilustrado na Etapa 2 da Figura 12. Mesmo que o módulo receba uma nova localização durante esse intervalo, o sistema aguarda os 5 segundos para realizar a atualização. Esse processo permite

que os usuários acompanhem a localização dos ônibus de forma precisa e em tempo real, enquanto o sistema continua a capturar dados de GPS e aguarda a aproximação de outros cartões.

Figura 12 – Diagrama de fluxo do sistema do ônibus.



Fonte: Própria do autor

4.2 Ferramentas e Implementações

A solução proposta consiste em um sistema de recarga de créditos para cartões de acesso a ônibus que utilizará um método de pagamento on-line utilizando a API do MercadoPago (v1.0) e a consulta da localização dos veículos por meio de GPS. O sistema é dividido em duas partes: o aplicativo móvel desenvolvido em Flutter e um sistema embarcado baseado em um microcontrolador, neste caso o ESP32, com o código em C++. Este sistema embarcado incluirá um módulo GPS (GY NEO-6MV2), um leitor RFID (RFID RC522) e um módulo GSM GPRS (SIM800L), ambas as partes serão conectadas ao banco de dados Firebase¹.

¹ <https://firebase.google.com/>

4.2.1 Modelagem de Dados

A modelagem de dados para este sistema tem como base o **Firestore** do Firebase, onde os dados são armazenados em documentos que pertencem a coleções. O modelo de dados se organizará em três coleções, cada uma representando uma entidade importante no sistema, que são usuários, cartões e ônibus.

Figura 13 – Modelo de dados Usuários.

Key	Value	Type
▼ usuarios		Query
▼ it8iEvHga4SIMMKTrTr9pSzulR02		Document
cartaoID	1c8ffa35	String
dataCadastro	Dec 31, 2024 2:07:40 PM	Timestamp
email	teste@email.com	String
nome	teste	String

Fonte: Própria do autor

A Figura 13 apresenta a modelagem da coleção de usuários: cada usuário registrado no sistema possui um documento dentro da coleção usuários. Este documento armazena os dados pessoais do usuário, como nome, e-mail e o id (identificador) do cartão vinculado à conta. A autenticação é realizada pelo Firebase Authentication, que gera o uid único de cada usuário; esse uid será o nome do documento do usuário gerado.

Figura 14 – Modelo de dados Cartão.

Key	Value	Type
▼ cartoes		Collection
▼ 06a1b8c3		Document
▼ historico		Collection
► 8D5nRQPMsKCWwdOksi2w	{data: Jan 18, 2025 1:23:05 PM, tipo: Recarga, valor: 5}	Document
► EHLswO59P6F6Q55qjXth	{data: Jan 18, 2025 1:07:57 PM, tipo: Recarga, valor: 5}	Document
dataVinculo	Jan 18, 2025 1:07:43 PM	Timestamp
saldo	14	Double
userId	uEV36vTaSqRloBqwFNQliV39acB3	String

Fonte: Própria do autor

Na coleção cartão (ver Figura 14), cada cartão de ônibus possui um documento identificado pelo número hexadecimal do RFID escaneado durante o cadastro. Esse documento armazena informações essenciais, como saldo disponível, `userId` vinculado, data de associação e um histórico de transações. O histórico é registrado em uma subcoleção chamada `historico_transacoes`, que contém detalhes como data da operação, tipo de recarga e valor adicionado. Dessa forma, todas as recargas e débitos podem ser facilmente rastreados, garantindo mais transparência e controle sobre o uso do cartão.

Figura 15 – Modelo de dados Ônibus.

Key	Value	Type
onibus		Collection
CVUUhPHDtSZrv3WjQWL6		Document
location		Map
lat	2.798164	Double
lng	-60.709054	Double
numero	215	String

Fonte: Própria do autor

Na Figura 15, é apresentada a estrutura da coleção de ônibus, desenvolvida para monitorar a localização dos veículos em tempo real. Cada documento nessa coleção representa um ônibus e contém um campo `location`, do tipo `map`, que armazena a latitude (`lat`) e a longitude (`lng`). De acordo com a localização do ônibus, esses valores são continuamente atualizados pelo sistema de GPS, garantindo um rastreamento preciso e confiável da sua posição.

4.2.2 Aplicativo móvel em Flutter

Para a interface do usuário e interação com o sistema, foi desenvolvido um aplicativo móvel utilizando Flutter, um framework moderno para aplicações multiplataforma. O aplicativo é responsável por gerenciar o acesso dos usuários, a recarga de créditos e a consulta da localização dos ônibus em tempo real. Para armazenar e gerenciar os dados de forma eficiente, foi adotado o

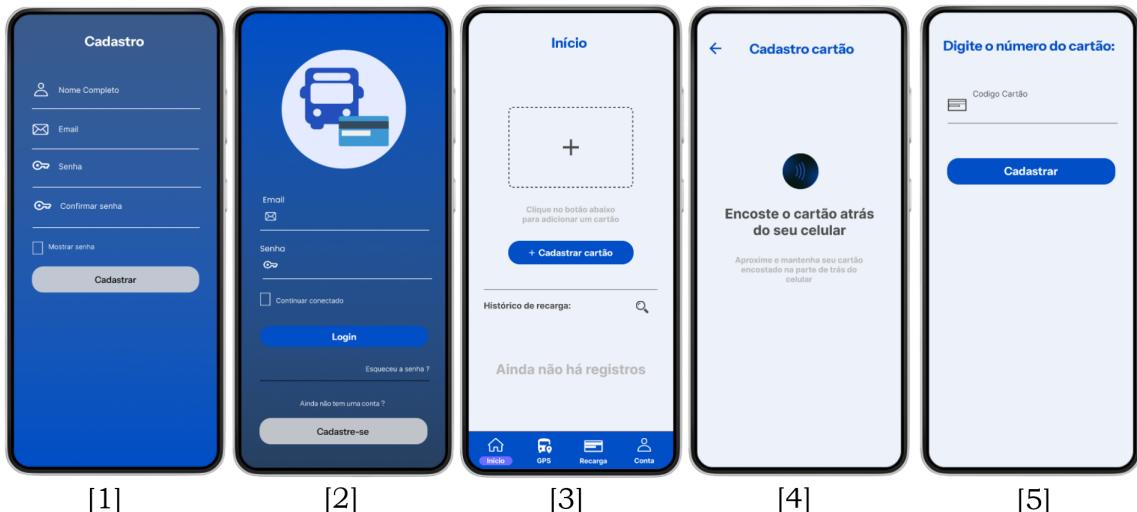
Firestore, que permite a estruturação das informações em coleções e documentos, facilitando a escalabilidade e a sincronização dos dados.

4.2.2.1 Telas de Gerenciamento do Cartão de Acesso

No contexto dos dados definidos, o desenvolvimento do *front-end* do aplicativo, adota o framework Flutter (*v3.24.5*) para criar a interface de usuário. As telas do aplicativo foram projetadas no Figma², levando em consideração os princípios de design essenciais, como a criação de *wireframes* e protótipos (SANDESARA et al., 2022).

A teoria do minimalismo foi aplicada no design das telas, priorizando simplicidade e clareza. A utilização de elementos essenciais e a eliminação de excessos visuais visam reduzir a sobrecarga cognitiva do usuário, proporcionando uma navegação fluida. A escolha das cores foi feita com base na psicologia do usuário, utilizando esquemas cromáticos que facilitam a navegação e promovem conforto visual (SANDESARA et al., 2022).

Figura 16 – Telas de cadastro e login.



Fonte: Própria do autor

A aplicação *mobile* permite que novos usuários se cadastrem inserindo

² <<https://www.figma.com/>>

suas credenciais conforme mostrado na primeira tela da Figura 16. As informações fornecidas são enviadas ao Firebase, que cria e armazena esses dados no Firestore na coleção **usuarios**, de forma segura, utilizando criptografia SSL/TLS para proteger as informações transmitidas entre o aplicativo e seus servidores.

Na segunda tela da Figura 16 é realizado o processo de autenticação utilizando o serviço Firebase Authentication, parte do banco de dados Firebase. A aplicação permite que os usuários façam login utilizando e-mail e senha, quando um usuário realiza o login, o Firebase gera um token de autenticação seguro que é enviado de volta ao cliente. Este token contém informações codificadas sobre a identidade do usuário e é utilizado para manter a sessão ativa. Após a autenticação bem-sucedida, os usuários são redirecionados para a interface principal da aplicação, onde podem realizar o cadastro do cartão de acesso e a consulta da localização dos ônibus, conforme mostra na terceira tela da Figura 16.

Na quarta tela da Figura 16, é realizado o processo de registro de cartões via NFC. O código abaixo implementa esta funcionalidade, utilizando a biblioteca `nfc_manager (v3.5.0)`³:

³ https://pub.dev/packages/nfc_manager

Figura 17 – Código NFC Service

```

1 import 'dart:async';
2 import 'package:nfc_manager/nfc_manager.dart';
3
4 Future<String> scanNfcTag() async {
5     try {
6         bool isAvailable = await checkNfcAvailability();
7         if (!isAvailable) {
8             throw 'O NFC não está disponível no dispositivo.';
9         }
10
11     Completer<String> completer = Completer();
12     NfcManager.instance.startSession(
13         alertMessage: "Aproxime a tag NFC para leitura.",
14         onDiscovered: (NfcTag tag) async {
15             try {
16                 List<int> cardIdBytes =
17                     tag.data['nfca']?[ 'identifier' ]?.toList()
18                     ?? [];
19
20                 String cardIdHex = bytesToHex(cardIdBytes);
21                 completer.complete(cardIdHex);
22             } catch (e) {
23                 completer.completeError('Erro ao processar a
24                     tag NFC.');
25             } finally {
26                 await NfcManager.instance.stopSession();
27             }
28         },
29     );
30
31     return await completer.future;
32 } catch (e) {
33     throw 'Erro no NFC: $e';
34 }
35 }
```

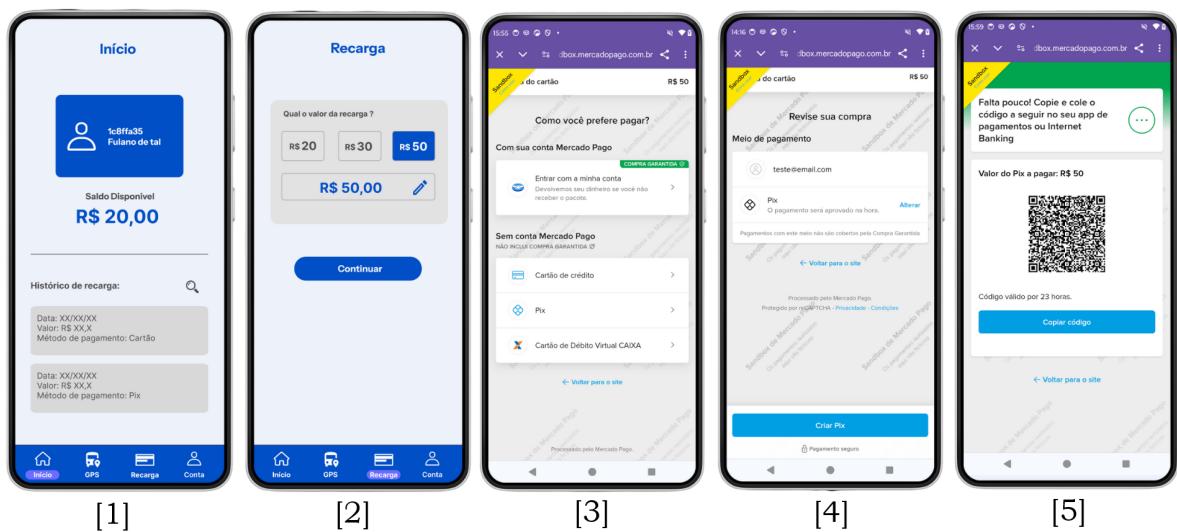
Fonte: Própria do autor.

O código na Figura fornece suporte completo para leitura de tags NFC em dispositivos Android. Ao aproximar o cartão de acesso ao dispositivo do ônibus, o sistema lê os dados da tag NFC. Essa leitura é realizada no método `scanNfcTag()` na linha 4 da Figura 17, que utiliza uma sessão NFC iniciada pela função `startSession()`. O identificador único (ID) do cartão é extraído, convertido para o formato hexadecimal pela função `bytesToHex()` e associado ao UID do usuário logado. Após a leitura bem-sucedida, os dados do cartão são armazenados em uma subcoleção `cartao` no Firestore, dentro do documento do usuário correspondente. Essa estrutura hierárquica garante que os dados

sejam organizados de forma eficiente e facilmente acessíveis.

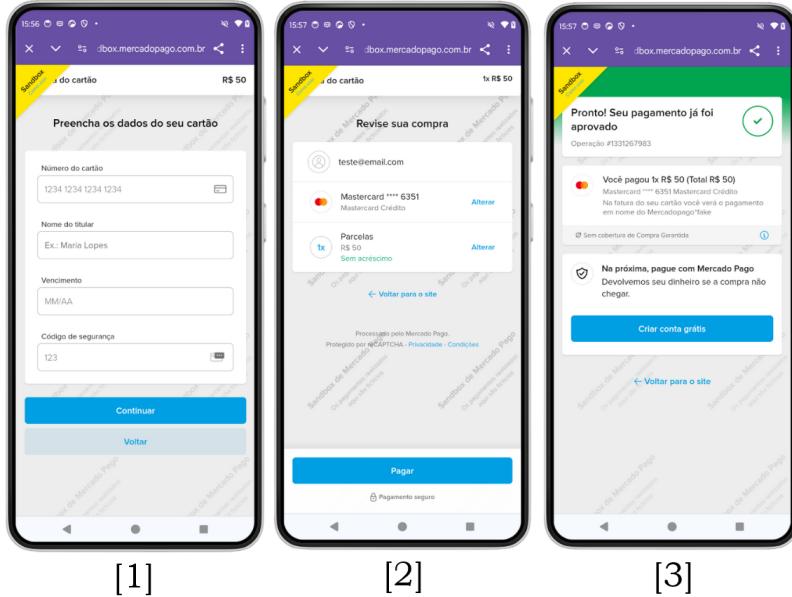
Caso o dispositivo móvel do usuário não possua suporte à tecnologia NFC, ele será direcionado para a tela 5 da Figura 16, onde poderá inserir manualmente o código do cartão. Nesse caso, o sistema realiza uma verificação para identificar se já existe um cartão cadastrado com aquele código. Caso não haja nenhum cartão associado, o código será registrado e vinculado à conta do usuário. A implementação dessa verificação é fundamental para garantir a integridade do banco de dados, prevenindo a duplicação de registros e mantendo a consistência dos dados no Firestore.

Figura 18 – Telas de recarga Pix.



Fonte: Própria do autor

Figura 19 – Telas de recarga cartão de crédito.



Fonte: Própria do autor

Após o cadastro do cartão, o usuário será direcionado à tela inicial, onde poderá visualizar informações como o código do cartão, o saldo disponível e o histórico das transações realizadas. A consulta dos dados do cartão e do histórico é feita diretamente por meio da integração com o Firestore, conforme ilustrado na primeira tela da Figura 18.

Na segunda tela da Figura 18, o usuário terá a opção de inserir o valor desejado para a recarga, além de poder escolher entre alguns valores pré-definidos, facilitando o processo. Assim que o valor for selecionado, o sistema redirecionará o usuário para o navegador, onde será aberta a página de checkout da API do Mercado Pago ⁴.

Na Figura 18, são ilustradas as etapas do processo de pagamento no Mercado Pago. O usuário pode optar entre três métodos: cartão de crédito, PIX ou cartão de débito. As telas 4 e 5 demonstram o pagamento via PIX, no qual o usuário pode escanear um QR Code gerado pela plataforma ou copiar o código PIX e colá-lo diretamente no aplicativo bancário de sua preferência. Já para pagamentos com cartão, conforme mostrado nas telas 1, 2 e 3, na Figura 19,

⁴ <<https://www.mercadopago.com.br/developers/pt/reference>>

o sistema permite que o usuário insira os dados do cartão diretamente na interface do Mercado Pago, revise as informações e conclua a transação de forma segura.

Para garantir um fluxo de pagamento automatizado e eficiente, foi desenvolvido um *webhook* em Node.js responsável por receber as notificações do checkout do Mercado Pago. A escolha do Node.js se baseia em sua arquitetura orientada a eventos e na sua capacidade de gerenciar múltiplas conexões sem bloquear o *thread* principal é um diferencial importante. Além disso, o Node.js é amplamente utilizado e documentado para o desenvolvimento de servidores webhook, o que facilita a implementação e manutenção do sistema. Em comparação, embora o Dart seja uma linguagem eficiente, ele não possui o mesmo nível de adoção e documentação voltada para servidores webhook.

Sempre que ocorre uma atualização no *status* do pagamento, o Mercado Pago envia uma requisição HTTP ao *webhook* contendo informações sobre a transação. Ao receber a notificação, o webhook realiza uma consulta à API do Mercado Pago por meio da rota <<https://api.mercadopago.com/v1/payments/{id}>>, substituindo *id* pelo identificador da transação, para obter detalhes atualizados sobre o pagamento.

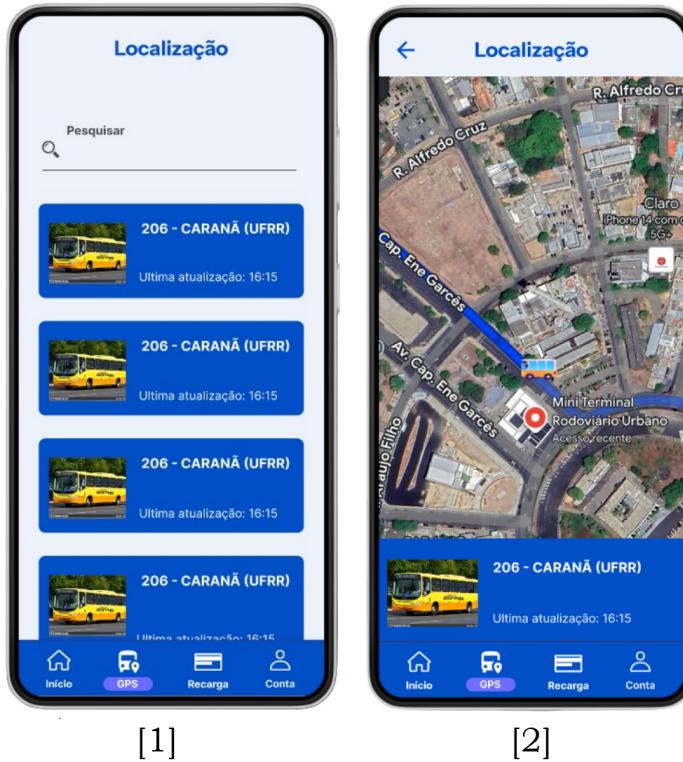
O aplicativo faz uso de *WebSockets* para escutar essas notificações em tempo real. Isso permite que o sistema valide e confirme automaticamente a aprovação do pagamento assim que o *webhook* notifique sobre a alteração de *status*. A API do Mercado Pago é fundamental para a viabilidade do sistema, pois facilita todo esse processo de pagamento, garantindo uma transação segura e eficiente.

4.2.2.2 Telas de Rastreamento do Ônibus

A funcionalidade de rastreamento de localização do ônibus é apresentado nas telas da Figura 20. Na tela 1, o usuário escolhe o número do ônibus desejado para consulta, sendo que cada ônibus possui um número associado às suas

coordenadas de latitude e longitude que são armazenadas no Firestore na coleção de ônibus.

Figura 20 – Telas de gps.



Fonte: Própria do autor

O rastreamento de localização do ônibus utiliza a API do Google Maps⁵ via Flutter, proporcionando uma interface dinâmica e responsiva para o rastreamento em tempo real, além de calcular a distância e o tempo estimado entre o usuário e o veículo. Utilizando as widgets no Flutter a interface do usuário e realiza as chamadas à API para exibir a localização atual do ônibus no mapa, conforme mostra na tela 2 da Figura 20. O aplicativo utiliza a biblioteca google_maps_flutter (*v2.10.0*)⁶. O Firebase Firestore armazenará os dados referentes à localização, incluindo as coordenadas de latitude e longitude, além da rota dos ônibus, garantindo a persistência e a sincronização em tempo real entre os dispositivos dos usuários.

⁵ <<https://developers.google.com/maps/documentation>>

⁶ <https://pub.dev/packages/google_maps_flutter>

A localização do ônibus é constantemente atualizada pelo sistema de GPS instalado no veículo. Esse sistema envia as coordenadas (latitude e longitude) ao Firestore, garantindo sincronização contínua. No lado do usuário, o aplicativo utiliza a funcionalidade de *streaming* de localização, implementada com o pacote Geolocator (*v13.0.2*)⁷, para monitorar mudanças na posição do dispositivo e recalcular a distância e o tempo estimado.

4.2.3 Sistema de Controle de Acesso e Comunicação do Ônibus

O sistema embarcado no ônibus adota uma arquitetura modular que integra hardware e software, centrada no microcontrolador ESP32⁸. Esse componente conecta os módulos GPS (GY-NEO6MV2)⁹ para obter a localização do ônibus, a cada 5 segundos, leitor RFID (RC522)¹⁰ para obter o identificador dos cartões de acesso e GSM (SIM800L)¹¹ para se comunicar com o banco de dados Firebase para gerenciar e atualizar informações sobre os saldos dos cartões dos passageiros. A escolha do ESP32 foi motivada, principalmente, pela sua compatibilidade com diversos pacotes do Firebase, módulo RFID e ser equipado com Wi-Fi e Bluetooth integrados.

O módulo GPS GY-NEO6MV2 é usado para obter a localização do ônibus em tempo real. Esse módulo opera com o sistema de satélites GPS e fornece coordenadas de latitude e longitude com uma precisão horizontal de 2,5 metros no modo autônomo e 2,0 metros com SBAS (Satellite-Based Augmentation System), conforme especificado em seu datasheet. No entanto, essa precisão é garantida apenas sob boas condições de sinal, podendo ser afetada por interferências, obstruções físicas e reflexões do sinal.

O leitor RFID RC522 é utilizado para identificar as *tags Mifare* presentes

⁷ <https://pub.dev/packages/geolocator>

⁸ https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

⁹ <https://www.datasheethub.com/wp-content/uploads/2022/08/NEO6MV2-GPS-Module-Datasheet.pdf>

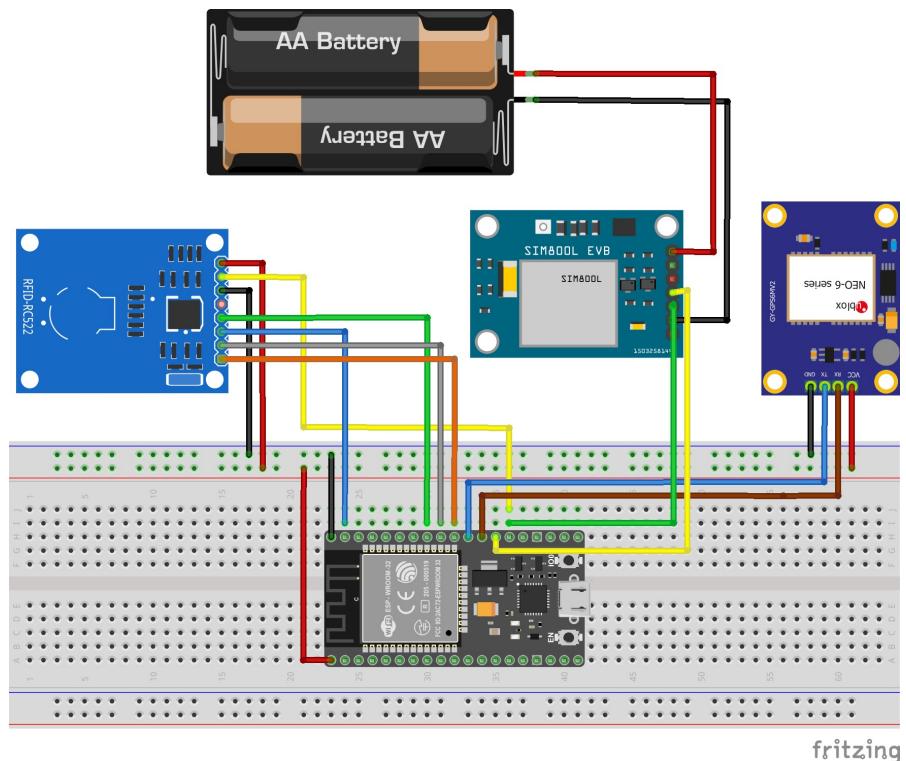
¹⁰ <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>

¹¹ <https://www.alldatasheet.com/datasheet-pdf/pdf/1741389/SIMCOM/SIM800L.html>

nos cartões de acesso dos passageiros. Operando na frequência de 13,56 MHz, o módulo utiliza a interface SPI para se comunicar com o ESP32. Ao detectar um cartão, o sistema obtém o saldo associado ao mesmo, acessando os dados no banco de dados Firebase. Caso o saldo seja suficiente para o débito da tarifa, o valor é deduzido automaticamente, e o saldo atualizado é enviado de volta ao Firebase. Em casos de saldo insuficiente, o sistema emite uma mensagem de aviso.

O banco de dados Firebase desempenha um papel central no sistema, armazenando informações de localização, saldos dos cartões RFID e histórico de transações. A biblioteca Firebase ESP Client (*v4.4.16*) foi utilizada para estabelecer a comunicação entre o ESP32 e o Firebase, suportando operações como leitura e escrita de dados em tempo real. A manipulação de documentos JSON, estruturados para otimizar o armazenamento e a recuperação de informações, é realizada pela biblioteca `FirebaseJson`. Essa estrutura assegura que o sistema seja escalável e possa ser adaptado a novos requisitos no futuro.

Figura 21 – Esquema de conexões do sistema do ônibus.



Fonte: Própria do autor

A comunicação entre o ESP32, o módulo GPS e o leitor RFID é feita visando garantir que as informações sejam enviadas corretamente para o Firebase. Conforme mostrado na Figura 21, a conexão SPI entre o leitor RFID e o ESP32 é configurada da seguinte forma:

- 3.3V: conectado ao pino 3.3V do ESP32 para alimentar o leitor RFID.
- GND: conectado ao pino GND do ESP32 para completar o circuito.
- SCK (*Serial Clock*): conectado ao pino *D18* do ESP32, utilizado para transmitir o sinal de clock.
- MISO (*Master In Slave Out*): conectado ao pino *D19* do ESP32, para receber os dados do leitor RFID.
- MOSI (*Master Out Slave In*): conectado ao pino *D23* do ESP32, para enviar os dados do ESP32 ao leitor RFID.
- SS (*Slave Select*): conectado ao pino *D5* do ESP32 para selecionar o leitor RFID.
- RST (*Reset*): conectado ao pino *D2* do ESP32, utilizado para resetar o leitor RFID.

O módulo GPS GY NEO-6MV2 é conectado ao ESP32 através da interface UART (Universal Asynchronous Receiver/Transmitter). Os pinos TX e RX são responsáveis pela troca de dados entre o GPS e o microcontrolador. O pino TX do GPS é conectado ao pino RX do ESP32 (*RX2*), e o pino RX do GPS é conectado ao pino TX do ESP32 (*TX2*). Dessa forma, as coordenadas de latitude e longitude são continuamente transmitidas pelo módulo GPS ao ESP32 no formato NMEA (National Marine Electronics Association), uma sequência padronizada de mensagens que contém as informações de posicionamento.

Para interpretar os dados brutos fornecidos pelo GPS e extrair informações relevantes, como a posição atual do veículo, é empregada a biblioteca TinyGPS++ (v1.0.3). Essa biblioteca processa as mensagens NMEA e converte

os dados para um formato utilizável pelo ESP32. As informações de localização são então armazenadas em variáveis e formatadas em um objeto JSON para envio ao Firebase Firestore.

Os dados são transmitidos para o Firebase em intervalos de 5 segundos, ao invés de a cada atualização recebida do GPS. Esse recurso é essencial para garantir o monitoramento eficiente da posição do ônibus ao longo do trajeto, permitindo que passageiros consultem a localização em tempo real via aplicativo.

Para viabilizar a transmissão dos dados para a nuvem, é utilizado o módulo GSM SIM800L. Esse módulo opera com uma voltagem de 3,4V a 4,4V e é alimentado por uma bateria de íons de lítio. Ele se conecta ao ESP32 através dos pinos TXD (*D4*) e RXD (*D2*), permitindo o envio das coordenadas via conexão de dados móveis do cartão SIM. Assim, a cada 5 segundos, o ESP32 coleta as coordenadas do GPS e as transmite para o Firebase, garantindo um fluxo contínuo e eficiente de informações para o sistema de monitoramento.

5 Avaliação Experimental

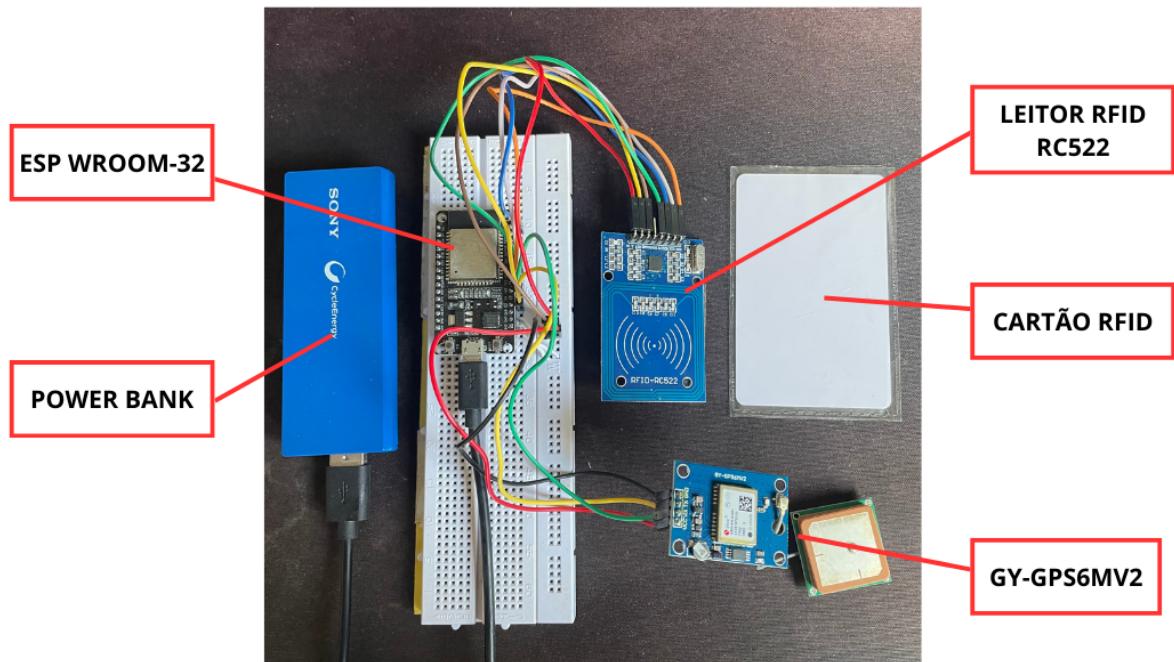
Este capítulo descreve o planejamento, execução e análise dos resultados referentes a solução proposta. Assim, visando avaliar de forma experimental o aplicativo móvel e o sistema computacional desenvolvidos a partir do método proposto, focando no cadastro e recarga de crédito do cartão RFID de ônibus, bem como na localização de ônibus por GPS.

5.1 Projeto da Avaliação Experimental

Esta seção detalha o planejamento e a concepção para a execução da avaliação das funcionalidades do sistema desenvolvido, com o objetivo de testar a eficácia do método proposto. A Figura 22 apresenta o protótipo do sistema embarcado para controle de acesso e rastreamento no ônibus. Esse sistema é composto por um ESP32, um leitor RFID RC522, responsável pela leitura dos cartões, e um módulo GPS Neo-6M, que fornece a localização do ônibus a cada 5 segundos. Todos os componentes estão interconectados por jumpers, conectando-os ao ESP32. Devido a problemas relacionados à lógica do sistema e à disponibilidade do módulo eSIM, a conexão à internet foi estabelecida utilizando o roteamento da rede móvel de um smartphone. A alimentação dos componentes é realizada por meio de um power bank Sony (bateria 3000mAh).

O aplicativo móvel, denominado **CardBus**, possui 56 MB na versão 1.0.0. O APK foi desenvolvido para versões do Android 6.0 ou superior e foi executado em um *smartphone* Motorola Edge 40 NEO 5G (CPU MediaTek Dimensity 7030 2.1 GHz, 8GB RAM, Android 14).

Figura 22 – Protótipo do sistema do ônibus.



Fonte: Própria do autor

A avaliação experimental desta solução tem como principal objetivo validar o funcionamento adequado das funcionalidades, além de contribuir para a melhoria da usabilidade no contexto do transporte público. A avaliação busca responder às seguintes questões de pesquisa (QP):

QP1 : O aplicativo móvel é capaz de realizar o gerenciamento da conta e créditos do cartão de acesso do usuário de forma precisa e eficiente, garantindo que todas as funcionalidades essenciais sejam executadas?

QP2 : A localização do ônibus, para o rastreamento pelo usuário, na base de dados são refletidas com precisão no aplicativo móvel, garantindo ao usuário uma visualização da posição do veículo durante a sua respectiva rota?

QP3 : O sistema proposto é capaz de gerenciar os créditos de usuário e atualização de localização sem comprometer a integridade dos dados?

Visando responder às questões de pesquisa, a execução deste experimento foi dividida em duas partes. Na **primeira parte**, foram implementados testes de validação de entrada utilizando o Strategy Pattern, garantindo flexibilidade na validação da entrada dos dados. Expressões regulares (regex) foram aplicadas para validar os campos de entradas, tais como, de e-mail e senha, assegurando que as entradas estivessem conforme os formatos esperados. A utilização do Strategy Pattern, conforme discutido por Gamma et al. (1995), permite encapsular diferentes algoritmos de validação, tornando-os intercambiáveis e garantindo que a lógica de validação seja independente do código que a utiliza. Além disso, foram realizados testes de unidade, cobrindo diferentes cenários de validação para cada campo e garantindo a qualidade do código.

Na **segunda parte**, foram simulados 2 cenários de uso prático da solução proposta, o primeiro foi o cadastro de usuários e dos cartões de acesso juntamente com a recarga dos mesmos, visando avaliar as funcionalidades do aplicativo. Por fim, foi realizada a simulação do sistema de rastreamento de ônibus, com a atualização da localização do veículo a cada 5 segundos. O desempenho do sistema foi validado em condições reais de uso (via simulação de rota de ônibus já conhecidas na cidade de Boa Vista-RR), incluindo a interação dos passageiros com o sistema de leitura de cartões RFID e a verificação da funcionalidade do rastreamento via GPS.

5.2 Execução e Resultado da Avaliação Experimental

Durante o desenvolvimento do aplicativo móvel, a execução do experimento foi cuidadosamente planejada para garantir a segurança e a qualidade do processo de cadastro e login dos usuários. Para a validação de entrada, a classe **EmailValidator**, como exemplificado no código da Figura 23, foi projetada para ser flexível e reutilizável, permitindo que diferentes cenários de teste fossem facilmente integrados. Este é um exemplo claro de como o Strategy Pattern foi aplicado, conforme descrito por Gamma et al. (1995), para desacoplar os

algoritmos de validação da lógica principal da implementação, resultando em um código mais modular e manutenível. A adoção desse padrão não só simplificou os testes, mas também garantiu que o código fosse facilmente adaptável a novos requisitos, como a adição de novos métodos de validação.

Figura 23 – Class EmailValidator

```

1 class EmailValidator {
2     String? validate({String? email}) {
3         if (email == null || email.isEmpty) {
4             return 'O e-mail é obrigatório';
5         }
6
7         // Expressão regular para validar o formato do e-mail
8         final emailRegex = RegExp(
9             r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$',
10            caseSensitive: false, // Torna a regex case-
11            insensitive
12        );
13
14         if (!emailRegex.hasMatch(email)) {
15             return 'O e-mail é inválido';
16         }
17
18         return null;
19     }
}

```

Fonte: Própria do autor.

Para assegurar que as entradas fornecidas pelos usuários estivessem no formato adequado, foram empregadas expressões regulares (regex) nos campos de e-mail e senha, que são essenciais para o processo de cadastro e login. As expressões regulares foram desenvolvidas seguindo as melhores práticas recomendadas, conforme descrito por Groner e Manricks (2015), que apresenta padrões amplamente reconhecidos e eficientes. Um exemplo de regex utilizada para validar o campo de e-mail verifica se o endereço contém caracteres válidos, como letras e números, seguidos de um domínio válido e sua extensão, conforme exemplificado na linha 9 do código da Figura 23.

Para a validação da senha do usuário, uma abordagem similar foi utilizada, garantindo que a senha atenda a certos requisitos, como conter pelo menos uma letra minúscula, uma letra maiúscula, um número, um caractere

especial e tenha no mínimo 6 caracteres de comprimento. A regex usada é apresenta da seguinte forma:

```
^ (?=.*[a-z]) (?=.*[A-Z]) (?=.*[0-9]) (?=.*[^a-zA-Z0-9]).{6,} $
```

Visando garantir a integridade e o funcionamento adequado dessas validações, foram realizados testes de unidade no aplicativo móvel, usando a linguagem de programação Dart utilizada no desenvolvimento do aplicativo. Ao todo, foram realizados 27 testes para validar as funcionalidades do sistema, sendo eles distribuídos da seguinte forma:

- **8 testes no serviço Firebase:** Esses testes utilizaram dados fictícios do banco, verificando a correta inserção e leitura de dados no banco de dados Firebase.
- **4 testes de e-mail:** Foram verificadas as seguintes condições: e-mail nulo, e-mail vazio, e-mail inválido e e-mail válido. Cada cenário foi testado para garantir que o sistema tratasse corretamente as entradas.
- **11 testes de senha:** Foram avaliadas as seguintes condições: senha nula, senha vazia, senha com menos de 6 caracteres, senha sem letra minúscula, senha sem letra maiúscula, senha sem caractere especial e confirmação de senha. Cada um desses cenários foi executado para garantir que o sistema validasse corretamente as entradas.
- **4 testes no campo de valor de recarga:** Foram testados os seguintes cenários: valor vazio, valor não numérico, valor menor ou igual a 0 e valor válido, assegurando que o sistema tratasse essas condições de forma adequada.

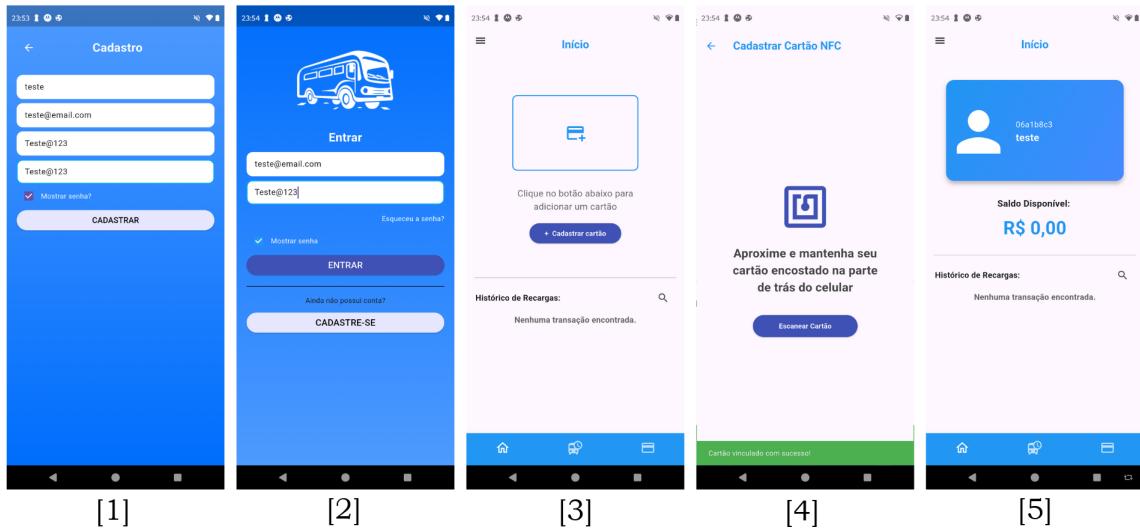
Esses testes foram executados utilizando o *Flutter Test* em Dart, garantindo que todas as validações e funcionalidades do sistema fossem corretamente verificadas. O framework de testes *Flutter Test* foi utilizado para automatizar e garantir a cobertura completa de todos os casos de teste.

Para a execução da avaliação experimental da solução proposta, dois cenários simulados (conforme apresentados nas seções a seguir) foram definidos com o objetivo de testar as funcionalidades do sistema no contexto da Universidade Federal de Roraima (UFRR). Os casos de uso foram projetados para simular cadastro de usuários no aplicativo e a rota de ônibus e o rastreamento atualizado a cada 5 segundos, avaliando a precisão e a eficácia do sistema em situações reais.

5.2.1 Caso de Uso 1: Cadastro de usuários e cartões RFID no sistema

Neste caso de uso, o teste foi conduzido com o cadastro de três usuários, identificados como **teste1**, **teste2** e **teste3**, juntamente com três cartões de ônibus RFID distintos no sistema. O primeiro cadastro seguiu o fluxo padrão do aplicativo, conforme ilustrado na Figura 24. Inicialmente, o usuário inseriu suas credenciais na tela de cadastro (tela 1) e, em seguida, efetuou o login (tela 2). Após o acesso, realizou o cadastro do cartão utilizando a tecnologia NFC (tela 3). Ao aproximar o cartão RFID, o sistema efetuou a leitura, registrou o cartão e vinculou-o à conta do usuário. Como resultado, exibiu a mensagem de "Cartão vinculado com sucesso" (tela 4) e, posteriormente, apresentou ao usuário o cartão cadastrado e o saldo disponível corretamente (tela 5).

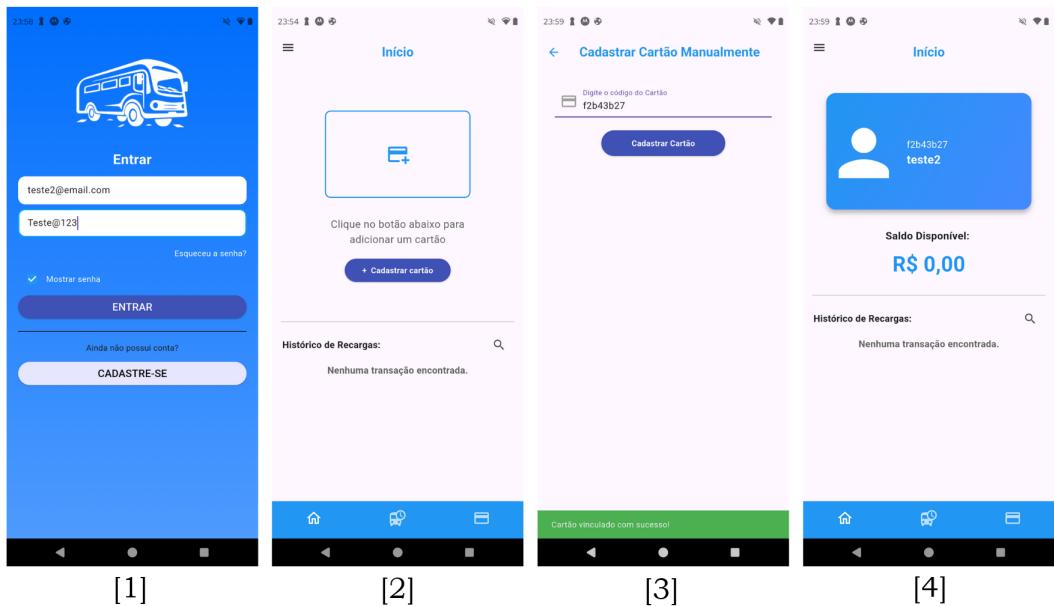
Figura 24 – Caso de uso 1 cadastro NFC.



Fonte: Própria do autor

O segundo cadastro do usuário seguiu o mesmo fluxo do primeiro, porém, na parte do cadastro do cartão foi realizado manualmente, para testar o procedimento em dispositivos sem suporte NFC. Após o cadastro e o login, O usuário digitou o número hexadecimal do cartão e o sistema registrou e vinculou o cartão ao usuário de maneira eficiente, conforme mostra na tela 3 da Figura 25.

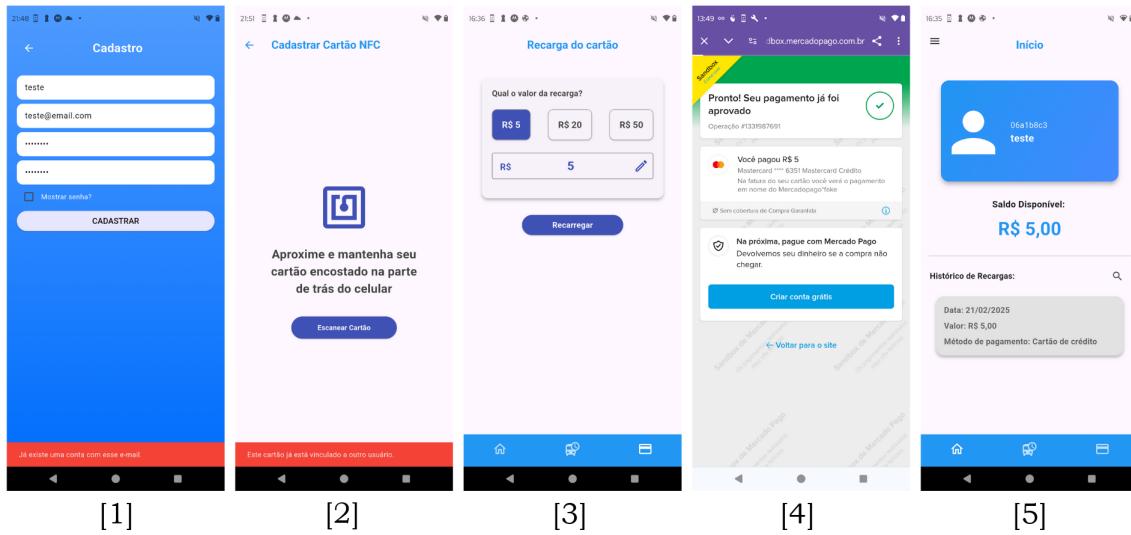
Figura 25 – Caso de uso 1 cadastro S/ NFC.



Fonte: Própria do autor

No terceiro cadastro, foi simulada a tentativa de registrar um usuário já cadastrado e um cartão já cadastrado. Em ambos os casos, o sistema impediu o cadastro, retornando uma mensagem informando que o cartão já estava associado a outro usuário, conforme detalhado na tela 1 e 2 da Figura26. Por fim, um cartão diferente foi utilizado para realizar o cadastro de um novo usuário e cartão, que foi registrado com sucesso no sistema. Em seguida, foi realizada uma recarga no aplicativo de R\$ 5,00 no cartão de transporte utilizando o método de pagamento via cartão de crédito no usuário **teste1**.

Figura 26 – Caso de uso 1 cadastro inválido.



Fonte: Própria do autor

Após a execução do caso de uso, os seguintes resultados foram obtidos: o aplicativo conseguiu cadastrar usuários **teste1**, **teste2** e **teste3** e cartões sem erros, armazenando corretamente todas as informações de autenticação no **Firebase Authentication** e os dados do cartão cadastrado vinculado ao usuário no **Firestore Database**, conforme ilustrado na Figura 27.

Figura 27 – Resultado do caso de uso 1.

Pesquise por endereço de e-mail, número de telefone ou UID do usuário					Adicionar usuário	Mais no Google Cloud
Identificador	Provedores	Data de criação	Último login	UID do usuário		
teste3@email.com	✉	19 de fev. de 2025	19 de fev. de 2025	4lzaJBjVjXutM09WBlyMWUz...		
teste2@email.com	✉	19 de fev. de 2025	19 de fev. de 2025	33wBzyeZ59RoVSfUkB99p05Bubw...		
teste@email.co	✉	19 de fev. de 2025	19 de fev. de 2025	VGzqt2mewGTHA1fQzb47vM1mGCC...		

(default)	usuarios	= :	VGzqt2mewGTHA1fQzb47vM1mGCC2
+ Iniciar coleção	+ Adicionar documento	+ Iniciar coleção	
cartoes	33wBzyeZ59RoVSfUkB99p05Bubw1	+ Adicionar campo	
onibus	4lzaJBjVjXutM09WBlyMWUzbyk1	cartaoID: "06a1b8c3"	
usuarios	VGzqt2mewGTHA1fQzb47vM1mGCC2	dataCadastro: 19 de fevereiro de 2025 às 22:34:10 UTC-4	
		email: "teste@email.co"	
		nome: "teste"	

Fonte: Própria do autor

Durante os testes, verificou-se que, após o cadastro do cartão na tela 4 da Figura 24 ao passar automaticamente para a tela 5, era necessário atualizar manualmente(arrastando a tela para baixo) a tela para exibir os dados recém-cadastrados. No entanto, o saldo do cartão foi atualizado em tempo real sempre que um valor era debitado ou uma recarga realizada. Além disso, o pagamento da recarga, realizado por meio de um cartão de teste do Mercado Pago, foi processado com sucesso. O servidor *webhook* confirmou corretamente a transação e retornou a validação ao aplicativo, comprovando que o sistema reconhece e autentica adequadamente a compra de créditos.

5.2.2 Caso de Uso 2: Simulação de Rastreamento de Ônibus na UFRR

Neste experimento, três participantes estiveram envolvidos: o motorista do veículo, o passageiro no banco da frente e um terceiro usuário, que aguardava no ponto final da rota simulada. O objetivo foi testar o rastreamento do transporte público dentro do campus da Universidade Federal de Roraima (UFRR), utilizando um carro como substituto do ônibus.

Conforme ilustrado na Figura 28 **A** representa o ponto de ônibus ao lado do Mini Terminal Luiz Canuto Chaves, na Av. Cap. Ene Garcês, no centro da cidade, local onde o motorista foi posicionado para iniciar o experimento. A Figura 28 **B** mostra o ponto de ônibus em frente ao Centro de Ciências da Saúde (CCS) da UFRR, na Avenida Nova Iorque, representando o destino, onde o terceiro usuário aguardava a chegada do transporte. Por fim, a Figura 28 **C** ilustra a rota percorrida, baseada no trajeto da linha de ônibus 308 - Cauamé/UFRR, conforme indicado pelo aplicativo *Bus2*¹, que serviu como referência para a realização do experimento.

¹ <https://bus2.info/rr/boavista>

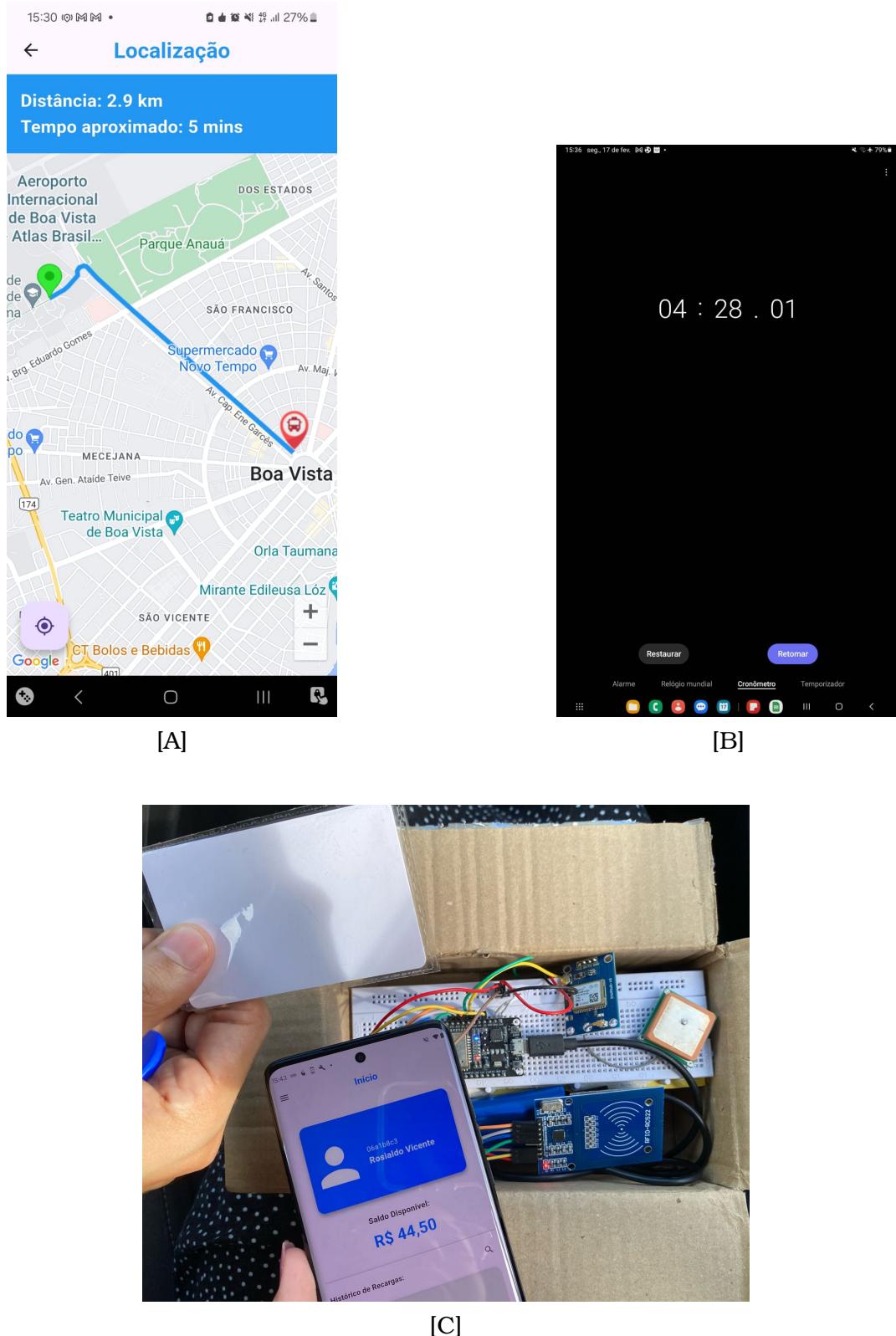
Figura 28 – Rota do experimento



Fonte: Própria do autor

Durante o percurso, o passageiro aproximou 3 cartões RFID diferentes para simular a utilização do transporte público por passageiros conforme mostra na Figura 29 C. Além disso, monitorou em tempo real a dedução do saldo de um dos cartões no aplicativo, verificando o correto funcionamento do sistema de pagamento eletrônico.

Figura 29 – Resultado do caso de uso 2



Fonte: Própria do autor

A rota sugerida pelo aplicativo móvel coincidiu com a rota padrão do

ônibus 308, utilizado como base de comparação. No entanto, o aplicativo não garante sempre a rota padrão, pois não possui acesso às rotas oficiais dos ônibus da prefeitura de Boa Vista. Para maior precisão, seria necessário importar dados no formato KML (Keyhole Markup Language) para cada ônibus, definindo a rota como fixa em vez de dinâmica. Durante os testes, observou-se que:

- a medida que o veículo avançava até o ponto final, o aplicativo atualizava sua posição no mapa e recalculava o tempo e a distância estimados a cada 5 segundos. Esse processo permitiu que o usuário no ponto final acompanhasse, em tempo real, o deslocamento do ônibus via rastreamento por GPS, conforme ilustrado na Figura 29 **A**;
- o tempo de espera foi cronometrado para avaliar a precisão da estimativa do aplicativo. Nesse teste, o tempo previsto para a chegada do ônibus foi de 5 minutos, enquanto o tempo real registrado foi de 4 minutos e 28 segundos, conforme demonstrado na Figura 29 **B**; e
- o passageiro da simulação conseguiu acompanhar, sem falhas, a atualização em tempo real do saldo do cartão de transporte no aplicativo. Ao aproximar o cartão do sistema do ônibus, que inicialmente possuía um saldo de R\$ 50,00, o aplicativo registrou corretamente a dedução da tarifa de R\$ 5,50, atualizando o saldo para R\$ 44,50, conforme ilustrado na Figura 29 **C**.

6 Considerações Finais e Trabalhos Futuros

O presente trabalho apresentou um sistema computacional para a recarga de créditos e rastreamento de ônibus utilizando tecnologia NFC e GPS, com suporte a comunicação em nuvem via Firebase. Os resultados da avaliação experimental realizada permitiram verificar a viabilidade e eficiência do sistema proposto, validando as funcionalidades principais, como o cadastro de usuários, recarga de créditos e rastreamento de transporte público.

Os experimentos realizados demonstraram que o aplicativo móvel e o sistema embarcado (baseado no ESP32) que compõem o sistema proposto funcionam de maneira integrada, garantindo a correta leitura e gestão de crédito dos cartões RFID, além da atualização dos dados no Firebase, proporcionando aos passageiros informações sobre o cartão cadastrado e a localização do veículo. Além disso, os testes de unidade, os testes de validação e a implementação do servidor webhook para confirmação de pagamentos reforçaram a segurança do sistema.

Observa-se que no sistema ainda existem pontos de melhorias para que se torne um produto final, por outro lado, a proposta tem pontos positivos e promissores. Um dos pontos a ser melhorado está no rastreamento dos ônibus, uma vez que, na versão atual, a rota gerada nem sempre corresponde exatamente ao trajeto real do veículo, podendo resultar em cálculos de tempo e distância diferentes do real. Como trabalho futuro, pretende-se realizar um estudo detalhado das rotas dos ônibus em Boa Vista–RR, cadastrando essas informações no banco de dados para aprimorar a precisão das estimativas de tempo e percurso.

Adicionalmente, busca-se explorar novas tecnologias para aprimorar a precisão da localização dos ônibus, como a utilização da tecnologia LoRaWAN. Essa abordagem permitiria ampliar a cobertura do rastreamento, especialmente

em áreas com pouca ou nenhuma conectividade com a internet, garantindo maior estabilidade na obtenção da localização dos ônibus.

Referências

- ALLIANCE, L. **LoRaWAN: Low Power, Wide Area Network.** 2022. Acessado em: 12 Set. 2024. Disponível em: <<https://lora-alliance.org/about-lorawan/>>.
- BHAGAT, S. A. et al. Review on mobile application development based on flutter platform. **International journal for research in applied science and engineering technology**, v. 10, n. 1, p. 803–809, 2022. Acessado em: 11 Set. 2024. Disponível em: <<https://doi.org/10.22214/ijraset.2022.39920>>.
- COCOLETE, D.; BORGES, J. H.; FLORIAN, F. Estudo comparativo entre a eficiÊncia de diferentes bancos de dados relacionais (mysql e postgresql) e nÂo relacionais (mongodb). **Revista ft**, v. 28, p. 13–14, 10 2024.
- COSKUN, V.; OZDENIZCI, B.; OK, K. A survey on near field communication (nfc) technology. **Wireless personal communications**, v. 71, n. 3, p. 2259–2294, 2013. Acessado em: 02 Out. 2024.
- COSTA, D. B. da. **Sistemas Digitais**. 1. ed. Porto Alegre: SAGAH, 2018. v. 1. 144 p.
- ESPRESSIF. **ESP32 DevKitC V4 Getting Started Guide**. 2023. Disponível em: <<https://docs.espressif.com/projects/esp-idf/en/v3.1.7/get-started/get-started-devkitc.html>>.
- FIREBASE. **Firebase Data Model**. 2024. Acessado em: 12 Set. 2024. Disponível em: <<https://firebase.google.com/docs/firestore/data-model?hl=pt-br>>.
- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Reading, MA: Addison-Wesley, 1995. ISBN 0-201-63361-2.
- GRONER, L.; MANRICKS, G. **JavaScript Regular Expressions**. [S.l.]: Packt Publishing Ltd, 2015. Acessado em: 07 Fev. 2025.
- GÜLCÜOĞLU, E.; USTUN, A.; SEYHAN, N. Comparison of flutter and react native platforms. **Journal of Internet Applications and Management**, 12 2021.
- IBRAHIM, D. **The Complete ESP32 Projects Guide: 59 Experiments with Arduino IDE and Python**. Elektor, 2019. ISBN 9781907920752. Disponível em: <<https://books.google.com.br/books?id=2VypwwEACAAJ>>.
- JABBAR, W. A. Development of lorawan-based iot system for water quality monitoring in rural areas. **Expert Systems with Applications**, v. 242, 2024. Acessado em: 12 Set. 2024.
- JAPARA, E. M.; ARIFIN, S.; IRWANSYAH, E. Geolocation system module creation to validate user location coordinates in an android application using flutter framework. **Journal of Computer Science**, v. 19, n. 8, p. 1050–1064, 2023. Acessado em: 03 Out. 2024.

- KAUSHIK, A.; JAIN, N. Rfid based bus ticket generation system. In: **2021 International Conference on Technological Advancements and Innovations (ICTAI)**. [S.I.]: IEEE, 2021. Acessado em: 27 Jun. 2024.
- KOLBAN, N. Kolban's book on esp32, september 2018. **Texas, USA**, 2018.
- MARTINS, N. A. **Sistemas Microcontrolados**. Brasil: Novatec Editora, 2005. ISBN 978-8575220740.
- MYSQL. **Creating EER Diagrams**. 2024. Acessado em: 12 Set. 2024. Disponível em: <<https://dev.mysql.com/doc/workbench/en/wb-creating-eer-diagram.html>>.
- NAILA; AJITH; MURSHID, M. Smart bus ticketing system. **IJISET-International Research Journal of Engineering & Technology**, v. 7, n. 6, 2020. Acessado em: 27 Jun. 2024. Disponível em: <<https://www.irjet.net/archives/V7/I6/IRJET-V7I6182.pdf>>.
- OLIVEIRA, B. B. Aplicabilidade dos microcontroladores em inovações tecnológicas. In: **VII CONNEPI-Congresso Norte Nordeste de Pesquisa e Inovação**. [S.I.: s.n.], 2012.
- RICARDO, S. C.; FREITAS, H. M. R. d. Análise do sistema de mobile payment implementado no transporte público na cidade de são paulo. **Revista Gestão & Tecnologia**, v. 17, n. 1, p. 258–276, 2017. Acessado em: 14 set. 2024. Disponível em: <<https://revistagt.fpl.emnuvens.com.br/get/article/view/923>>.
- SANDESARA, M.; BODKHE, U.; TANWAR, S.; ALSHEHRI, M. D.; SHARMA, R.; NEAGU, B.-C.; GRIGORAS, G.; RABOACA, M. S. Design and experience of mobile applications: A pilot survey. **Mathematics**, v. 10, n. 14, p. 2380, 2022. Academic Editor: Jong-Min Kim, Received: 19 May 2022, Accepted: 5 July 2022, Published: 6 July 2022. Disponível em: <<https://doi.org/10.3390/math10142380>>.
- SHAH, M. J.; PRASAD, R. P.; SINGH, A. S. Iot based smart bus system. In: **2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)**. [S.I.]: IEEE, 2020. p. 130–134. Acessado em: 27 Jun. 2024.
- SILVA, R. O. da; ARAUJO, W. M.; CAVALCANTE, M. M. Visão geral sobre microcontroladores e prototipagem com arduino. **TECNOLOGIAS EM PROJEÇÃO**, v. 10, n. 1, p. 36–46, ago. 2019. Disponível em: <<https://projecaociencia.com.br/index.php/Projecao4/article/view/1357>>.
- SUDIARTHA, I. K. G.; INDRAYANA, I. N. E.; SUASNAWA, I. W. et al. Data structure comparison between mysql relational database and firebase database nosql on mobile based tourist tracking application. **Journal of physics. Conference series**, v. 1569, p. 032092, 2020. Acessado em: 11 Set. 2024.
- WANG, Y. **Review and testing of plugins in Flutter for Android and IOS**. Tese (Dissertação de Mestrado) — Politecnico di Torino, Torino, 2022. Acessado em: 03 Out. 2024. Disponível em: <<https://webthesis.biblio.polito.it/secure/22858/1/tesi.pdf>>.

WEBER, ; CANTARELLI, G. **Agendei: proposta de desenvolvimento de uma aplicação móvel para realização de oferta de serviços e controle de agendamentos online.** 2020. Trabalho de Conclusão de Curso Sistemas De Informação - Universidade Franciscana (UFN), p. 28. Acessado em: 14 Set. 2024. Disponível em: <https://tfgonline.lapinf.ufn.edu.br/media/midias/TFG_2_-_EricoWeber.pdf>.

WU, W. **React Native vs Flutter, cross-platform mobile application frameworks.** 2018. Accessed: 20-Jul-2019. Disponível em: <<https://www.theses.usf.edu/handle/10024/146232>>.

ZANOTTA, D. C.; CAPPELLETTO, E.; MATSUOKA, M. T. O gps: unindo ciência e tecnologia em aulas de física. **Revista Brasileira de Ensino de Física**, Sociedade Brasileira de Física, v. 33, n. 2, p. 2313, Apr 2011. ISSN 1806-1117. Disponível em: <<https://doi.org/10.1590/S1806-11172011000200014>>.