UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINICIUS FLYSSAK

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL BASEADO EM GEOLOCALIZAÇÃO PARA AUXÍLIO NA LOCALIZAÇÃO DE VEÍCULOS ESTACIONADOS

PATO BRANCO

2025

VINICIUS FLYSSAK

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL BASEADO EM GEOLOCALIZAÇÃO PARA AUXÍLIO NA LOCALIZAÇÃO DE VEÍCULOS ESTACIONADOS

Development of a Mobile Application Based on Geolocation to Assist in Locating Parked Vehicles

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Érick Oliveira Rodrigues.
Coorientador: Prof. Robison Cris Brito.

PATO BRANCO

2025

VINICIUS FLYSSAK

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL BASEADO EM GEOLOCALIZAÇÃO PARA AUXÍLIO NA LOCALIZAÇÃO DE VEÍCULOS ESTACIONADOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Nome do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 17 de fevereiro de 2025.

Vinicius Pegorini
Mestrado
Universidade Tecnológica Federal do Paraná

Andreia Scariot Beulke
Mestrado
Universidade Tecnológica Federal do Paraná

Érick Oliveira Rodrigues
Doutorado
Universidade Tecnológica Federal do Paraná

Robison Cris Brito
Doutorado
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2025

Dedico este trabalho à minha esposa, meus pais e minha irmã, os quais me apoiaram do início ao fim nesta jornada.

Qualquer tecnologia suficientemente avançada é
indistinguível da magia.
(Clarke; Arthur, 1962).

RESUMO

O aumento da quantidade de veículos tem tornado o estacionamento e a lembrança do local de estacionamento tarefas desafiadoras, especialmente em grandes cidades e para pessoas com rotinas intensas. Este projeto propõe o desenvolvimento de um aplicativo para registro automatizado do local de estacionamento de veículos, visando facilitar a vida dos motoristas, reduzir perdas de tempo e evitar situações de estresse. O aplicativo permitirá o registro manual e automático da localização utilizando sensores do dispositivo, como GPS, giroscópio e acelerômetro. A solução foi desenvolvida inicialmente para dispositivos Android, utilizando Flutter para o *front-end* e Firebase para a gestão do banco de dados, com a possibilidade de expansão para iOS futuramente. Os testes realizados demonstraram a eficácia do aplicativo. Em 10 testes de precisão em estacionamentos abertos, o sistema registrou corretamente a posição em todos os casos, com apenas um registro duplicado, indicando um alto nível de precisão.

Palavras-chave: aplicativo móvel; estacionamento; GPS; automação, nuvem.

ABSTRACT

The increase in the number of vehicles has made parking and remembering where to park challenging tasks, especially in large cities and for people with intense routines. This project proposes the development of an application for automated registration of vehicle parking locations, aiming to make drivers' lives easier, reduce wasted time and avoid stressful situations. The application will allow manual and automatic location recording using device sensors, such as GPS, gyroscope and accelerometer. The solution was initially developed for Android devices, using Flutter for the front-end and Firebase for database management, with the possibility of expanding to iOS in the future. The tests carried out demonstrated the effectiveness of the application. In 10 accuracy tests in open parking lots, the system correctly recorded the position in all cases, with only one duplicate record, indicating a high level of accuracy.

Keywords: mobile application; parking; GPS; automation, cloud.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AOT	*Ahead-of-Time*
API	*Application Programming Interface* (Interface de Programação de Aplicação)
GPS	*Global Positioning System*
JIT	*Just-in-Time*
RF	Requisito Funcional
RNF	Requisito Não Funcional

SUMÁRIO

1 INTRODUÇÃO

Esse capítulo contém as considerações iniciais, os objetivos, a justificativa e a estrutura do trabalho.

1.1 Considerações iniciais

O aumento expressivo da quantidade de veículos automotores em circulação tem gerado desafios para os motoristas, incluindo a dificuldade em localizar o local de estacionamento, especialmente em ambientes movimentados, como shoppings, supermercados ou até mesmo em ruas com grande fluxo. Esse problema é recorrente e causa desconforto, perda de tempo e estresse, afetando a rotina de muitas pessoas.

Embora existam algumas soluções no mercado, como o recurso de marcar estacionamento no Google Maps, essas ferramentas dependem da interação manual do usuário, que precisa abrir o aplicativo e registrar a localização a cada uso. No entanto, devido à correria do dia a dia, essa etapa muitas vezes é esquecida.

Este trabalho propõe o desenvolvimento de um aplicativo que registra automaticamente o local onde o veículo foi estacionado, eliminando a necessidade de interação direta do usuário.

Para isso, o aplicativo utilizará sensores do dispositivo, como GPS (Global Positioning System), acelerômetro e giroscópio, para identificar alterações nos padrões de movimentação. Quando o veículo parar, o sistema detectará essa mudança e registrará automaticamente a localização, garantindo precisão e praticidade.

1.2 Objetivos

Em seguida serão apresentados o objetivo geral e os objetivos específicos do presente trabalho, sendo o objetivo geral relacionado com o resultado principal da realização deste trabalho e os objetivos específicos sendo complementares para o geral em termos de funcionalidades do sistema.

1.2.1 Objetivo Geral

Desenvolver um aplicativo para localizar veículos estacionados, demarcando o estacionamento de forma automática ou manual.

1.2.2 Objetivos específicos

- Implementar a capacidade de obter dados via sensores do celular, para definir quando o usuário saiu do veículo.
- Implementar conexão com a API do Google Maps para apresentação dos estacionamentos.
- Utilizar tecnologias de geolocalização para obter as coordenadas do local de estacionamento.
- Implementar um sistema de localização que permite ao usuário visualizar a posição do veículo estacionado em tempo real.

1.3 Justificativa

Atualmente, a crescente frota de veículos tem intensificado os desafios relacionados ao estacionamento, especialmente a dificuldade de localizar o veículo após longos períodos. Em 2023, por exemplo, o Brasil registrou 4.108.041 novos emplacamentos, segundo Albuquerque (2024), aumentando a lotação de estacionamentos e gerando perda de tempo significativa para motoristas.

Além disso, estudos indicam que o esquecimento do local de estacionamento é uma fonte comum de estresse, levando motoristas a acreditarem, equivocadamente, que seus veículos foram roubados, conforme relatado pela RPC Londrina (2021). Essa situação se agrava com a rotina caótica e a falta de soluções práticas para o problema.

As ferramentas disponíveis atualmente, como o recurso de marcação manual do Google Maps, dependem do registro ativo por parte do usuário, o que pode ser inconveniente, especialmente em situações em que o indivíduo está com as mãos ocupadas ou simplesmente esquece de realizar a tarefa. Essa limitação revela uma lacuna significativa nas soluções existentes: a falta de um método eficiente e automatizado para registrar o local de estacionamento.

O presente projeto busca preencher essa lacuna ao propor um aplicativo capaz de registrar automaticamente a localização do veículo ao identificar o momento em

que ele foi estacionado. Esse processo utilizará sensores já disponíveis nos dispositivos móveis, como GPS, acelerômetro e giroscópio, para detectar padrões de parada. A automatização desse registro visa não apenas economizar tempo, mas também reduzir o estresse diário e evitar situações desnecessárias de preocupação.

Como suporte técnico para o desenvolvimento do projeto, será utilizada a pesquisa de Frandaloso (2023), intitulada “Análise de dados para identificação de atividades de estacionamento de veículos por meio de aprendizado de máquina”. Essa pesquisa oferece um modelo de identificação de atividades relacionadas ao estacionamento, que pode ser adaptado para aprimorar o algoritmo de detecção do aplicativo proposto.

Dessa forma, o projeto pretende oferecer uma solução que impacte positivamente a rotina dos usuários, contribuindo para a otimização de tempo e melhoria da experiência em estacionamentos.

1.4 Estrutura do trabalho

Este trabalho de conclusão de curso está organizado em capítulos, onde o capítulo atual representa as considerações iniciais, os objetivos e a justificativa, o capítulo seguinte representa o referencial teórico, o terceiro capítulo representa materiais e método utilizados para o desenvolvimento, o quarto capítulo apresenta os resultados da realização do trabalho e, por fim, o quinto capítulo apresenta a conclusão.

2. REFERENCIAL TEÓRICO

O referencial teórico deste trabalho busca explorar os conceitos e tecnologias fundamentais para a construção do aplicativo proposto, com foco em sensores de *smartphones*, tecnologias de geolocalização, e integração com APIs de mapeamento.

2.1 Sensores em smartphones

Os *smartphones* modernos estão equipados com uma variedade de sensores que permitem realizar diversas tarefas, desde medir movimento até determinar localização. Esses sensores têm se tornado ferramentas essenciais para aplicações que dependem de informações sobre a posição e o comportamento do dispositivo. No contexto deste trabalho, os sensores mais relevantes incluem o acelerômetro, o giroscópio e o GPS.

O acelerômetro é um sensor que mede a aceleração linear do dispositivo em três eixos (x, y, z). Ele é capaz de detectar a velocidade com que o *smartphone* se movimenta e a direção do movimento. Essa funcionalidade permite, por exemplo, estimar se o usuário está em um veículo com base na aceleração registrada, além de identificar quando o dispositivo está em repouso. Aplicações como pedômetros e monitoramento de atividades físicas já utilizam amplamente esse sensor para medir passos e padrões de movimento.

Outro sensor crucial é o giroscópio, que mede a rotação angular em torno dos eixos do dispositivo. Ele complementa o acelerômetro ao oferecer informações mais precisas sobre a orientação do *smartphone*. De acordo com Nield (2020), o giroscópio foi introduzido amplamente no mercado de *smartphones* em 2010, com o lançamento do iPhone 4, e desde então tornou-se um componente padrão na maioria dos dispositivos modernos. Esse sensor é especialmente útil em aplicações como realidade aumentada, jogos e navegação, onde a precisão na detecção de movimentos e orientação é essencial.

O GPS é um dos sensores mais utilizados para determinar a localização geográfica de um dispositivo. Ele funciona conectando-se a uma rede de satélites para calcular a posição do *smartphone* com alta precisão, sem a necessidade de conexão à internet. Essa característica torna o GPS indispensável em aplicativos de navegação, mapeamento e localização, mesmo em áreas onde não há sinal de rede

móvel. No contexto deste trabalho, o GPS desempenha um papel fundamental ao registrar automaticamente a localização do veículo estacionado.

Estudos e aplicações existentes demonstram a eficácia do uso combinado desses sensores. Por exemplo, sistemas de monitoramento de deslocamento urbano utilizam o acelerômetro e o GPS para traçar rotas e medir distâncias, enquanto o giroscópio é empregado em soluções de estabilização e orientação, como drones e câmeras. A integração desses sensores em um aplicativo móvel oferece uma abordagem robusta para a detecção automatizada de atividades, como estacionar um veículo, e garante precisão na coleta de dados para melhorar a experiência do usuário.

O magnetômetro é um sensor capaz de medir a intensidade e a direção dos campos magnéticos ao redor do dispositivo, funcionando de maneira semelhante a uma bússola digital. Ele é amplamente utilizado em sistemas de navegação e realidade aumentada, permitindo que os dispositivos determinem sua orientação no espaço. No contexto deste projeto, o magnetômetro auxilia na identificação de mudanças na posição do *smartphone*, complementando as informações fornecidas pelo acelerômetro e giroscópio. Isso é especialmente útil para detectar se o usuário pegou o celular após estacionar o veículo, reduzindo a ocorrência de falsos positivos na detecção automática de estacionamento. Estudos indicam que a fusão de dados do magnetômetro com outros sensores melhora a precisão na estimativa de orientação e localização (Gentile *et al*., 2012).

2.2 Geolocalização

A geolocalização é uma tecnologia que permite determinar a posição geográfica de um dispositivo em tempo real, utilizando diferentes métodos e tecnologias para esse fim. Entre as abordagens mais comuns estão o GPS (Global Positioning System), a triangulação de torres de celular e o Wi-Fi, cada uma com suas vantagens e limitações em termos de precisão e cobertura.

O GPS, uma das formas mais populares de geolocalização, opera através de uma constelação de satélites que transmitem sinais continuamente. Ao receber esses sinais, o dispositivo calcula sua posição com base no tempo que cada sinal levou para chegar até ele, atingindo uma precisão que pode variar de alguns metros a centímetros, dependendo das condições ambientais e do receptor utilizado.

A triangulação de torres de celular é um método amplamente utilizado em áreas urbanas. Nesse caso, a localização é estimada com base na distância do dispositivo em relação a três ou mais torres de celular próximas, o que fornece uma posição aproximada. Embora esse método seja menos preciso que o GPS, ele é útil em locais onde o sinal de satélite está obstruído, como dentro de edifícios ou túneis.

O uso de redes Wi-Fi para geolocalização também é comum, especialmente em ambientes fechados. O dispositivo coleta informações sobre redes Wi-Fi próximas e as compara com um banco de dados que contém as localizações dessas redes. Essa abordagem oferece boa precisão em áreas densamente povoadas e é frequentemente usada em conjunto com outros métodos para aumentar a confiabilidade.

Segundo Gentile *et al*. (2012), as abordagens tradicionais de geolocalização combinam medições de alcance e ângulo com correlações geométricas entre coordenadas de pontos de referência, utilizando dispositivos com informações de localização pré-programadas ou derivadas de GPS como base. Além disso, sistemas modernos como o *Global Navigation Satellite System* (GNSS), que inclui outras constelações de satélites como GLONASS, Galileo e BeiDou, oferecem maior robustez e precisão em condições adversas.

No contexto do aplicativo, a geolocalização desempenha um papel central ao registrar automaticamente a localização do veículo estacionado. Ao combinar tecnologias como GPS e redes Wi-Fi, o aplicativo busca oferecer precisão mesmo em locais onde os métodos tradicionais enfrentam desafios, como estacionamentos cobertos. Estudos prévios demonstram a eficácia dessas tecnologias para rastreamento em tempo real, como em aplicativos de navegação e gerenciamento logístico, mostrando que a integração desses métodos pode melhorar significativamente a experiência do usuário.

2.3 Google Maps API

A API do Google Maps consiste em um serviço disponibilizado de forma gratuita, onde é possível anexar os mapas do Google em sua aplicação. Existem diversas opções disponíveis atualmente, cada uma possuindo suas particularidades, as quais podem ser vistas no Quadro 1.

Quadro 1 – Tipos de APIs do Google Maps

Api	Descrição
Google Maps JavaScript API	Incorpora um mapa do Google em sua página da web usando JavaScript. Manipula o mapa e adicione conteúdo com a ajuda de vários serviços.
Google Earth API	Incorpora um verdadeiro globo digital em 3D na página da web. Leva os visitantes a qualquer lugar da Terra (até mesmo nas profundezas dos oceanos) sem tirá-los de sua página da web.
Google Static Maps API	Incorpora uma imagem simples e rápida do Google Maps em sua página da web ou site para celular sem precisar de códigos JavaScript ou qualquer carregamento dinâmico de página.
Serviços da web	Usa solicitações de URL para acessar informações de geocodificação, rotas, elevação e lugares dos aplicativos cliente e manipule os resultados em JSON ou XML.
Google Maps Data API	Visualiza, armazena e atualiza dados de mapa por meio de feeds da Google Data API, usado um modelo de elementos (marcadores, linhas e formas) e coleções de elementos.

Fonte: Adaptado de DevMedia (2024).

2.4 Detecção Automática de Estacionamento

A automação do processo de registro da localização de estacionamento tem sido um desafio estudado por diversas pesquisas, visto que muitos motoristas não anotam conscientemente onde estacionaram. Estudos mostram que, em centros urbanos congestionados, uma parcela significativa do tráfego é composta por motoristas em busca de uma vaga ou tentando localizar seus veículos (Nguyen *et al*., 2017). Para solucionar esse problema, diversos sistemas têm sido desenvolvidos, utilizando diferentes abordagens e sensores para detectar automaticamente o momento em que um veículo é estacionado.

Dentre as soluções propostas, destaca-se o ParkSense, um sistema que emprega variações de campos magnéticos e eletromagnéticos para determinar automaticamente quando o motorista parou e desligou o veículo. Segundo Nguyen *et al*. (2017), esse sistema utiliza uma combinação de magnetômetro, acelerômetro e giroscópio para detectar alterações no ambiente magnético dentro do veículo e, com isso, identificar o momento exato em que o carro foi estacionado. Além disso, o ParkSense emprega um modelo probabilístico, capaz de distinguir motoristas de passageiros e minimizar falsos positivos.

Os testes realizados pelos autores demonstraram que essa abordagem é altamente eficiente, atingindo mais de 90% de precisão na identificação do estacionamento, além de possuir baixo consumo de energia. Isso ocorre porque o sistema evita verificações constantes de GPS, operando de forma passiva na

detecção de padrões magnéticos. Dessa maneira, o ParkSense se apresenta como uma alternativa interessante para aplicações que demandam maior eficiência energética e precisão na detecção de eventos veiculares.

Contudo, essa solução possui algumas limitações. Como sua abordagem se baseia em variações magnéticas internas do veículo, a precisão pode ser comprometida em diferentes modelos de carros, especialmente naqueles que possuem menor interferência eletromagnética. Além disso, a dependência do magnetômetro pode tornar o sistema menos genérico, exigindo calibração específica para diferentes veículos e condições ambientais.

Diferente dessa abordagem, o presente projeto propõe um sistema alternativo de detecção automática do estacionamento, utilizando uma combinação de GPS, acelerômetro e giroscópio. O uso do GPS permite registrar com precisão a localização exata do estacionamento, enquanto os sensores de movimento identificam padrões de desaceleração e imobilização do veículo. Essa abordagem apresenta algumas vantagens em relação ao ParkSense, como:

- Independência do modelo do veículo, garantindo maior aplicabilidade, pois não depende de sensores internos do carro.

- Maior precisão na geolocalização, permitindo ao usuário visualizar diretamente no mapa onde seu veículo foi estacionado.

- Menor necessidade de calibração, tornando o sistema mais genérico e funcional em qualquer contexto.

Entretanto, o sistema baseado em GPS também apresenta desafios, especialmente em ambientes fechados, como estacionamentos subterrâneos, onde o sinal do satélite pode ser comprometido. Além disso, o consumo de bateria pode ser mais alto, caso não sejam adotadas estratégias eficientes para otimizar a ativação do GPS e minimizar seu uso contínuo.

Dessa forma, uma possível melhoria para este projeto seria a implementação de um modelo híbrido, combinando a abordagem do ParkSense com a metodologia atual. Isso poderia incluir:

- Fusão de sensores, utilizando padrões do acelerômetro e giroscópio para ativar o GPS apenas quando necessário.

- Modelos probabilísticos, semelhantes aos utilizados no ParkSense, para minimizar falsos positivos e evitar registros desnecessários.

Otimização energética, garantindo que o GPS só seja ativado em situações críticas, reduzindo o consumo de bateria.

Com essas melhorias, o sistema proposto poderia oferecer maior precisão, menor consumo energético e maior aplicabilidade, tornando-se uma solução robusta para auxiliar motoristas na localização de seus veículos.

2.5 Estado da arte

Nesta seção, serão apresentados aplicativos existentes voltados para marcar a localização do veículo estacionado, com uma análise comparativa das funcionalidades, vantagens, limitações e a forma como eles abordam o problema da localização do carro.

2.5.1 Google Maps

O Google Maps é amplamente utilizado para navegação e mapas, oferecendo a funcionalidade de marcar a localização onde o veículo foi estacionado. No entanto, essa funcionalidade exige que o usuário insira manualmente o local de estacionamento, o que pode ser inconveniente e suscetível ao esquecimento. Além disso, o processo manual pode ser impreciso, já que o usuário precisa parar e interagir com o aplicativo para registrar o ponto de estacionamento. Embora amplamente utilizado, o Google Maps carece de automação no processo de marcação da localização, um ponto que pode levar a erros de registro.

2.5.2 Waze

Waze é outro aplicativo popular de navegação, com uma funcionalidade semelhante ao Google Maps para marcar a localização do veículo estacionado. Assim como o Google Maps, o processo de registrar o estacionamento é manual, o que exige que o usuário abra o aplicativo, insira a localização ou permita que o aplicativo registre a posição. Embora o Waze seja eficaz para navegação e encontrar o caminho mais rápido, a funcionalidade de registro da localização do veículo não difere muito do que já é oferecido pelo Google Maps, o que é insuficiente em termos de automação e conveniência.

2.5.3 Find My Car

Aplicativos como Find My Car são especificamente desenvolvidos para ajudar os usuários a encontrarem seu veículo estacionado. Esses aplicativos geralmente utilizam a geolocalização para salvar a posição do carro quando o usuário estaciona. No entanto, mesmo com essa funcionalidade, o processo ainda exige que o usuário interaja com o aplicativo para registrar a localização, o que pode ser negligenciado, especialmente em situações de pressa ou distração. Embora o app seja mais específico do que o Google Maps ou o Waze, ele ainda depende da ação manual do usuário.

2.5.4 Comparativo com o projeto

As soluções existentes para registrar a localização de um veículo dependem da ação manual do usuário, o que pode levar a esquecimentos, registros incorretos e falta de praticidade. Em contraste, o projeto proposto adota uma abordagem automatizada, eliminando a necessidade de interação do usuário para salvar a posição do carro.

O aplicativo utiliza sensores do *smartphone*, como acelerômetro, giroscópio e magnetômetro, para identificar padrões de movimento. Assim que o veículo para e o motorista sai do carro, o sistema detecta automaticamente o estacionamento e salva a localização sem necessidade de abrir o aplicativo ou tocar em um botão.

Esse método resolve uma das principais limitações dos aplicativos mencionados, que exigem que o usuário ativamente registre a posição. O Quadro 2 apresenta uma comparação entre as funcionalidades e limitações das soluções existentes em relação ao aplicativo desenvolvido.

Quadro 2 – Comparação de Aplicativos para Localização de Estacionamento

Aplicativo	Registro Automático	Dependência de ação manual	Vantagens	Desvantagens
Google Maps	Não	Sim	Muito conhecido, acesso amplo	Requer interação manual, não é específico para estacionamento
Waze	Não	Sim	Navegação otimizada, comunidade ativa	Processo manual, não é específico para estacionamento

Find My Car	Não	Sim	Focado em estacionamento	Requer ação manual
Aplicativo proposto	Sim	Não	Automático, aplicativo focado em estacionamentos, sincronização com a nuvem	Necessita GPS, giroscópio, acelerômetro e magnetômetro

Fonte: Autoria própria (2024).

3 MATERIAIS E MÉTODO

Esse capítulo refere-se aos materiais e método utilizados durante o desenvolvimento deste trabalho de conclusão de curso. Os materiais se referem as tecnologias, editores de código e linguagem do banco de dados empregadas para o desenvolvimento do aplicativo. O método se refere às atividades principais realizadas para se obter os resultados.

3.1 Materiais

As ferramentas e tecnologias que serão utilizadas para o desenvolvimento deste trabalho estão listadas no Quadro 3.

Quadro 3 – Lista de ferramentas e tecnologias

Ferramenta/Tecnologia	Versão	Finalidade
Flutter	3.22.1	*Framework full-Stack*
Dart	3.2.4	Linguagem de programação
Visual Studio Code	1.82.2	IDE de desenvolvimento
Cloud Firestore	Não se aplica	Banco de dados
Google Maps API	Não se aplica	API necessária para obter localização e apresentar em mapa

Fonte: Autoria própria (2024).

3.1.1 *Flutter*

O *Flutter* é um framework de desenvolvimento multiplataforma criado pelo Google em 2017, com foco na criação de aplicativos para dispositivos móveis, web e desktop. Sua principal vantagem é a capacidade de desenvolver para múltiplas plataformas (iOS, Android, web e desktop) a partir de um único código-base, reduzindo significativamente o tempo e o custo de desenvolvimento.

O *Flutter* utiliza a linguagem de programação Dart, também desenvolvida pelo Google, que foi projetada para oferecer alta produtividade e desempenho. O Dart combina características de linguagens modernas, como tipagem forte, inferência de tipos e suporte à programação assíncrona, o que facilita a criação de aplicativos complexos e responsivos.

A arquitetura do *Flutter* é baseada em três componentes principais: o motor de renderização *Skia*, a biblioteca de widgets e o *runtime* do *Dart*. O *Skia* é um motor

gráfico de alto desempenho que permite ao *Flutter* renderizar interfaces de usuário diretamente na tela, sem depender de componentes nativos do sistema operacional. Isso garante que os aplicativos *Flutter* tenham uma aparência e comportamento consistentes em todas as plataformas.

O *Flutter* utiliza dois modos de compilação: *Just-in-Time (JIT)* durante o desenvolvimento e *Ahead-of-Time (AOT)* para produção. No modo JIT, o código *Dart* é compilado em tempo real, permitindo o uso do *hot reload*, uma funcionalidade que atualiza a interface do aplicativo instantaneamente após alterações no código, sem a necessidade de recompilar o projeto inteiro. Isso acelera significativamente o ciclo de desenvolvimento, permitindo que os desenvolvedores vejam as mudanças em tempo real. Já no modo AOT, o código é compilado para binários nativos antes da execução, garantindo desempenho otimizado e eficiência em aplicativos publicados.

O desempenho do *Flutter* é frequentemente comparado ao de aplicativos nativos, e estudos indicam que ele pode ser tão eficiente quanto o desenvolvimento nativo em muitos cenários. De acordo com um estudo publicado pela Invertase (2022), o *Flutter* apresenta desempenho próximo ao do Java no Android e é cerca de 15% mais rápido que o Swift no iOS em operações de renderização e processamento de dados. Isso se deve à sua arquitetura única, que elimina a necessidade de uma "ponte" entre o código e os componentes nativos, reduzindo a sobrecarga de comunicação.

Além disso, o *Flutter* é otimizado para operações gráficas intensivas, como animações e transições, graças ao uso do *Skia* e à capacidade de renderização direta no *canvas*. Isso o torna uma escolha popular para aplicativos que exigem interfaces de usuário ricas e interativas.

O *Flutter* tem sido amplamente adotado por empresas de grande porte devido à sua eficiência e versatilidade. Aplicativos como iFood, Nubank e Carteira Digital de Trânsito foram desenvolvidos utilizando o *Flutter*, demonstrando sua capacidade de suportar projetos complexos e de alta escala. A escolha do *Flutter* para esses projetos se deve não apenas ao seu desempenho, mas também à sua capacidade de unificar equipes de desenvolvimento e reduzir custos de manutenção.

O *Flutter* se destaca como uma solução moderna e eficiente para o desenvolvimento de aplicativos multiplataforma. Sua arquitetura única, combinada com ferramentas como o *hot reload* e o uso do *Dart*, oferece uma experiência de desenvolvimento ágil e produtiva. Além disso, seu desempenho comparável ao de

aplicativos nativos e sua adoção por grandes empresas reforçam sua credibilidade como uma escolha robusta para projetos de software.

3.1.2 *Cloud Firestore*

O *Cloud Firestore* é um banco de dados *NoSQL* baseado em nuvem, desenvolvido pelo Google como parte da plataforma *Firebase*. Diferente de bancos de dados relacionais tradicionais, o *Cloud Firestore* é projetado para armazenar e sincronizar dados em tempo real, sendo ideal para aplicações que exigem baixa latência e alta escalabilidade, como aplicativos de mensagens, sistemas de colaboração e jogos online (Google, 2024).

Segundo a documentação oficial do Google, o *Cloud Firestore* organiza os dados em coleções e documentos, onde cada documento pode conter subcoleções e pares de chave-valor. Essa estrutura flexível permite que os desenvolvedores modelem dados de forma hierárquica, adaptando-se a diferentes necessidades de aplicações (Google, 2024). Além disso, o *Cloud Firestore* oferece suporte a consultas avançadas, incluindo filtros, ordenação e paginação, o que facilita a recuperação eficiente de dados (Google, 2024).

Uma das principais vantagens do *Cloud Firestore* é sua capacidade de sincronização em tempo real. Conforme destacado na documentação oficial, qualquer alteração nos dados é automaticamente propagada para todos os clientes conectados, eliminando a necessidade de operações manuais de sincronização. Isso é particularmente útil para aplicativos que exigem atualizações instantâneas, como chats ou dashboards interativos (Google, 2024).

O *Cloud Firestore* também é altamente escalável, sendo capaz de gerenciar grandes volumes de dados e suportar milhares de usuários simultâneos sem comprometer o desempenho. Ele oferece recursos como transações ACID, garantindo a consistência dos dados, e suporta integração com outras ferramentas do Google, como o Google Cloud Platform (GCP), para funcionalidades avançadas como machine learning e análise de big data (Google, 2024).

Outro ponto forte do *Cloud Firestore* é sua segurança. Ele permite a definição de regras de acesso granular, que podem ser configuradas para restringir ou permitir o acesso a dados com base em permissões de usuários ou condições específicas.

Isso torna o Cloud Firestore uma escolha segura para aplicativos que lidam com dados sensíveis (Google, 2024).

Além disso, o *Cloud Firestore* é compatível com diversas linguagens de programação e frameworks, como *JavaScript, Node.js, Flutter, React Native, Swift e Kotlin*, o que o torna uma solução versátil para desenvolvedores (Google, 2024). Sua integração com o ecossistema *Firebase*, incluindo serviços como *Firebase Authentication, Cloud Functions e Firebase Storage*, facilita o desenvolvimento de aplicativos completos e funcionais.

Por fim, o *Cloud Firestore* é suportado por uma comunidade ativa e documentação abrangente, além de oferecer planos gratuitos e pagos, dependendo das necessidades do projeto. Sua combinação de flexibilidade, escalabilidade e facilidade de uso faz do *Cloud Firestore* uma das soluções mais populares para desenvolvimento de aplicativos modernos (Google, 2024).

3.1.3 Google Maps API

A Google Maps API é um conjunto de ferramentas desenvolvidas pelo Google que permite a integração de mapas interativos em aplicações web e mobile. A API oferece funcionalidades como exibição de mapas, cálculos de rotas, geolocalização, geocodificação e pontos de interesse. Seu uso se tornou amplamente difundido devido à precisão dos dados e à flexibilidade oferecida para personalizar e adaptar mapas de acordo com as necessidades dos desenvolvedores (Google, 2024).

No contexto deste projeto, a API do Google Maps foi utilizada para exibir a localização atual do usuário e o último local onde o veículo foi estacionado. Através dos serviços de geolocalização, o aplicativo consegue identificar a posição do usuário em tempo real, enquanto a funcionalidade de marcadores permite indicar visualmente o ponto de estacionamento no mapa.

Além disso, a API do Google Maps permite a personalização da interface, possibilitando ajustes no estilo do mapa e na exibição dos elementos gráficos. Outra funcionalidade relevante utilizada foi o cálculo de rotas, permitindo que o usuário possa navegar até a localização do estacionamento de maneira eficiente.

Entre as principais vantagens da API do Google Maps, destacam-se a facilidade de implementação, alta precisão dos dados e suporte multiplataforma. No entanto, um dos desafios de sua utilização está no uso contínuo da conexão com a

internet, o que pode impactar a experiência do usuário em áreas com sinal fraco (Devmedia, 2013).

3.2 Método

A metodologia de desenvolvimento escolhida para este projeto foi o *Kanban*, devido à sua simplicidade, flexibilidade e capacidade de proporcionar uma visão clara do fluxo de trabalho. O *Kanban* é amplamente utilizado no contexto ágil por sua abordagem visual, permitindo gerenciar tarefas de maneira eficiente. Esse modelo se mostrou ideal para um projeto individual, pois facilita a organização das atividades e a identificação de gargalos, sem a necessidade de cerimônias formais presentes no *Scrum*, como o Planejamento de *Sprint*, a Revisão e a Retrospectiva da *Sprint*.

O *Kanban* é baseado em um sistema de quadros (*boards*) que organizam as tarefas em colunas, representando diferentes estágios do processo de desenvolvimento.

As principais características do *Kanban* aplicadas neste projeto são:

- *Backlog*: uma lista de todas as tarefas a serem realizadas, priorizadas de acordo com a importância e urgência.

- A Fazer (*To Do*): tarefas que foram priorizadas e estão prontas para serem iniciadas.

- Em Progresso *(In Progress*): tarefas que estão sendo executadas no momento.

- Concluído *(Done*): tarefas finalizadas e revisadas.

Diferente de metodologias como o *Scrum*, o *Kanban* não possui iterações fixas (como sprints) ou reuniões obrigatórias (como *Daily Scrum*). Em vez disso, o foco está na visualização contínua do fluxo de trabalho e na limitação do número de tarefas em andamento, o que ajuda a evitar sobrecarga e a manter o foco nas atividades mais importantes.

A escolha do Kanban se mostrou eficaz para este projeto, pois permitiu uma gestão visual e dinâmica das tarefas, além de facilitar a identificação de gargalos e a priorização de atividades. Essa metodologia também se alinha bem com o desenvolvimento incremental, onde funcionalidades são implementadas e testadas de forma contínua, garantindo entregas consistentes e de qualidade.

3.3 Decisões de implementação

Nesta seção, serão discutidos os motivos para a escolha do *Flutter*, do *Firestore* e da API do Google Maps no desenvolvimento deste projeto.

3.3.1 Por que *Flutter*?

O *Flutter* foi escolhido para este projeto devido à sua alta capacidade de adaptação, permitindo o desenvolvimento de um único código que pode ser executado tanto em Android quanto em iOS. Essa característica reduz significativamente o esforço necessário para manter versões separadas do aplicativo para diferentes plataformas.

Além disso, por ser uma tecnologia desenvolvida pelo Google, o *Flutter* possui integração nativa com o *Firebase*, o que facilita a implementação de serviços de *backend* sem necessidade de configurações complexas.

Outro fator determinante para essa escolha foi o conhecimento prévio do desenvolvedor na tecnologia, tornando o desenvolvimento mais ágil e eficiente. Além disso, o *Flutter* conta com uma documentação extensa e suporte ativo da comunidade, ambos mantidos pelo Google, o que contribui para a rápida resolução de dúvidas e problemas técnicos.

Uma vantagem essencial do *Flutter* no desenvolvimento deste projeto foi o recurso *Hot Reload*, que permite visualizar alterações no código em tempo real, sem a necessidade de recompilar a aplicação. Esse recurso foi importante para a redução do tempo de desenvolvimento, pois possibilitou testes e ajustes rápidos durante a implementação das funcionalidades do sistema

3.3.2 Por que *Firebase*?

O *Firebase* foi escolhido como banco de dados devido à sua facilidade de integração com o *Flutter* e à sua capacidade de armazenamento em nuvem, permitindo acesso remoto aos dados do aplicativo.

Uma das principais vantagens do *Firebase* é ser um banco de dados *NoSQL* escalável, ideal para aplicações de pequeno e médio porte. Além disso, sua política

de precificação permite que projetos com poucos acessos e usuários operem gratuitamente, tornando-o uma solução econômica e viável para esta aplicação.

Outro diferencial do *Firebase* é a possibilidade de definir limites de custo e consumo, evitando gastos inesperados com a infraestrutura do banco de dados.

Além disso, o *Firebase* elimina a necessidade de um *backend* tradicional, pois fornece recursos que permitem configurar regras de segurança para controle de acesso à adição, edição, visualização e exclusão de registros. Isso simplifica a implementação e torna o desenvolvimento mais rápido e seguro.

Por fim, o *Firebase* possui uma comunidade ativa e uma documentação detalhada, desenvolvida e mantida pelo próprio Google, o que facilita a resolução de problemas e a adoção de boas práticas no desenvolvimento.

3.3.3 Por que Google Maps API?

A API do Google Maps foi escolhida devido ao seu suporte abrangente e documentação robusta, disponibilizados pelo próprio Google. Sua popularidade e confiabilidade a tornam uma das soluções mais utilizadas para integração de mapas interativos em aplicativos móveis.

Outro fator relevante é que o Google oferece um crédito gratuito de até 200 dólares mensais para uso da API. Dessa forma, enquanto o aplicativo estiver em fase de testes ou possuir poucos usuários ativos, não haverá custo para o uso dos serviços de mapas. Isso torna a API uma opção viável e econômica para aplicações em desenvolvimento.

Além disso, a API do Google Maps oferece diversos recursos essenciais, como:

- Geolocalização em tempo real, permitindo que o usuário visualize sua posição no mapa;

- Adição de marcadores personalizados, utilizados para indicar a localização do estacionamento;

- Cálculo de rotas, facilitando a navegação até o local onde o veículo foi estacionado.

Por conta dessas vantagens, a API do Google Maps se mostrou a melhor alternativa para este projeto.

3.4 Escopo

O aplicativo proposto neste trabalho tem como público-alvo indivíduos que enfrentam desafios cotidianos, como moradores de grandes cidades e pessoas com uma rotina acelerada. O sistema busca auxiliá-los a localizar seus veículos de maneira rápida e eficiente, evitando perda de tempo e estresse desnecessário.

Neste aplicativo, o usuário pode cadastrar o local onde estacionou seu veículo de forma precisa, com apresentação em formato de mapa. Além disso, é possível visualizar um histórico dos estacionamentos realizados.

O sistema permite também o cadastro de usuários, possibilitando o acesso ao histórico de estacionamentos em diferentes dispositivos. Essa funcionalidade é especialmente útil em situações em que o usuário sai acompanhado e a outra pessoa deseja retornar ao veículo previamente estacionado.

Se desejado, o usuário pode também tirar uma foto do local atual onde o veículo está estacionado, podendo auxiliá-lo a encontrar mais facilmente o veículo. Cada cadastro de estacionamento deve gerar um registro no sistema que deverá estar disponível para visualização *offline* e caso o usuário possua conexão com a internet, o seu histórico deve ser automaticamente sincronizado com a nuvem. Caso desejado, o mesmo deve possuir a opção de não manter histórico e de excluir o histórico completo da conta, visando preservar a privacidade do usuário.

Por padrão, o aplicativo deve ser capaz de registrar o estacionamento de forma automatizada, sem a necessidade de intervenção do usuário, utilizando sensores do seu dispositivo como acelerômetro, giroscópio e GPS, percebendo por meio da velocidade de movimentação do dispositivo quando o automóvel foi estacionado. Deve possuir também um método de economia de energia, onde o aplicativo não ficará em segundo plano e usará somente o sensor de GPS, onde neste caso, o usuário deve efetuar o registro de estacionamento de forma manual.

3.5 Modelagem do Sistema

No Quadro 4, são apresentados os requisitos funcionais do sistema, os quais referem-se as funcionalidades que o sistema deve possuir.

Quadro 4 – Requisitos funcionais

Identificação	Nome	Descrição
RF01	Salvar automaticamente a localização do veículo estacionado	Deve ser possível salvar a localização de onde o veículo foi estacionado de forma automatizada, sem a necessidade de interação do usuário com o aplicativo.
RF02	Salvar manualmente a localização do veículo estacionado	Deve ser possível que o usuário salve de forma manual a localização do estacionamento.
RF03	Gerenciar histórico	O sistema deve permitir que o usuário gerencie o histórico de estacionamentos, oferecendo opções para manter ou não manter os registros de estacionamentos efetuados.
RF04	Manter foto do último estacionamento efetuado	O sistema deve ser possível que o usuário consiga realizar a captura do último local estacionado, mantendo a foto localmente em seu dispositivo.
RF05	Emitir notificação ao estacionar	Deve emitir uma notificação para o usuário caso ele utilize o aplicativo de forma automatizada, onde avisa que o estacionamento foi salvo.
RF06	Criar perfil	Deve ser possível criar um perfil de usuário para que salve o local onde estacionou na nuvem.
RF07	Efetuar autenticação	Deve ser possível que usuários com perfil cadastrado, consigam realizar login no sistema, de forma que seja possível ver seu histórico de estacionamentos.
RF08	Desativar histórico	Deve possuir uma opção onde o usuário poderá optar por não manter o histórico de estacionamentos efetuados
RF09	Visualizar estacionamento	Deve ser possível visualizar o último estacionamento realizado no mapa.
RF10	Sincronizar histórico automaticamente	O histórico de estacionamentos deve ser sincronizado com a nuvem automaticamente ao possuir conexão com internet.
RF11	Modo economia de energia	Deve possuir um modo de economia de energia, onde o aplicativo não ficará em segundo plano e o registro do estacionamento deve ser feito manualmente.

Fonte: Autoria própria (2024).

No Quadro 5 são apresentados os requisitos não funcionais definidos para o projeto. Estes requisitos são relacionados a desempenho, otimização, disponibilidade, segurança, confiabilidade, entre outros.

Quadro 5 – Requisitos não funcionais

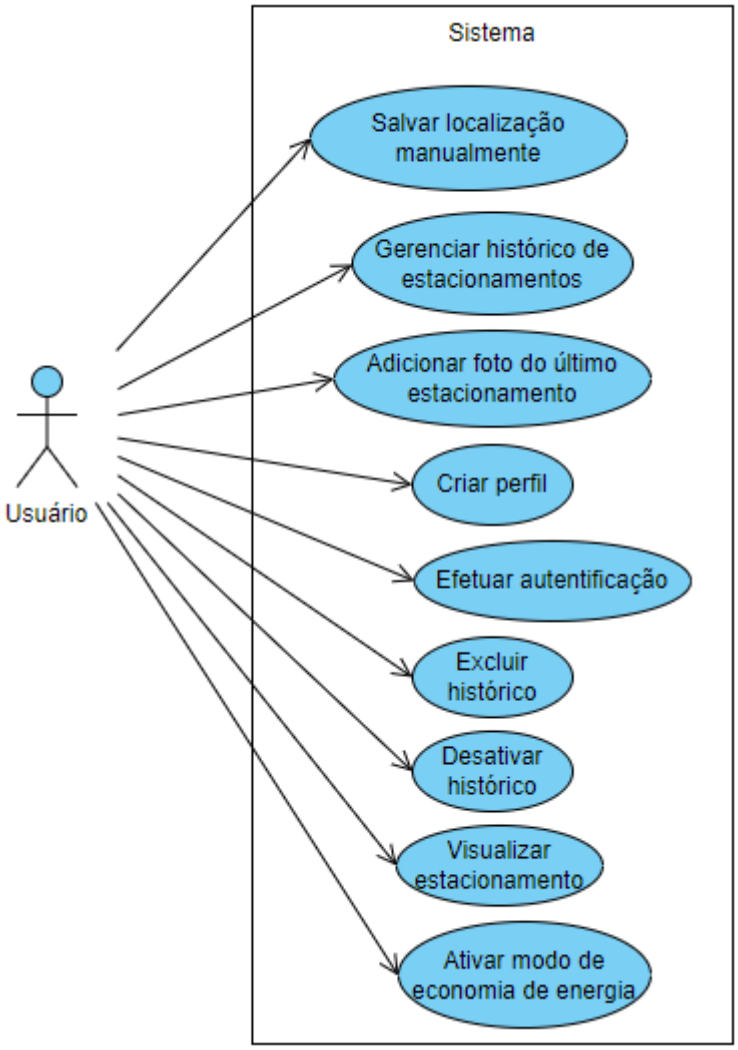
Identificação	Nome	Descrição
RNF01	Desempenho	O aplicativo deve apresentar baixo consumo de energia, principalmente para em casos em que o usuário optar por usar o modo automatizado.
RNF02	Disponibilidade	O aplicativo deve estar disponível na maior parte do tempo, com exceção em momento de manutenção programada.

RNF03	Segurança	O aplicativo deve ser seguro, garantindo que dados de cadastro dos usuários não fiquem expostos para terceiros.
RNF04	Usabilidade	A interface do aplicativo deve ser simples e intuitiva, de forma que novos usuários se adequem com facilidade.
RNF05	Confiabilidade	O aplicativo deve registrar com precisão onde o veículo foi estacionado.
RNF06	Privacidade	O aplicativo deve possibilitar que o usuário gerencie suas configurações de privacidade, com opções para possibilitar excluir ou editar dados.

Fonte: Autoria própria (2024).

A Figura 1 apresenta o diagrama de caso de uso, contendo as principais funcionalidades do sistema e seus atores onde, neste caso, é apenas um, o usuário. O usuário representa todas as pessoas que devem utilizar o aplicativo.

Figura 1 – Diagrama de caso de uso



Fonte: Autoria própria (2024).

Em seguida, serão apresentados quadros com expansão dos casos de uso relatados anteriormente.

No Quadro 6, está a expansão do caso de uso "salvar localização manualmente".

Quadro 6 - Expansão do caso de uso "salvar localização manualmente"

Caso de uso: Salvar localização manualmente.
Descrição: O usuário opta por salvar de forma manual a localização onde estacionou seu veículo.
Evento iniciador: Necessidade de salvar o local onde o veículo está estacionado.

Atores: Usuário.	
Pré-condição: - O dispositivo deve possuir acesso ao GPS.	
Sequência de eventos: 1. O usuário acessa o aplicativo; 2. Sistema apresenta mapa contendo a localização atual; 3. O usuário clica em "Marcar estacionamento"; 4. O sistema salva a localização atual como último estacionamento registrado.	
Pós-condições: Local onde o veículo está estacionado é salvo com sucesso.	
Extensão	Descrição
Localização desativada	Apresenta mensagem indicando que localização está desativada e solicita que o usuário realize a ativação

Fonte: Autoria própria (2024).

No Quadro 7, está a expansão do caso de uso "gerenciar histórico de estacionamentos".

Quadro 7 - Expansão do caso de uso "gerenciar histórico de estacionamentos"

Caso de uso: Gerenciar histórico de estacionamentos	
Descrição: O sistema mantém um histórico dos estacionamentos realizados pelo usuário salvos no aplicativo.	
Evento iniciador: Necessidade de incluir novo estacionamento.	
Atores: Usuário.	
Pré-condição: - O usuário deve estar autenticado.	
Sequência de eventos: 1. O usuário acessa o aplicativo; 2. O usuário registra novo estacionamento; 3. O sistema salva o estacionamento em um histórico; 4. O usuário acessa o histórico clicando em "Histórico de estacionamentos";	
Pós-condições: O histórico é atualizado com o novo registro.	
Extensão	Descrição
Histórico desativado	Nesse caso, não deve salvar nada no histórico e ao acessar a aba, deve apenas apresentar uma mensagem informando que o histórico está desativado.

Fonte: Autoria própria (2024).

No Quadro 8, está a expansão do caso de uso "adicionar foto do último estacionamento".

Quadro 8 - Expansão do caso de uso "adicionar foto do último estacionamento"

Caso de uso: Adicionar foto do último estacionamento	
Descrição: O sistema mantém uma foto do último estacionamento efetuado.	
Evento iniciador: Necessidade de incluir uma foto como auxílio para encontrar o último estacionamento registrado.	
Atores: Usuário	
Pré-condição: - O sistema deve possuir acesso a câmera; - A câmera deve estar funcionando corretamente.	
Sequência de eventos: 1. Ator acessa o aplicativo. 2. O usuário registra novo estacionamento. 3. O sistema solicita se o usuário deseja tirar uma foto do local. 4. O usuário confirma. 5. Irá iniciar a câmera e permitirá que o usuário realize a captura. 6. Captura é realizada e fica salva em vínculo com o registro.	
Pós-condições: O estacionamento fica salvo com uma foto vinculada.	
Extensão	Descrição
Sem acesso a câmera	Nesse caso, o aplicativo solicitará permissão e tentará executar novamente a tarefa.
Usuário recusa a solicitação de tirar foto	Nesse caso, o sistema irá salvar apenas o estacionamento.

Fonte: Autoria própria (2024).

No Quadro 9, está a expansão do caso de uso "criar perfil".

Quadro 9 - Expansão do caso de uso "criar perfil"

Caso de uso: Criar perfil
Descrição: O usuário cria um perfil para salvar os dados na nuvem.
Evento iniciador: Necessidade de salvar histórico de estacionamentos na nuvem
Atores: Usuário.

Pré-condição: - O usuário deve possuir acesso à internet.	
Sequência de eventos: 1. O usuário acessa o aplicativo. 2. O usuário acessa a tela de login. 3. O usuário seleciona "cadastrar-se". 4. O usuário coloca as informações solicitadas. 5. O usuário clica em "Cadastrar". 6. O sistema valida informações. 7. O sistema apresenta mensagem "Cadastro efetuado com sucesso!". 8. O sistema redireciona para tela inicial com usuário logado.	
Pós-condições: O perfil do usuário é criado.	
Extensão	Descrição
Dados inválidos	Nesse caso, o sistema irá apresentar uma mensagem para o usuário informando o que deve ser corrigido.
Email já cadastrado	Nesse caso, o sistema apresenta uma mensagem para o usuário informando que o e-mail já está cadastrado e solicita para que efetue login ou altere o e-mail.

Fonte: Autoria própria (2024).

No Quadro 10, está a expansão do caso de uso "efetuar autentificação".

Quadro 10 - Expansão do caso de uso "Efetuar autentificação"

Caso de uso: Efetuar autentificação	
Descrição: O sistema deve permitir que o usuário entre em sua conta.	
Evento iniciador: Necessidade de entrar em sua conta cadastrada.	
Atores: Usuário.	
Pré-condição: - O usuário deve possuir um perfil; - O usuário deve possuir acesso à internet.	
Sequência de eventos: 1. O usuário acessa ao aplicativo; 2. O usuário seleciona a opção "Login"; 3. O usuário insere as credenciais; 4. O sistema valida as credenciais informadas; 5. O sistema retorna a tela inicial com o usuário logado.	
Pós-condições: O usuário fica autenticado e pode acessar seus dados/histórico.	
Extensão	Descrição
Credenciais inválidas	Neste caso, o sistema irá apresentar uma mensagem de erro e solicitará que o usuário tente novamente.

Fonte: Autoria própria (2024).

No Quadro 11, está a expansão do caso de uso "excluir histórico".

Quadro 11 - Expansão do caso de uso "excluir histórico"

Caso de uso: Excluir histórico
Descrição: O sistema deve permitir que o usuário exclua o histórico.
Evento iniciador: Necessidade de excluir histórico de estacionamentos.
Atores: Usuário.
Pré-condição: - O usuário deve possuir um perfil; - O usuário deve possuir acesso à internet.
Sequência de eventos: 1. O usuário acessa ao aplicativo; 2. O usuário seleciona a opção "Configurações"; 3. O usuário seleciona o histórico que deseja excluir; 4. O usuário aperta em "excluir histórico de localizações"; 5. O sistema excluir os registros selecionados;
Pós-condições: O histórico completo será apagado.

Fonte: Autoria própria (2024).

No Quadro 12, está a expansão do caso de uso "desativar histórico".

Quadro 12 - Expansão do caso de uso "desativar histórico"

Caso de uso: Desativar histórico	
Descrição: O sistema deve permitir que o usuário desabilite o histórico em sua conta.	
Evento iniciador: Necessidade de desabilitar o histórico de estacionamentos por completo.	
Atores: Usuário.	
Pré-condição: - O usuário deve possuir um perfil; - O usuário deve possuir acesso à internet.	
Sequência de eventos: 1. O usuário acessa ao aplicativo; 2. O usuário entra nas configurações; 3. O usuário desabilita a opção "manter histórico de localizações"; 4. O sistema excluí todo o histórico e para de gerar novos históricos.	
Pós-condições: O usuário exclui completamente seu histórico e o sistema para de registrar novos históricos.	
Extensão	Descrição

Usuário cancela	Neste caso, o sistema irá continuar na tela de configurações, porém o histórico continuará como antes.

Fonte: Autoria própria (2024).

No Quadro 13, está a expansão do caso de uso "Visualizar estacionamento".

Quadro 13 - Expansão do caso de uso "visualizar estacionamento"

Caso de uso: Visualizar estacionamento
Descrição: O sistema deve permitir que o usuário consiga visualizar o estacionamento salvo.
Evento iniciador: Necessidade de visualizar o estacionamento realizado.
Atores: Usuário.
Pré-condição: - O usuário deve possuir ao menos um estacionamento salvo.
Sequência de eventos: 1. O usuário acessa ao aplicativo; 2. O sistema carrega o último estacionamento realizado; 3. O sistema exibe o estacionamento em formato de mapa.
Pós-condições:
O usuário pode ver um mapa com o último estacionamento realizado.

Fonte: Autoria própria (2024).

No Quadro 14, está a expansão do caso de uso "ativar modo de economia de energia".

Quadro 14 - Expansão do caso de uso "ativar modo de economia de energia"

Caso de uso: Ativar modo de economia de energia
Descrição: O sistema deve permitir que o usuário ative o modo de economia de energia, onde o registro do estacionamento é realizado apenas manualmente.
Evento iniciador: Necessidade de prolongar a duração da bateria.
Atores: Usuário.
Pré-condição: - Não há nenhuma pré-condição.

Sequência de eventos:
1. O usuário acessa o aplicativo; 2. O usuário acessa as configurações; 3. O usuário seleciona a opção "Modo economia de energia"; 4. O sistema apresenta uma mensagem informando que neste modo, o registro do estacionamento deve ser efetuado manualmente; 5. O sistema para de ser executado em segundo plano.
Pós-condições:
O aplicativo não funcionará mais em segundo plano, poupando bateria.

Fonte: Autoria própria (2025).

3.6 Implementação das principais funcionalidades

Nesta seção, será abordado a forma que foram implementadas as principais funcionalidades do aplicativo, em forma de Pseudocódigo.

3.6.1 Cadastro de Usuário

O cadastro de usuário é realizado através da autenticação com *Firebase Authentication*. O sistema verifica se todos os campos foram preenchidos e, caso contrário, exibe uma mensagem de erro. Se os dados forem válidos, um novo usuário é criado e as informações são armazenadas no *Firestore*.

A Listagem 1 apresenta o pseudocódigo referente ao cadastro de usuários no sistema. Ele realiza a verificação dos campos obrigatórios, cria um usuário no *Firebase Authentication* e armazena seus dados no *Firestore*. Além disso, trata possíveis erros, como e-mail já cadastrado ou senha fraca.

Listagem 1 – Pseudocódigo do cadastro de usuário

```
1  INÍCIO
2    SE email, senha ou nome estiverem vazios ENTÃO
3      Exibir mensagem "Preencha todos os campos"
4      SAIR
5    TENTE
6      Criar usuário no Firebase Authentication com email e senha fornecidos
7      Atualizar nome do usuário autenticado
8      SE usuário for criado com sucesso ENTÃO
```

```
            Salvar dados no Firestore:
                - Nome
                - Email
                - UID
                - Configurações padrão do usuário
                - Data de cadastro
            Exibir mensagem "Cadastro realizado com sucesso!"
            Retornar à tela anterior

        EXCEÇÃO (erro ao criar usuário)
            SE erro for "email já em uso" ENTÃO
                Exibir mensagem "E-mail já está em uso."
            SE erro for "senha fraca" ENTÃO
                Exibir mensagem "Senha fraca."
    FIM

```

Fonte: Autoria própria (2025).

3.6.2 Login de usuário

O processo de autenticação de usuários no aplicativo é realizado por meio do *Firebase Authentication*, garantindo um login seguro e eficiente. O sistema verifica se todos os campos obrigatórios foram preenchidos e, caso contrário, exibe uma mensagem de erro para o usuário.

Se os dados forem válidos, o sistema autentica o usuário no *Firebase*, recupera suas configurações armazenadas e realiza a sincronização do histórico de estacionamentos, caso essa opção esteja ativada. Além disso, o sistema verifica a opção de manter o histórico, caso esteja desmarcada, irá apagar o histórico local.

Listagem 2 apresenta o pseudocódigo do processo de login, abordando as verificações de segurança, autenticação e sincronização de dados.

Listagem 2 – Pseudocódigo do Processo de Login

```
INÍCIO

        SE email ou senha estiverem vazios ENTÃO
            Exibir mensagem "Preencha todos os campos"
            SAIR

        TENTE
            Autenticar usuário no Firebase Authentication com email e senha fornecidos

            SE autenticação for bem-sucedida ENTÃO
                Exibir mensagem "Login realizado com sucesso!"
                Sincronizar configurações do usuário

                Recuperar preferências do usuário armazenadas localmente
                SE "manter histórico" estiver desativado ENTÃO
                    Excluir todos os registros de estacionamento do banco local
                SENÃO
                    SE "sincronizar histórico automaticamente" estiver ativado ENTÃO
                        Sincronizar registros de estacionamento com o banco de dados remoto

                Retornar à tela anterior
                Executar ação pós-login

        EXCEÇÃO (erro ao autenticar usuário)
            SE erro for "usuário não encontrado" ENTÃO
                Exibir mensagem "Nenhum usuário encontrado com este email."
            SE erro for "senha incorreta" ENTÃO
                Exibir mensagem "Senha incorreta!"
```

```
33                  SENÃO
34                      Exibir mensagem "Erro ao realizar login"
35  FIM
```

Fonte: Autoria própria (2025).

3.6.3 Detecção automática de estacionamento

O aplicativo utiliza os sensores do smartphone (acelerômetro, giroscópio e magnetômetro) para identificar se o veículo está em movimento e detectar automaticamente quando ele foi estacionado. O sistema analisa padrões de desaceleração e alterações significativas nos sensores para registrar a localização do veículo sem necessidade de interação do usuário.

Para garantir um funcionamento preciso em segundo plano, o aplicativo monitora constantemente os sensores e o GPS, permitindo detectar o momento exato em que o veículo parou. Assim que a velocidade do GPS for inferior a 5km/h, o sistema verifica se o dispositivo sofreu uma mudança notável de posição, sugerindo que o usuário se levantou.

Para maior precisão, o algoritmo utiliza tanto o giroscópio quanto o magnetômetro, pois a sensibilidade de apenas um deles pode não ser suficiente para detectar corretamente o movimento. A combinação desses sensores reduz falsos positivos e aumenta a confiabilidade da detecção automática de estacionamento.

Além disso, o sistema verifica se o usuário dirigiu no último minuto, evitando registros incorretos caso ele apenas se levante de uma cadeira sem estar dentro de um veículo.

A Listagem 3 apresenta o pseudocódigo da verificação automática de direção do veículo. Ele utiliza sensores do smartphone (GPS, acelerômetro, giroscópio e magnetômetro) para determinar se o usuário está dirigindo. Caso a movimentação seja interrompida e a condição de estacionamento seja atendida, a função dispara o salvamento automático da localização.

Listagem 3 – Pseudocódigo da detecção automática de estacionamento

```
1   INÍCIO
2     TENTE
```

```
    Obter posição atual do usuário com alta precisão (tempo limite de 5
segundos)
  EXCEÇÃO (falha ao obter posição)
    Utilizar última posição conhecida

  SE posição for inválida ENTÃO
    Definir posição como a última posição válida

  Definir "dirigindo" como verdadeiro SE velocidade da posição for maior que
5 km/h
  SE modo de simulação estiver ativado ENTÃO
    Definir "dirigindo" como verdadeiro

  SE usuário não estiver dirigindo e tenha dirigido no último minuto ENTÃO
    Se passou mais de 1 minuto sem dirigir, definir "dirigiuNoUltimoMinuto"
como falso

  SE sensores (acelerômetro, giroscópio ou magnetômetro) indicarem
movimentação e usuário parou de dirigir ENTÃO
    Resetar sensores
    Chamar função de salvamento automático do estacionamento

  SE usuário estiver dirigindo ENTÃO
    Atualizar registro de que dirigiu recentemente
    Resetar variáveis com valores dos sensores
FIM
```

Fonte: Autoria própria (2025).

3.6.4 Salvamento automático do estacionamento

O salvamento automático da localização do estacionamento ocorre ao detectar que o usuário estacionou e se levantou do veículo.

O sistema verifica se o veículo foi estacionado recentemente para evitar salvamentos duplicados em um curto período. Caso a posição seja válida, a

localização é armazenada no banco de dados local (*SQLite*) e, se o usuário estiver autenticado e as configurações permitirem, os dados também são sincronizados com o *Firebase Firestore*.

Além disso, uma notificação é exibida para informar o usuário de que a posição do estacionamento foi salva com sucesso.

A Listagem 4 apresenta o pseudocódigo do salvamento automático da localização do estacionamento, incluindo as verificações de tempo, o armazenamento local e a sincronização com a nuvem.

Listagem 4 – Pseudocódigo do salvamento automático do estacionamento

```
1  INÍCIO
2      Obter data e hora atuais
3
4      SE último salvamento foi realizado há menos de 2 minutos ENTÃO
5          SAIR (evitar salvamento duplicado)
6
7      SE posição for inválida ENTÃO
8          SAIR
9
10     Atualizar data e hora do último salvamento
11
12     Obter instância do banco de dados local
13     Carregar preferências do usuário
14
15     SE "manter histórico" estiver desativado ENTÃO
16         Excluir todas as localizações armazenadas no banco local
17
18     Salvar nova localização no banco SQLite com:
19         - Latitude
20         - Longitude
21         - Data e hora atual
22         - Caminho da imagem (vazio para salvamento automático)
23
```

```
    Exibir notificação informando que o estacionamento foi salvo
automaticamente

    SE usuário estiver autenticado E preferências permitirem sincronização E
"manter histórico" estiver ativo ENTÃO
        Salvar localização no Firebase Firestore com:
            - Latitude
            - Longitude
            - Data e hora
            - Caminho da imagem (vazio)
FIM
```

Fonte: Autoria própria (2025).

3.6.5 Salvamento manual do estacionamento

A função de salvamento manual do estacionamento foi projetada para ser simples, mas eficiente, permitindo ao usuário registrar a posição do veículo sem depender da detecção automática. O fluxo segue os seguintes passos:

Validação da posição atual – O sistema verifica se a localização é válida. Caso contrário, exibe uma mensagem de erro informando que a posição não foi encontrada;

Solicitação de captura de foto – Se a posição for válida, o aplicativo pergunta ao usuário se deseja tirar uma foto do local de estacionamento;

Captura e armazenamento da foto – Caso o usuário aceite, a câmera é acionada e a imagem é salva localmente. A URL ou o caminho do arquivo da foto é armazenado para acesso futuro;

Registro dos dados localmente – Independentemente do login, o aplicativo salva os seguintes dados no banco de dados local (*SQLite*): latitude, longitude, data e hora do registro e o caminho da imagem (se houver);

Sincronização com a nuvem – Se o usuário estiver autenticado, os dados do estacionamento são enviados para o banco de dados na nuvem (*Firebase Firestore*).

A Listagem 5 apresenta o pseudocódigo do salvamento manual da localização do estacionamento. Ele obtém a posição atual do usuário, solicita a

captura de uma imagem do local (opcional), salva os dados no banco local (*SQLite*) e, se o usuário estiver autenticado, também realiza a sincronização com o *Firebase*.

Listagem 5 – Pseudocódigo do salvamento manual do estacionamento

```
INÍCIO
    Carregar localização atual do usuário
    SE latitude e longitude forem inválidas (0.0, 0.0) ENTÃO
        Exibir mensagem "Localização não encontrada"
        SAIR

    Definir variável "urlImagemLocal" como vazia
    Solicitar ao usuário se deseja salvar uma imagem do local
    SE usuário optar por salvar imagem ENTÃO
        Capturar imagem
        SE imagem capturada for válida ENTÃO
            Definir caminho da imagem e armazená-la no dispositivo
            Atualizar "urlImagemLocal" com o caminho da imagem armazenada

    Carregar preferências do usuário
    SE "manter histórico" estiver desativado ENTÃO
        Excluir todas as localizações armazenadas no banco local

    Salvar localização no banco SQLite com:
        - Latitude
        - Longitude
        - Data e hora atual
        - Caminho da imagem armazenada (se houver)

    Atualizar marcadores no mapa para refletir a nova posição salva

    SE usuário estiver autenticado E preferências permitirem sincronização
ENTÃO
        Definir "salvando" como verdadeiro
        SE imagem foi capturada ENTÃO
```

```
31          Salvar imagem no Firebase e armazenar URL
32      Salvar localização no Firebase Firestore com:
33          - Latitude
34          - Longitude
35          - Data e hora
36          - URL da imagem (se houver)
37      Definir "salvando" como falso
38      Exibir mensagem "Localização do estacionamento salva"
39
40  Carregar último estacionamento salvo
41
42  FIM
```

Fonte: Autoria própria (2025).

3.6.6 Exibição do Histórico de Estacionamentos

O histórico de estacionamentos é recuperado do banco de dados local (*SQLite*), permitindo que os registros sejam acessados mesmo sem conexão com a internet.

Após a recuperação dos dados, cada registro de estacionamento é exibido individualmente em um mapa interativo, permitindo que o usuário visualize sua localização com precisão. Além disso, os registros são formatados com data e hora ajustadas para facilitar a leitura.

A Listagem 6 apresenta o pseudocódigo do carregamento do histórico de estacionamentos. Ele obtém os registros armazenados no banco de dados *SQLite*, processa e formata as datas para exibição, ordena os registros do mais recente para o mais antigo e atualiza a interface do usuário.

Listagem 6 – Pseudocódigo do Carregamento do Histórico de Estacionamentos

```
1  INÍCIO
2
3      Obter instância do banco de dados local
4      Recuperar todas as localizações armazenadas
5
```

```
6     Limpar variável com lista do histórico de estacionamentos
7
8     Definir "_carregando" como verdadeiro para indicar carregamento
9
10    PARA CADA registro em estacionamentos FAÇA
11        Criar um novo objeto contendo:
12            - Latitude
13            - Longitude
14            - Data e hora original
15            - Data e hora formatada (DD/MM/AAAA HH:MM)
16        Adicionar objeto à lista de histórico de estacionamentos
17
18    Ordenar lista de histórico por data, do mais recente para o mais antigo
19
20    Definir "_carregando" como falso para indicar fim do carregamento
21
22  FIM
```

Fonte: Autoria própria (2025).

4 RESULTADOS

Esse capítulo irá apresentar os resultados relacionados a este trabalho, que é o desenvolvimento de um aplicativo para marcar onde o veículo do usuário foi estacionado de forma automatizada.

4.1 Interfaces do aplicativo

Nesta seção, serão apresentadas as interfaces principais do aplicativo proposto.

4.1.1 Tela principal

A tela principal do aplicativo apresenta um mapa interativo, onde o usuário pode visualizar sua localização atual e o último estacionamento salvo. Para facilitar a navegação, o mapa contém dois marcadores: um vermelho, indicando a posição atual do usuário, e um verde, representando o local onde o veículo foi estacionado.

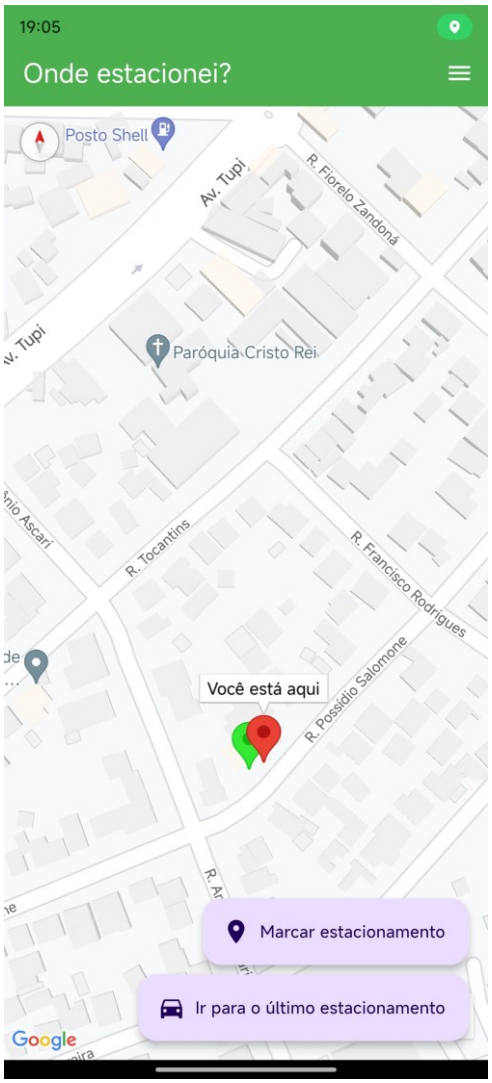
No canto inferior direito, há dois botões de ação rápida:

- Botão “Marcar estacionamento”: Permite marcar manualmente um novo estacionamento.

- Botão “Ir para o último estacionamento”: Direciona o usuário até a localização do último estacionamento salvo.

No canto superior, há um ícone de menu que permite acessar configurações e histórico de estacionamentos. A Figura 2 apresenta a interface principal do aplicativo.

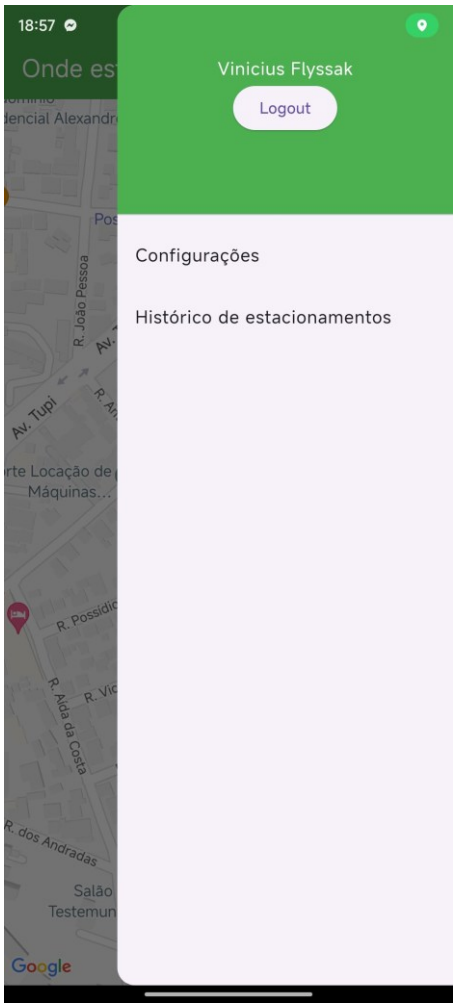
Figura 2 – Tela principal



Fonte: Autoria própria (2025).

No canto superior, existe um ícone que abre um *drawer*, onde é possível realizar login / logout, visualizar o histórico de estacionamentos e acessar as configurações. Na Figura 3, é possível visualizar o *drawer*.

Figura 3 – Tela principal com *drawer* aberto



Fonte: Autoria própria (2025).

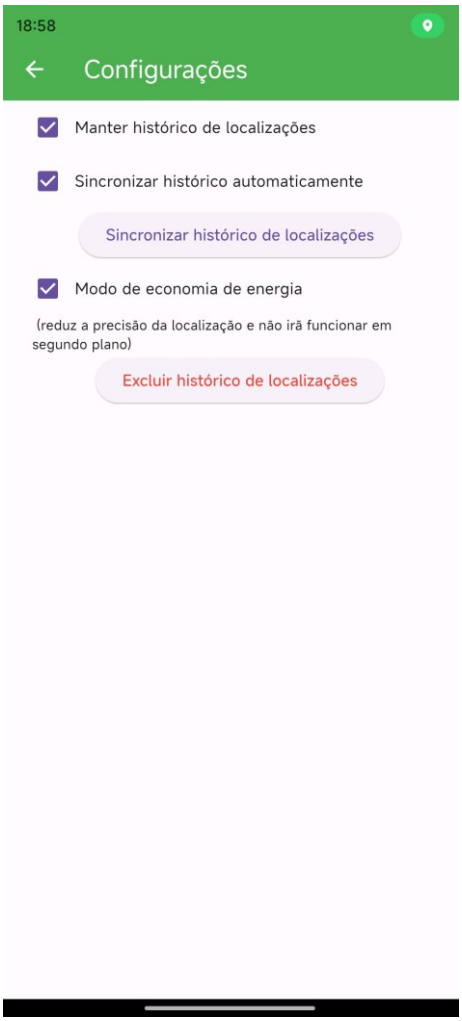
4.1.2 Tela de configurações

A tela de configurações permite ao usuário personalizar as preferências do aplicativo, tornando sua experiência mais adaptável às necessidades do dia a dia. Nesta tela, o usuário pode:

- Ativar ou desativar o armazenamento do histórico de estacionamentos.
- Sincronizar manualmente os estacionamentos salvos com a nuvem.
- Ativar o modo de economia de energia, onde o registro do estacionamento deve ser feito manualmente, reduzindo o consumo da bateria.
- Excluir completamente o histórico de estacionamentos, garantindo maior privacidade.

A Figura 4 apresenta a interface da tela de configurações do aplicativo.

Figura 4 – Tela de configurações

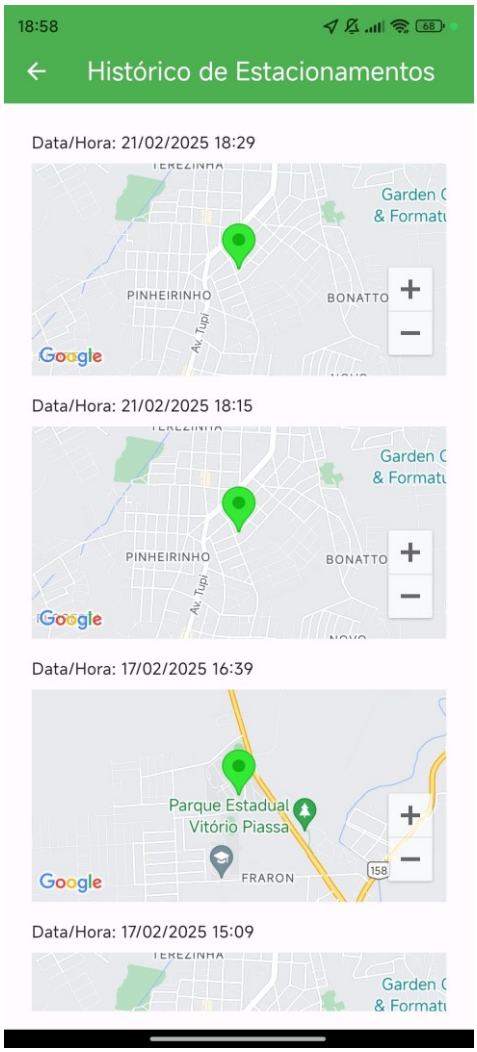


Fonte: Autoria própria (2025).

4.1.3 Tela de histórico de estacionamentos

Na tela de histórico, é possível visualizar o histórico completo de estacionamentos, onde mostra a localização em um pequeno mapa com uma marcação em verde, exibindo acima do mapa a data e hora onde o estacionamento foi realizado. No mapa, existem dois atalhos principais, onde um abre o Google Maps e traça uma rota e o outro apenas abre a localização no Google Maps. A tela pode ser vista na Figura 5.

Figura 5 – Tela de histórico de estacionamentos



Fonte: Autoria própria (2025).

4.1.4 Tela de login

Nas imagens apresentadas anteriormente, o usuário já estava logado, mas em casos em que precise se autenticar, existe um botão que leva a tela de login. Nesta tela, existem dois campos, um para informar o e-mail e outro para informar a senha. A tela pode ser vista na Figura 6.

Figura 6 – Tela de *login*

Fonte: Autoria própria (2025).

4.1.5 Tela de cadastro

Em casos em que o usuário ainda não possui acesso, é possível realizar o cadastro, através da opção “Cadastre-se”. Na tela de cadastro, existem 3 campos, sendo respectivamente, um campo para informar o nome, um para informar o e-mail e por fim, um campo para informar a senha. A tela de cadastro pode ser vista na Figura 7.

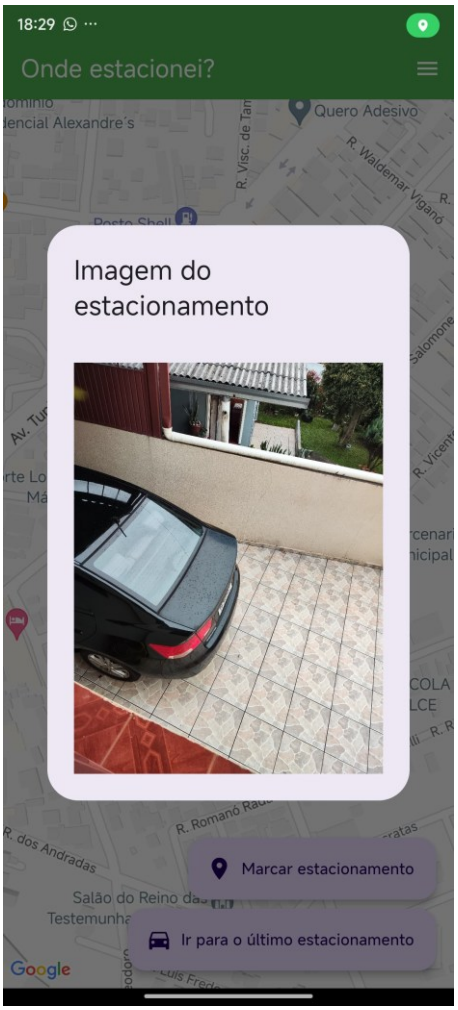
Figura 7 – Tela de cadastro

Fonte: Autoria própria (2025).

4.1.6 Tela de imagem do estacionamento

Caso o usuário salve um estacionamento manualmente, ele pode optar por adicionar uma imagem do local. Se essa opção for selecionada, a foto ficará vinculada ao registro do estacionamento e poderá ser visualizada posteriormente. A Figura 8 apresenta a tela de exibição da imagem associada ao estacionamento.

Figura 8 – Tela de imagem do estacionamento

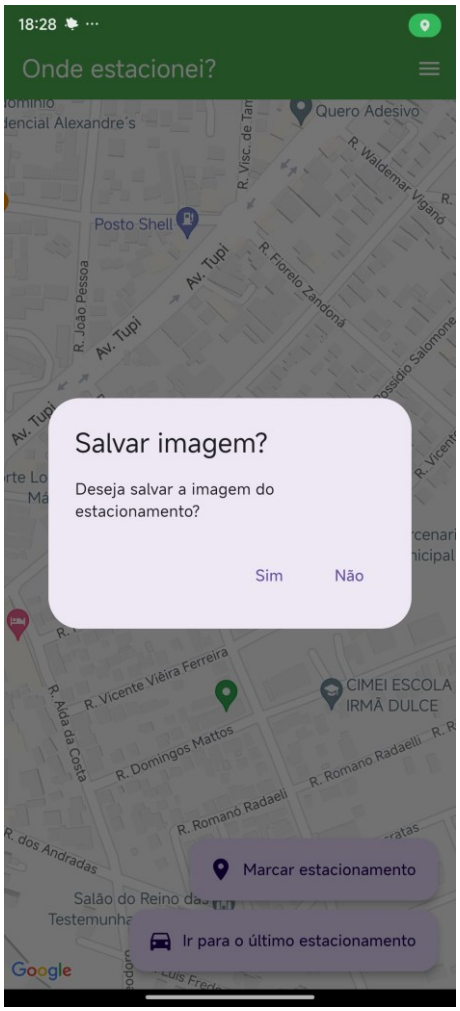


Fonte: Autoria própria (2025).

4.1.7 Tela de solicitação de captura de imagem

Ao salvar um estacionamento manualmente, o sistema exibe uma solicitação perguntando se o usuário deseja capturar uma imagem do local. Essa funcionalidade permite que o usuário registre visualmente o ambiente para facilitar a localização posterior. A Figura 9 ilustra a tela de solicitação de captura de imagem.

Figura 9 – Tela de solicitação de captura e imagem

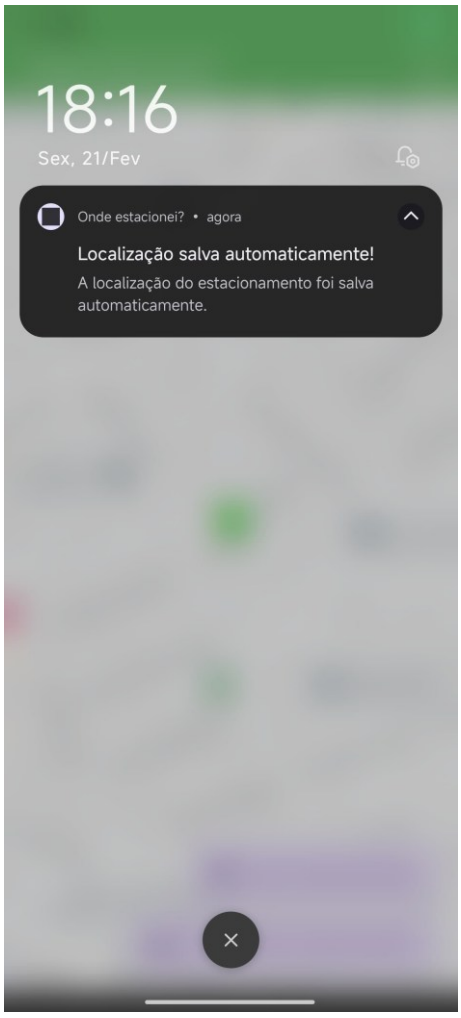


Fonte: Autoria própria (2025).

4.1.8 Notificação ao salvar estacionamento de forma automática

Quando o aplicativo detecta automaticamente que o veículo foi estacionado e salva a localização sem intervenção do usuário, uma notificação informativa é exibida. Essa notificação confirma o registro do estacionamento. A Figura 10 apresenta a notificação emitida pelo sistema.

Figura 10 – Notificação de estacionamento salvo automaticamente



Fonte: Autoria própria (2025).

4.2 Resultados dos Testes de Precisão e Consumo de Bateria

Esta seção apresenta os resultados dos testes realizados para avaliar a precisão do sistema de detecção de estacionamento e o consumo de bateria do aplicativo.

4.2.1 Teste de precisão de marcação de estacionamentos de forma automática

Os testes de precisão foram realizados em estacionamentos abertos, utilizando o dispositivo Xiaomi 12T Pro. Durante os testes, o smartphone foi mantido no bolso, simulando condições reais de uso.

Em 10 testes realizados, o sistema registrou corretamente a posição do veículo em todos os casos, com apenas uma ocorrência de marcação duplicada

durante o quarto teste. Esse problema não foi reproduzido em tentativas subsequentes, indicando uma estabilidade aprimorada na detecção automática com a versão mais recente do aplicativo.

A precisão obtida foi considerada alta e acima das expectativas iniciais, demonstrando a eficácia do sistema para registrar a localização de estacionamento em ambientes abertos.

4.2.2 Teste de consumo de energia

Para avaliar o consumo de energia, os testes foram realizados utilizando o Xiaomi 12T Pro, com as seguintes condições:

- Duração do teste: 30 minutos
- Atividade: 10 minutos com a tela ligada e o aplicativo em primeiro plano, seguidos de 20 minutos com a tela desligada e o aplicativo em segundo plano
- Nível de brilho da tela: Aproximadamente 75%
- Nível de bateria inicial: 98%
- Nível de bateria final: 95%

O consumo total foi de 3% em 30 minutos, o que representa uma estimativa de 6% a 8% por hora, valor considerado aceitável para aplicativos que utilizam geolocalização e sensores em tempo real.

5 CONCLUSÃO

O presente trabalho teve como objetivo o desenvolvimento de um aplicativo móvel baseado em geolocalização para auxiliar os usuários na localização de veículos estacionados, eliminando a necessidade de intervenção manual. Para isso, o sistema foi projetado para detectar automaticamente quando o veículo foi estacionado, utilizando os sensores internos do *smartphone*, como GPS, acelerômetro e giroscópio, garantindo a precisão na marcação da localização.

A implementação do aplicativo foi realizada utilizando o *framework Flutter*, o que permitiu o desenvolvimento multiplataforma com um único código-base, assegurando a compatibilidade inicial com dispositivos Android e possibilitando futuramente a implementação para iOS. A integração com o *Firebase* proporcionou um ambiente seguro e escalável para autenticação de usuários e armazenamento de dados em nuvem, permitindo que os usuários sincronizem seu histórico de estacionamentos entre diferentes dispositivos. Além disso, a utilização da Google Maps API permitiu a visualização interativa da posição do veículo estacionado, otimizando a experiência do usuário.

A validação do sistema foi realizada por meio de testes experimentais conduzidos em diferentes dispositivos móveis e cenários urbanos. O aplicativo foi testado em um Samsung Galaxy S20 FE e um Xiaomi 12T Pro, avaliando a compatibilidade da solução com diferentes fabricantes e modelos de *smartphones*. Os resultados demonstraram que o sistema, em sua versão final, registrou corretamente a localização do estacionamento em todos os testes, com apenas um caso de marcação duplicada, evidenciando um alto nível de precisão.

Durante os testes, foi observado que a precisão do GPS pode ser comprometida em ambientes fechados, o que representa um desafio a ser aprimorado em versões futuras. O consumo de bateria foi avaliado e os resultados indicaram um gasto de 6% a 8% por hora, considerado aceitável para um aplicativo que utiliza geolocalização e sensores em tempo real.

O desenvolvimento do projeto enfrentou desafios, como a integração eficiente dos sensores do dispositivo e a otimização da detecção do estacionamento. A necessidade de calibrar adequadamente os parâmetros do acelerômetro, giroscópio e magnetômetro para minimizar falsos positivos e garantir uma experiência mais confiável para o usuário foi um dos pontos aprimorados ao longo do desenvolvimento.

Este estudo contribuiu para a área de desenvolvimento de aplicativos móveis, demonstrando a viabilidade da utilização de sensores de *smartphones* para automatizar tarefas cotidianas, como o registro automático da posição do veículo estacionado. A solução desenvolvida elimina a necessidade de intervenção manual, tornando o processo mais simples.

Dentre as possíveis melhorias para trabalhos futuros, destacam-se:

- Expansão para a plataforma iOS, garantindo compatibilidade entre sistemas operacionais;

- Aprimoramento da precisão em ambientes fechados, empregando técnicas de fusão de sensores para minimizar a dependência exclusiva do GPS;

- Implementação de inteligência artificial para identificar padrões de estacionamento, visando maior precisão.

Dessa forma, conclui-se que o sistema desenvolvido é funcional e viável, apresentando um impacto positivo para os usuários. No entanto, melhorias podem ser implementadas para tornar o aplicativo ainda mais preciso e otimizado, consolidando-o como uma ferramenta no auxílio à localização de veículos estacionados.

REFERÊNCIAS

ALBERTO, Matheus. Flutter: o que é e tudo sobre o framework. 2023. Disponível em: https://www.alura.com.br/artigos/flutter. Acesso em: 27 mar. 2024.

ALBUQUERQUE, Flávia. Venda de veículos cresce 12% em 2023, diz balanço da Fenabrave. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2024-01/venda-de-veiculos-cresce-12-em-2023-diz-balanco-da-fenabrave. Acesso em: 01 ago. 2024.

BARCELOS, Naiche Amantino. A study on geolocalization through GSM systems. 2017. Acesso em: 03 mai. 2024.

DEVMEDIA. Introdução à Google Maps API. 2013. Disponível em: https://www.devmedia.com.br/introducao-a-google-maps-api/26967. Acesso em: 10 abr. 2024.

FRANDOLOSO, Vinícius Rodrigues. Análise de dados para identificação de atividades de estacionamento de veículos por meio de aprendizado de máquina. 2023. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/31678. Acesso em: 23 mar. 2024.

GENTILE, C. *et al*. Geolocation techniques: principles and applications. [S. l.]: Springer Science & Business Media, 2012. Acesso em: 03 mai. 2024.

GOOGLE. Cloud Firestore. 2024. Disponível em: https://firebase.google.com/docs/firestore. Acesso em: 10 dez. 2024.

GOOGLE. Documentação da Plataforma Google Maps. 2024. Disponível em: https://developers.google.com/maps/documentation?hl=pt-br. Acesso em: 15 dez. 2024.

GOOGLE. Flutter Documentation. 2024. Disponível em: https://flutter.dev/docs. Acesso em: 15 dez. 2024.

LONDRINA, RPC. Motorista esquece onde estacionou carro e registra Boletim de Ocorrência por furto de veículo, em Apucarana. 2021. Disponível em: https://g1.globo.com/pr/norte-noroeste/noticia/2021/05/04/motorista-esquece-onde-estacionou-carro-em-apucarana-e-registra-boletim-de-ocorrencia-por-furto-de-veiculo.ghtml. Acesso em: 05 abr. 2024.

MACIULEVICIUS, Paula. Tenha paciência com quem costuma esquecer onde estacionou o carro - CREDITO: CAMPO GRANDE NEWS. 2015. Disponível em: https://www.campograndenews.com.br/lado-b/comportamento-23-08-2011-08/tenha-paciencia-com-quem-costuma-esquecer-onde-estacionou-o-carro. Acesso em: 28 mar. 2024.

NIELD, David. Conheça todos os sensores do seu smartphone e como eles funcionam. 2017. Disponível em: https://gizmodo.uol.com.br/sensores-smartphones-guia/. Acesso em: 22 abr. 2024.

OFICIAL, CNB Cascavel. Motorista esquece onde estaciona o carro e registra boletim de ocorrência. 2022. Disponível em: https://cbncascaveloficial.com.br/noticia/motorista-esquece-onde-estaciona-o-carro-e-registra-boletim-de-ocorrencia. Acesso em: 28 mar. 2024.

PONTOTEL. Metodologia Scrum: entenda o conceito, e veja como ela pode auxiliar na gestão de projetos! 2024. Disponível em: https://www.pontotel.com.br/metodologia-scrum/#:~:text=A%20Metodologia%20Scrum%20é%20considerada,os%20recursos%20humanos%20e%20materiais. Acesso em: 16 jun. 2024.

RIBAS, Thomaz. Scrum: O que É, Como Funciona e Exemplos Práticos [GUIA]. Disponível em: https://thomazribas.com/blog/scrum. Acesso em: 01 ago. 2024.

SANTOS, Adby. Empresas que usam FLUTTER em 2022 e você não sabia - Parte 1. 2022. Disponível em: https://www.folhape.com.br/colunistas/tecnologia-e-games/empresas-que-usam-flutter-em-2022-e-voce-nao-sabia-parte-1/29540/. Acesso em: 16 jun. 2024.

Techtudo. Como marcar e encontrar o lugar que você estacionou pelo Waze. 2017. Disponível em: https://www.techtudo.com.br/dicas-e-tutoriais/2017/04/como-marcar-e-encontrar-o-lugar-que-voce-estacionou-pelo-waze.ghtml. Acesso em: 01 jun. 2024.

Techtudo. Economize tempo com o aplicativo Find My Car. Disponível em: https://www.techtudo.com.br/tudo-sobre/find-my-car/. Acesso em: 01 jun. 2024.