UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA

INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL - IEDS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

APLICAÇÃO MÓVEL DE APOIO À MOBILIDADE URBANA: USO DO ALGORITMO KNN PARA SUGERIR POSTOS DE COMBUSTÍVEIS MAIS PRÓXIMO COM MENOR CONGESTIONAMENTO.

Trabalho de Conclusão de Curso

JORGE WILSON MUNGANJA

Redenção

2025

UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA

INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL - IEDS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JORGE WILSON MUNGANJA

APLICAÇÃO MÓVEL DE APOIO À MOBILIDADE URBANA: USO DO ALGORITMO KNN PARA SUGERIR POSTOS DE COMBUSTÍVEIS MAIS PRÓXIMO COM MENOR CONGESTIONAMENTO.

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Engenharia de Computação do Instituto de Engenharia e Desenvolvimento Sustentável, Ceará - CE, Campus Auroras, como requisito à obtenção do título de Engenheiro de Computação.

Orientador(a): Prof. Dr. Sabi Yari Moïse BANDIRI

Redenção

2025

Universidade da Integração Internacional da Lusofonia Afro-Brasileira
Sistema de Bibliotecas da UNILAB
Catalogação de Publicação na Fonte.

Munganja, Jorge Wilson.

M966a

Aplicação móvel de apoio à mobilidade urbana: uso do algoritmo KNN para sugerir postos de combustíveis mais próximo com menor congestionamento / Jorge Wilson Munganja. - Redenção, 2025.
107f: il.

Monografia - Curso de Engenharia De Computação, Instituto De Engenharias E Desenvolvimento Sustentável, Universidade da Integração Internacional da Lusofonia Afro-Brasileira, Redenção, 2025.

Orientador: Prof. Dr. Sabi Yari Moïse BANDIRI.

1. Mobilidade urbana. 2. Aplicação móvel. 3. Algoritmo K-Nearest Neighbors (KNN). I. Título

CE/UF/BSCA CDD 388.4

JORGE WILSON MUNGANJA

APLICAÇÃO MÓVEL DE APOIO À MOBILIDADE URBANA: USO DO ALGORITMO KNN PARA SUGERIR POSTOS DE COMBUSTÍVEIS MAIS PRÓXIMO COM MENOR CONGESTIONAMENTO.

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Engenharia de Computação do Instituto de Engenharia e Desenvolvimento Sustentável, Ceará - CE, Campus Auroras, como requisito à obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Redenção, xx de novembro de 2025:

Documento assinado digitalmente
gov.br
SABI YARI MOISE BANDIRI
Data: 07/12/2025 16:15:05-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Sabi Yari Moïse BANDIRI
Orientador
Universidade da Integração Internacional da Lusofonia Afro-Brasileira - UNILAB

Documento assinado digitalmente
gov.br
CICERO SARAIVA SOBRINHO
Data: 05/12/2025 17:26:32-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cicero Saraiva Sobrinho
Avaliador
Universidade da Integração Internacional da Lusofonia Afro-Brasileira - UNILAB

Documento assinado digitalmente
gov.br
MILENE VIEIRA FIGUEIRA
Data: 06/12/2025 08:13:40-0300
Verifique em https://validar.iti.gov.br

Prof. Msc. Milene Vieira Figueira
Avaliadora
Universidade Federal Rural de Pernambuco - UFRPE

Redenção
2025

Este trabalho é dedicado à minha mãe, Domingas Glória Agostinho, por tudo o que fez para que eu me tornasse o homem que sou hoje, e ao meu irmão Amílcar Alexandre dos Anjos Agostinho Baptista pois, mesmo sendo menor de mim, honra com as suas palavras de cuidar e salvaguardar a nossa Mãe na minha ausência com um amor incondicional baseado em atitudes.

Dedico, ainda, este trabalho a todos os homens e mulheres que integram a comunidade científica, os quais vivem desafiando seus próprios limites com o propósito de contribuir para o desenvolvimento das ciências sociais, exatas e tecnológicas através do universo acadêmico e não só.

Agradecimentos

Primeiramente, gratidão a DEUS, por toda sabedoria e sustento que me concedeu ao longo desta jornada acadêmica, honra e profunda gratidão as minhas mães, Domingas Glória Agostinho, Ana Leonia Ngonga e seu esposo Benjamim Celestino Saimina (que Deus o tenha), por serem a minha maior referência de humildade, respeito, força, foco, fé em Deus, fé em mim mesmo e por serem o melhor exemplo de união para mim.

Ao Prof. Dr. Sabi Yari Moïse BANDIRI, declaro minha sincera gratidão pois agiu como Pai antes mesmo de agir como Orientador, por não desistir em meio as minhas falhas enquanto orientando, por sua majestosa orientação, por ser sempre caloroso e respeito à todo momento e também pelos conhecimentos compartilhados, que foram fundamentais não só para a realização deste trabalho, como também para o meu próprio crescimento acadêmico, pessoal e profissional.

Agradecimentos aos meus irmãos Amílcar dos Anjos Agostinho (“Alex”), Acácio Benjamim Celestino “ABC”, Emanuela Saimina “Nina” e Alceno Benjamim Saimina “Djidji”, por se tornarem as minhas maiores fontes de inspiração acadêmica e de verdadeiro amor e união. Vós sois a razão pela qual não sei quando parar e não pretendo em algum momento parar. Não deixo de agradecer também os meus tios Alberto Neves Calunga, Mirlles Luneta “Tio Mirillas”, Marenus Luneta “Marenus Priceless” e Nelma Luneta, pelo apoio constante e pela disponibilidade sempre que precisei. Aos meus avós, Guilherme Bernardo Luneta e José Garcia Wazanga, pelo suporte e incentivo que deram à minha ambição de concluir a formação superior acadêmica fora do meu país.

Agradeço ao Engº Adilson Leonardo Morgado Cabaça por não deixar de ser amigo e irmão ao logo da nossa formação, por todo apoio dado desde a minha exclusão no programa de assistência estudantil, por ser praticamente o meu Coorientador da parte teórica "Conceitos"deste trabalho e ao Engº Milton Lucas Kakupa amigo e irmão desde o ensino médio até nos dias contemporâneos e por ser o meu segundo Coorientador da parte prática "Programação"deste trabalho de conclusão de curso.

Expresso meus agradecimentos Neusa Vunge, Dorcas Domingos e Crismura Chiteculo pelo incentivo, hospitalidade e respeito que me concederem, e claramente, agradeço especialmente a Verónica Bembo Bari, que sacrificou do seu tempo de manuseio do seu próprio computador, para disponibilizá-lo para mim numa ótica que permitisse eu finalizar a parte prática deste projeto. E, por fim — mas não menos importante —, quero agradecer a mim mesmo, por não me limitar, não desistir e não mudar de curso, independentemente das falhas, dificuldades, perdas, decepções, descrenças e turbulência.

"Não temas, porque eu sou contigo; não te assombres, porque eu sou o teu DEUS; eu te fortaleço, e te ajudo e te sustento com a minha destra fiel."
(Isaías: 41-10)

Resumo

A implementação de novas urbanizações ou o desenvolvimento das urbanizações já existentes culmina sempre no aumento de mais frotas de veículos, no distanciamento das instituições de prestação de serviços público ou privado para a população, fatores esses que obrigam os governos e instituições privadas fornecerem serviços de transportes a população, ou também o cidadão adquirir seu próprio meio de transporte motorizado e disso, temos alguns detalhes que têm intensificado os problemas da mobilidade urbana, especialmente nos grandes centros, onde o tráfego intenso torna-se parte da rotina diária. Um dos principais desafios enfrentados pelos motoristas consiste em localizar postos de combustíveis próximos à sua posição atual, garantindo ao mesmo tempo a escolha de rotas que ofereçam menor nível de congestionamento em tempo real. Essa dificuldade evidencia a necessidade de soluções tecnológicas capazes de integrar informações de geolocalização com dados de tráfego dinâmicos, proporcionando maior eficiência, economia de tempo e segurança na tomada de decisão durante os deslocamentos. Neste contexto o presente Trabalho de Conclusão de Curso propõe o desenvolvimento de um aplicativo móvel de apoio à mobilidade urbana, capaz de recomendar ao usuário o posto de combustível mais próximo com menor congestionamento nas vias para o seu acesso, utilizando o algorítimo de aprendizado de máquina K-Nearest Neighbors (KNN). A aplicação móvel será projetada para coletar dados de localização via GPS, por meio de APIs que integram informações de trânsito em tempo real por intermédio de mapas e aplicar o KNN para classificar os postos de combustíveis disponíveis, considerando critérios como distância geográfica e nível de congestionamento e a recomendação ao usuário será baseada não apenas na proximidade física, porém, na viabilidade do trajeto em que obtém-se melhor fluidez de trânsito.

Palavra-chave: Aplicação Móvel; Mobilidade Urbana; Localização de Posto de Combustível; Algoritmo KNN.

Abstract

The implementation of new urban developments, or the expansion of existing ones, invariably results in an increase in vehicle fleets and in the distancing of public and private service institutions from the population. These factors compel governments and private institutions to provide transportation services to citizens, or alternatively, lead individuals to acquire their own motorized means of transport. Consequently, several aspects have intensified urban mobility problems, particularly in large metropolitan areas, where heavy traffic has become part of daily routine. One of the main challenges faced by drivers is locating fuel stations near their current position while simultaneously ensuring the selection of routes with lower levels of real-time congestion. This difficulty highlights the need for technological solutions capable of integrating geolocation data with dynamic traffic information, thereby providing greater efficiency, time savings, and safety in decision-making during travel. Within this context, the present Final Graduation Project proposes the development of a mobile application to support urban mobility, designed to recommend to users the nearest fuel station with the least congested access routes, employing the K-Nearest Neighbors (KNN) machine learning algorithm. The mobile application will be designed to collect location data via GPS, through APIs that integrate real-time traffic information using maps, and apply KNN to classify the available fuel stations, considering criteria such as geographic distance and congestion levels. The recommendation provided to the user will be based not only on physical proximity but also on the feasibility of the route offering better traffic flow.

Keywords: : Mobile Application; Urban Mobility; Fuel Station Location; K-Nearest Neighbors (KNN) Algorithm.

Lista de ilustrações

Figura 1 – Censo da População Brasileira a Viver em Áreas Urbanas e Rurais 27
Figura 2 – Grau de Urbanização 28
Figura 3 – Arquitetura do Android 35
Figura 4 – Arquitetura do iOS 36
Figura 5 – Comparativo entre *Front-end* e *Back-end*. 37
Figura 6 – Funcionamento de uma API. 38
Figura 7 – Funcionamento de uma API REST. 39
Figura 8 – SOAP Proxy Server. 40
Figura 9 – Arquitetura GraphQL API. 41
Figura 10 – Fluxograma do algoritmo KNN 44
Figura 11 – Representação Gráfica do Grafo. 45
Figura 12 – APIdog Usa-se Para Testar a API do Aplicativo 59
Figura 13 – Arquiterura Ionic. 61
Figura 14 – Modelo do Funcionamento do KNN na Aplicação. 63
Figura 15 – Fases do Funcionamento do KNN na Aplicação. 63
Figura 16 – Diagrama de Uso Requisitos Funcionais 71
Figura 17 – Diagrama de Uso Requisitos Não Funcionais 72
Figura 18 – Gráfico Relacionado Sexo dos Participantes 76
Figura 19 – Gráfico Relacionado a Idade dos Participantes 76
Figura 20 – Gráfico Relacionado a Frequência de Dirigir 76
Figura 21 – Gráfico Relacionado ao Interesse da Aplicação 77
Figura 22 – Gráfico Relacionado ao Funcionamento da Aplicação em Diferentes Locais . 78
Figura 23 – Gráfico Relacionado ao Funcionamento da Aplicação em Diferentes Locais . 78
Figura 24 – Gráfico Relacionado a Funcionalidade da Aplicação 80
Figura 25 – Gráfico Relacionado a Funcionalidade da Aplicação 80
Figura 26 – Gráfico Relacionado a Funcionalidade da Aplicação 82
Figura 27 – Comentário e Sugestões dos Usuários 83
Figura 28 – Comentário e Sugestões dos Usuários 84
Figura 29 – Tela Inicial da Aplicação 85
Figura 30 – Tela tela de Busca daAplicação 86
Figura 31 – Tela de Processamento de Dados da Aplicação 87
Figura 32 – Localizações dos Postos de Combustíveis Sugeridos pela Aplicação. 88
Figura 33 – Tela dos Resultados e Sugestão do Postos 89
Figura 34 – Tela dos Resultados e Sugestão do Postos 90

Lista de tabelas

Tabela 1 – Funções de cada Camada do iOS, (Fonte: Elaborada pelo Autor, 2025.) . . . 36
Tabela 2 – Comparativo entre Flutter, React Native e Kotlin para Desenvolvimento Mobile, (Fonte: Elaborada pelo Autor, 2015). 37
Tabela 3 – Conceitos de Back-end e Integração de APIs, (Fonte: Elaborada pelo Autor, 2015) 41
Tabela 4 – Uso do CSS em Diferentes Tipos de Aplicativos Móveis, (Fonte: Elaborada pelo Autor, 2015). 49
Tabela 5 – Principais Recursos do HTML5 Aplicados à Aplicação Móvel, (Fonte: Elaborada pelo Autor, 2015). 50
Tabela 6 – Fluxo de funcionamento do Aplicativo com Django, (Fonte: Elaborada pelo Autor, 2015). 52
Tabela 7 – Contribuição do Python na Aplicação, (Fonte: Elaborada pelo Autor, 2015). 53
Tabela 8 – Abordagem de Desenvolvimento do Aplicativo, (Fonte: Elaborada pelo Autor, 2025). 54
Tabela 9 – Contribuição do JavaScript na Aplicação, (Fonte: Elaborada pelo Autor, 2015). 55
Tabela 10 – Contribuição do Node.js na Aplicação, (Fonte: Elaborada pelo Autor, 2015). 56
Tabela 11 – Principais Serviços de Geolocalização e Mapas (Fonte: Elaborada pelo Autor, 2015). 57
Tabela 12 – Benefícios Combinados da IDE VSCode e Git/GitHub no Aplicação. (Fonte: Elaborada pelo Autor, 2015). 60

Lista de abreviaturas e siglas

AWS	Amazon Web Service
API	Application Program Interface ou Interface de Programação de Aplicativos
CAD	Computer Aided Design ou Desenho Auxiliado por Computador
CE	Comunidade Europeia
CSS3	Cascading Style Sheets ou Folhas de Estilo em Cascata
DRF	Django Rest Framework
DSP	Dynamic Shortest Path ou Caminho Mais Curto Dinâmico
FPS	Frames Per Second ou Quadros Por Segundo
HTML	HyperText Markup Language ou Linguagem de Marcação de Hipertexto
IDE	Integrated Development Environment ou Ambiente de Desenvolvimento Integrado
IBGE	Instituto Brasileiro de Geografia e Estatística
IPEA	Instituto de Pesquisa Econômica Aplicada
IoT	Internet Of Thing ou Internet das Coisas
IP	Internet Protocol ou Protocolo de Internet
SDKs	Kits de Desenvolvimento de Software
KNN	K-nearest Neighbors ou K-vizinhos mais próximos
ML	Machine Learning ou Aprendizado de Máquina
OMS	Organização Mundial da Saúde
OSM	OpenStreetMap
CTA	Sistema de Controlo de Tráfego Urbano
RkSP	Random K Shortest Paths ou Caminhos mais curtos aleatórios K
REST	Representational State Transfer ou Transferência de Estado Representacional
RSU	Road Side Unit ou Unidade de beira de Estrada

SIG	Geographic Information System ou Sistema de Informação Geográfica
GPRS	General Packet Radio Service ou Serviço Geral de Rádio por Pacotes
GPS	Sistema de Posicionamento Global
EcoTec	Sistema Roteamento de Veículo Baseado em VANET
SOAP	Simple Object Access Protocol ou Protocolo de Acesso a Objetos Simples
SUMP	Sustainable Urban Mobility Plan ou Plano de Mobilidade Urbana Sustentável
TIC	Tecnologia de Informação e Comunicação
TP	Transporte Público
VANET	Vehicular Ad Hoc Network ou Uma Rede Veicular Ad Hoc (VANET)
VSCode	Visual Studio Code
YOLO	You Only Look Once ou Você Só Olha Uma Vez

Lista de símbolos

CO_2	Dióxido de carbono.
$\sqrt{}$	Raiz quadrada.
$\sum$	Somatório.
XX	Numeração romana equivalente a 20.
XXI	Numeração romana equivalente a 21.

Sumário

1 Introdução 16
1.1 Problema da Pesquisa 18
1.2 Hipótese 18
1.3 Objetivos 20
1.3.1 Objetivo Geral 20
1.3.2 Objetivos Específicos 20
1.4 Justificativa 20
1.5 Estrutura do Trabalho 22

2 Fundamentação Teórica 24
2.1 Conceito de Mobilidade Urbana 24
2.1.1 Desafios da Mobilidade Urbana 25
2.1.1.1 Aplicação Móvel no Contexto de Mobilidade 29
2.1.1.2 Mobilidade Urbana Em Tempo Real 30
2.2 Conceitos de Sistemas de Informação Geográfica e Geolocalização 31
2.2.1 Tecnologias Para Mapeamento e Navegação (Google Maps API, OpenStreetMap) 31
2.2.1.1 Google Maps API 32
2.2.1.2 *OpenStreetMap* 32
2.2.2 Tráfego e Monitoramento em Tempo Real — Uso de Sensores e APIs de Trânsito 32
2.3 Inteligência Artificial na Otimização do Tráfego Urbano 33
2.4 Desenvolvimento de Aplicativos Móveis 34
2.4.0.1 Arquitetura de Sistemas Móveis (*Android/iOS*) 35
2.4.1 Frameworks e Possíveis Linguagens a Serem Utilizadas (Flutter, React Native e Kotlin) 36
2.5 Conceitos de *Back-end* e Integração de APIs 37
2.5.0.1 REST 38
2.5.0.2 SOAP 39
2.5.0.3 GraphQL 40
2.5.0.4 Integração de APIs 41
2.5.1 Algoritmo K-nearest Neighbors KNN 41
2.6 Algoritmo K-Vizinhos Mais Próximos (KNN) 41
2.7 Trabalhos Relacionados 45

3 Referencial Tecnológico. 48
3.1 Linguagens e *Frameworks* Utilizadas no Desenvolvimento da Aplicação 48
3.1.1 CSS3. 49
3.1.2 HTML5. 50
3.1.3 Django 51
3.1.4 Python 52
3.1.5 JavaScript 54
3.1.6 Node.js 55
3.2 Serviços de Geolocalização e Mapas. 56
3.2.1 APIDOG Plataforma Utilizada para Teste da API da Aplicação. 57
3.2.1.1 Por Que Usar Frameworks de Desenvolvimento de API? . . . 57
3.2.2 Ambiente de Desenvolvimento 59
3.2.2.1 VSCode 59
3.2.2.2 Git e GitHub 60
3.3 *Ionic Framework* com Angular para Desenvolvimento de Aplicativos Híbridos . 61
3.4 KNN: Principais Componentes 62

4 Metodologia 64
4.1 Metodologia da Pesquisa Teórica (Revisão Bibliográfica) 64
4.1.1 Metodologia da Pesquisa Prática (Desenvolvimento do Aplicativo) . . . 65
4.2 Levantamento de Requisitos 66
4.3 Coleta de Requisitos 66
4.3.1 Introdução à Coleta de Requisitos 66
4.3.2 Técnicas e Ferramentas Utilizadas 66
4.3.2.1 Questionários Online 67
4.3.2.2 Observação Direta 67
4.3.2.3 Análise de Documentos 67
4.3.2.4 Prototipação de Interfaces 68
4.3.3 Resultados da Coleta de Requisitos 68
4.3.4 Requisitos Funcionais (RF) 68
4.3.5 Requisitos Não Funcionais (RNF) 69
4.4 Modelagem de Requisito 70
4.5 Arquitetura do Sistema 73

5 Apresentação e Análise de Resultados 74
5.1 Interpretação dos Dados Coletados 74
5.2 Perfil dos Participantes 75
5.3 Utilidade e Interesse a Aplicação 77
5.4 Funcionamento da Aplicação em Diferentes Locais 77
5.5 Funcionalidades Essenciais da Aplicação 79

5.6 Avaliação Quanto À Capacidade do Aplicativo em Localizar Postos de Combustíveis 80
5.7 Tempo de Resposta para Localização e Recomendação dos Postos 81
5.8 Desafios da Implementação da Aplicação na versão Beta com Base aos Feedback dos Usuários 82
5.8.1 Síntese dos Desafios 84
5.9 Tela Principal da Aplicação 84
5.10 Tela tela de Busca da Aplicação 86
5.11 Tela de Processamento de Dados da Aplicação 87
5.12 Representação da Localização dos Postos de Combustíveis Sugeridos no Mapa em Grafos 88
5.13 Tela dos Resultados e Sugestão do Postos 88

6 Conclusões e Trabalhos Futuros 91

Referências 93

Apêndices 100
.1 Pseudocódigo da Função extract_features_from_station 101
.2 Pseudocódigo da Função compute_knn_score 101
.3 Pseudocódigo da Classe PredictionView 102
.4 Pseudocódigo da Classe PredictFromLocationView 103

Anexos 104

1 Introdução

As cidades contemporâneas lidam com um grande desafio até nos tempos de hoje, que é a mobilidade urbana, mormente nos grandes centros, onde há enorme fluxo de frota de veículos, a concentração populacional e a infraestrutura viária limitada resultam no crescimento exponencial de congestionamento. Esse cenário abala diretamente a qualidade de vida da população, gerando perda de tempo, emissão de poluentes e aumento do consumo de combustível. Dentre as diversas situações enfrentadas no trânsito, sobreleva-se a necessidade de localizar postos de combustíveis durante o deslocamento diário. Por mais que hajam aplicativos online e offline de navegação, como Waze, Google Maps e outros, que já auxiliam o motorista a obter informações sobre o estado do tráfego, os mesmos não são o suficiente para cobrir essa rotura existente na tomada de decisão eficiente que há na busca desses estabelecimentos responsáveis por prestar o serviço de abastecimento, pois tais aplicações são generalistas, ou seja, elas oferecem uma infinidade de funcionalidades, mas não são otimizados exclusivamente para decisões rápidas sobre postos de combustíveis. Desta forma, não são especificamente voltadas para otimizar o processo de escolha do posto de combustível ideal considerando múltiplos fatores simultaneamente.

O padrão de mobilidade da população brasileira vem passando por fortes modificações desde meados do século passado, reflexo principalmente do intenso e acelerado processo de urbanização e crescimento desordenado das cidades, além do uso cada vez mais intenso do transporte motorizado individual pela população (CARVALHO et al., 2011).

O aumento do transporte individual motorizado e consequente redução das viagens do transporte público vêm contribuindo para a deterioração das condições de mobilidade da população dos grandes centros urbanos, principalmente em função do crescimento dos acidentes de trânsito com vítimas, dos congestionamentos urbanos e também dos poluentes veiculares (VASCONCELLOS; CARVALHO; PEREIRA, 2011).

- Estudo realizado por (BAPTISTA et al., 2024) trata a respeito da possibilidade de aplicar a

> Inteligência Logística de roteirização nos processos de empresas de transporte rodoviário. A ideia conceito visa a criação de uma plataforma que integra digitalmente inteligência de rotas, eficiência operacional, segurança, conformidade e sustentabilidade, utilizando algoritmos avançados para otimização de rotas, monitoramento em tempo real através de dispositivos IoT, interfaces amigáveis para usuários, e análise contínua de dados para melhorar o desempenho da frota; além de promover a economia de combustível, a manutenção preventiva, e a redução de emissões de CO_2, garantindo a segurança dos veículos e a conformidade com as regulamentações de transporte evidenciando, dessa forma a sua utilização como facilitadores no dia a dia.

Para contexto do gênero, torna-se viável buscar por soluções que possam unir tecnologias de geolocalização, análise de tráfegos em tempo real e algoritmos de aprendizado de máquina, de modo a oferecer recomendações mais práticas e mais inteligentes aos motoristas. Entre os algoritmos disponíveis, o K-Nearest Neighbors (KNN) que se destaca por sua eficiência e simplicidade de aplicabilidade em problemas de classificação e recomendação dos melhores caminhos ao destino almejado.

O mesmo pode pelo seu comportamento ser utilizado para identificar, dentre um conjunto de postos de combustíveis, aqueles que representam as melhores opções de acesso para abastecimento com base em critérios como distância e congestionamento. Logo, esse trabalho propõe o desenvolvimento de um aplicativo móvel de apoio à mobilidade urbana, cuja o objetivo principal é recomendar ao usuário o posto de combustível mais próximo e acessível, levando em consideração não apenas a localização, mas também as condições de tráfego em tempo real. A aplicação utilizará o algoritmo KNN para concretizar a seleção do posto mais apropriado, buscando otimizar o tempo de deslocamento, contribuir na redução de impactos ambientais gerados por congestionamento e reduzir gasto desnecessário do combustível. (JENKINS, 2009) diz que, com o avanço da tecnologia, os dispositivos móveis tornam-se cada vez mais populares e vêm ganhando espaço no cotidiano das pessoas e no mercado de trabalho. Dessa Forma, o mercado em dispositivos móveis passa por grandes mudanças, visto que, tem se incorporado à telecomunicação à tecnologia digital. No entanto, (FIGUEIREDO; NAKAMURA, 2003), mesmo com a crescente inovação e a aplicabilidade de dispositivos móveis, nem sempre atendem a todas as necessidades dos usuários. Sendo assim uma forma de garantir que a aplicação móvel esteja de acordo com a necessidade dos usuários faz-se necessário avaliar sua usabilidade.

A mobilidade tem-se tornado num dos maiores desafios que as cidades enfrentam nos dias de hoje, cada vez mais, influenciada por fatores que dificultam a circulação (congestionamento de tráfego, informação não atualizada, eventos, acidentes, supressão de vias, obras, entre outros). Mais de metade da população, cerca de 54% reside nas áreas urbanas e em 2050 é previsto que este número alcance os 67% (WARD; SMITH, 2015). Todo este crescimento cria um conjunto de vários desafios para os quais as cidades necessitam estar preparadas. A necessidade constante de

melhoramento das infraestruturas e da adaptação da rede de transportes capazes de responder às necessidades dos cidadãos é hoje uma realidade. O aumento da poluição, que advém do excesso de veículos em circulação, através das emissões de dióxido de carbono para a atmosfera é por si só, um fator preocupante. A venda de automóveis tem continuado a crescer tendo-se como estimativa um aumento de 125 milhões de vendas em 2025 sendo que alguns analistas indicam que tal aumento poderá mesmo duplicar no ano de 2030 (DARGAY; GATELY; SOMMER, 2007). Segundo a Organização Mundial de Saúde, já mesmo no ano de 2014, sete milhões de mortes prematuras ocorreram devido à poluição do ar e uma parte significativa dessa causa foi diretamente relacionada com o trânsito urbano (EFAGUNDES, 2014).

1.1 Problema da Pesquisa

A crescente complexidade da mobilidade urbana, associada ao aumento do fluxo de veículos e à necessidade constante de abastecimento, faz com que motoristas enfrentem dificuldades para identificar rapidamente o posto de combustível mais acessível, considerando fatores como distância, tempo de deslocamento e níveis de congestionamento. Embora existam aplicativos de navegação e localização, a maioria não oferece uma solução integrada e direcionada exclusivamente à escolha otimizada de postos de combustível com base em condições de tráfego em tempo real.

Considerando as dificuldades enfrentadas pelos motoristas para identificar postos de combustíveis acessíveis e com baixa demanda, como projetar uma aplicação móvel que utilize o algoritmo KNN para recomendar, com precisão, os postos mais próximos e com menor congestionamento, otimizando o fluxo de deslocamento no ambiente urbano?

1.2 Hipótese

O uso do algoritmo KNN em uma aplicação móvel permite recomendar postos de combustíveis de forma mais eficiente, identificando opções próximas e com menor congestionamento.

A eficiência da mobilidade urbana está diretamente relacionada à disponibilidade de informações precisas e atualizadas sobre o ambiente de tráfego. Contudo, motoristas muitas vezes desconhecem quais postos de combustíveis oferecem menor tempo de atendimento ou têm menor movimento, o que pode gerar deslocamentos prolongados, desperdício de combustível e agravamento do congestionamento urbano. Frente a esse cenário, torna-se relevante propor uma solução tecnológica capaz de centralizar e processar dados de mobilidade, indicando de forma inteligente os postos mais vantajosos. A aplicação do algoritmo KNN justifica-se por sua capacidade de identificar padrões e realizar recomendações rápidas com base em múltiplos parâmetros, promovendo maior eficiência nas escolhas dos usuários e contribuindo para uma mobilidade mais sustentável e estratégica.

Hipótese da Pesquisa: Se um aplicativo móvel integrar o algoritmo KNN para recomendar postos de combustíveis com base em proximidade, fluxo de tráfego e tempo estimado de atendimento, então o tempo de deslocamento dos motoristas será reduzido, resultando em menor consumo de combustível e maior eficiência na mobilidade urbana.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver um aplicativo móvel de apoio à mobilidade urbana que utilize o algoritmo de aprendizado de máquina K-Nearest Neighbors (KNN) para recomendar ao usuário o posto de combustível mais próximo, de forma a otimizar o tempo de deslocamento e contribuir para a eficiência do transporte urbano.

1.3.2 Objetivos Específicos

E para os objetivos específicos do trabalho temos os seguintes pontos:

- Levantar requisitos funcionais e não funcionais do aplicativo, considerando as necessidades do usuário no processo de abastecimento;
- Coletar e integrar dados de localização via GPS, garantindo precisão na identificação da posição do usuário;
- Obter informações de trânsito em tempo real por meio de APIs de mapas e mobilidade (ex.: Google Maps API, OpenStreetMaps ou Similares);
- Modelar e implementar o algoritmo KNN para classificar e recomendar postos de combustível de cordo com a distância e congestionamento;
- Desenvolver a interface do aplicativo móvel com foco em usabilidade, acessibilidade e eficiência;
- Testar, validar a solução proposta e eficiência do algoritmo;

1.4 Justificativa

O crescimento acelerado da frota de veículos e a expansão desordenada das cidades têm intensificado os desafios relacionados à mobilidade urbana. De acordo com (CARVALHO et al., 2011), nos últimos anos, no Brasil, com o aumento do transporte individual motorizado, as condições de mobilidade da população vêm se degradando muito, principalmente em função do crescimento dos acidentes de trânsito com vítimas, dos congestionamentos urbanos e também dos poluentes veiculares. A partir desse contexto, procurou-se iniciar um debate sobre os desafios mais importantes para dotar as cidades brasileiras com sistemas de mobilidade mais qualificados. Este texto apresenta alguns desses desafios destacados pelo autor, discutindo também políticas públicas necessárias para superar os problemas expostos.

Perante a este cenário, não se descarta a possibilidade da procura de soluções tecnológicas que possam melhorar a eficiência do deslocamento dos motoristas, pois a mesma se mostra

essencial. Aplicativos de navegação e mobilidade já desempenham um papel relevante no cotidiano dos condutores, mas devido suas limitações em muitos casos ainda não complementam de maneira eficaz critérios que extrapolam a distância geográfica. Para casos como a necessidade de abastecimento em postos de combustíveis, um sistema que considere também o nível de congestionamento das vias de acesso pode trazer ganhos significativos em tempo de escassez e economia.

Além do ambiental, os congestionamentos também possuem um aspecto econômico relacionado aos custos oriundos do mesmo. Diante desse quadro, há a necessidade premente de se buscar soluções e alternativas sustentáveis para esse problema, seja por meio da alteração da matriz energética, de novas tecnologias automotivas, de políticas públicas ou de programas de conscientização, entre outros (LONDON; ROMIEU, 2000).

A utilização de algoritmos de aprendizado de máquina, como o K-Nearest Neighbors (KNN), justifica-se pela capacidade de analisar múltiplos fatores simultaneamente e oferecer recomendações mais inteligentes e contextualizadas. Segundo (TAN; STEINBACH; KUMAR, 2016), o KNN é um dos algoritmos mais utilizados em problemas de classificação e recomendação pela sua simplicidade e eficiência em cenários que envolvem dados geoespaciais. Nesse contexto, ao empregar o KNN na escolha do posto de combustível mais próximo e com menor tempo de acesso, o sistema proposto contribuirá para a otimização de trajetos, a redução de custos com combustível e a diminuição da emissão de poluentes, alinhando-se a princípios de sustentabilidade e cidades inteligentes (BATTY, 2018).

Segundo (JUNIOR, 2017), o uso das Tecnologias de Informação e Comunicação (TIC) promoveu mudanças radicais na sociedade, especialmente devido ao maior acesso à informação e às novas formas de comunicação, agora mais céleres e síncronas. A introdução do computador e os avanços proporcionados pelo uso da Internet foram fatores decisivos para que essas transformações ocorressem. Além disso, o surgimento e popularização de dispositivos móveis, como celulares, *tablets*, notebooks e netbooks, tornaram o acesso à web cada vez mais simples e presente no cotidiano, favorecendo uma conectividade ubíqua.

Num âmbito mais acadêmico, a relevância do tema está na aplicação prática de técnicas e conceitos de programação, ciência de dados e sistemas em tempo real em um problema real de mobilidade urbana. Além disso, a integração de métodos de classificação com recursos de geolocalização e informação de tráfego em tempo real reforça o caráter interdisciplinar da pesquisa, estabelecendo uma conexão entre Engenharia da Computação, Engenharia de Transporte e Inovação Tecnológica.

Nos últimos anos, nota-se a crescente demanda do pensamento computacional como uma habilidade do século XXI (GROVER; PEA, 2013). O pensamento computacional denota a ideia de desenvolver uma solução genérica para um problema decompondo-o, identificando variáveis e padrões relevantes, derivando um procedimento de solução algorítmica (WING,

2006). Como tal, o pensamento computacional representa uma capacidade cognitiva para aplicar conceitos fundamentais e raciocínio que derivam da ciência da computação em geral e da programação/codificação de computadores em particular para diferentes outros domínios, incluindo atividades da vida real (FERREIRA et al., 2020).

Portanto, a escolha desse tema se justifica exclusivamente pela relevância social, econômica e científica. Além de conceder uma solução potencialmente útil para os motoristas, o projeto contribui para o avanço de estudos que aplicam métodos de desenvolvimento de aplicações e sua interligação com algoritmos de inteligência artificial, especificamente o KNN, em contextos de mobilidade urbana. Ademais, reforça a relação entre o desenvolvimento de soluções tecnológicas aplicáveis ao cotidiano e a pesquisa acadêmica.

1.5 Estrutura do Trabalho

Este trabalho está organizado em seis (6) capítulos primordiais, especificando o propósito de cada elemento e suas interconexão, por consequência gerando uma leitura bem mais específica, clara e precisa.

- **Capítulo 1: Introdução**, contextualiza de modo geral o problema, os objetivos gerais e específicos que dinamizaram o desenvolvimento do trabalho, assim como a justificativa da escolha do tema e a importância da aplicação proposta.
- **Capítulo 2: Fundamentação Teórica**, neste capítulo, apresentamos os conceitos teóricos que embasaram o desenvolvimento da aplicação. Partimos da análise de aspectos relacionados à mobilidade urbana e seus principais desafios. Em seguida, abordamos fundamentos sobre informação geográfica, geolocalização e tecnologias utilizadas para mapeamento e navegação. Também foram considerados conceitos sobre tráfego em tempo real e inteligência artificial, que sustentam a relevância da solução proposta. Esses elementos justificam o desenvolvimento da aplicação e demonstram a necessidade de um sistema centralizado e otimizado para sugerir postos de combustível aos usuários.
- **Capítulo 3: Referencial Tecnológico**, apresentamos neste capítulo, as tecnologias utilizadas no desenvolvimento do sistema, desde os *frameworks*, bibliotecas, ferramentas de desenvolvimento, serviços externos empregados de APIs e a implementação de mecanismo de aprendizado de máquina especificamente o algorítmo K-nearst Neighbors KNN.
- **Capítulo 4: Metodologia**, baseá-se este capítulo nos mecanismos e ferramentas adotadas para o desenvolvimento do aplicativo, partindo pelo processo de levantamento de requisitos; além disso, apresentou-se no mesmo técnicas utilizadas para garantir que o aplicativo alcance os objetivos propostos.

- **Capítulo 5: Apresentação e Análise de Resultados**, apresenta as análises e os resultados obtidos da aplicação desenvolvida.
- **Capítulo 6: Conclusão e Trabalhos Futuros**, nesse capítulo abordamos sobre as considerações finais, resumiu-se também sobre a metodologia utilizada e seu contributo, as limitações do trabalho e a possibilidade dos trabalhos futuros.

2

Fundamentação Teórica

Neste capítulo, apresentam-se, de forma objetiva, os principais conceitos que fundamentam o tema proposto neste trabalho. São abordados tópicos referentes à mobilidade urbana, aos aplicativos móveis, às técnicas de aprendizado de máquina e, por fim, ao algoritmo K-Nearest Neighbors (KNN), que constitui o elemento central do sistema de recomendação desenvolvido.

2.1 Conceito de Mobilidade Urbana

O conceito mais recorrente de mobilidade urbana está relacionado ao deslocamento de pessoas no cotidiano, tanto dentro das cidades quanto entre elas (WACHS, 2010; HANSON, 2010). Tradicionalmente, compreendia-se que a movimentação física era um requisito essencial para o acesso a serviços como habitação, saúde, lazer, educação e emprego. Contudo, com o avanço das tecnologias digitais, essa premissa deixou de ser absoluta, uma vez que grande parte desses serviços pode ser acessada de forma remota.

Dessa forma, a mobilidade urbana passou a ir além do deslocamento físico e passou a incorporar a noção de acessibilidade aos serviços urbanos (WACHS, 2010). O conceito também se expandiu, abrangendo aspectos relacionados à inclusão social e ao desenvolvimento socioeconômico (BACKES et al., 2016).

A mobilidade urbana é um atributo relacionado às pessoas e aos agentes econômicos presentes no ambiente urbano, que, de diferentes formas, buscam atender e suprir suas necessidades de deslocamento para a realização de atividades cotidianas, como trabalho, educação, saúde, lazer e cultura. Para esse fim, os indivíduos podem empregar esforço próprio — realizando o deslocamento a pé — ou utilizar meios de transporte não motorizados (bicicletas, carroças, cavalos) e motorizados, sejam eles coletivos ou individuais (FANINI, 2011).

Com o crescimento dos territórios urbanos e a expansão de regiões periféricas, em um

processo acelerado e historicamente marcado pela falta de planejamento, observa-se a formação de cidades cada vez mais dispersas, nas quais as distâncias entre serviços, locais de trabalho e moradia aumentam proporcionalmente ao avanço urbano (CEBDS, 2016). Esse cenário afeta tanto a oferta de serviços e infraestrutura quanto a capacidade e efetividade das cidades em gerir o intenso fluxo de pessoas que se deslocam diariamente entre diferentes pontos do território.

Considerando que grande parte desses deslocamentos ocorre entre residência e trabalho, e que, especialmente em países em desenvolvimento, eles são realizados majoritariamente por veículos altamente poluentes — como ônibus, automóveis e motocicletas —, evidencia-se um conjunto de desafios urgentes e crescentes a serem enfrentados.

2.1.1 Desafios da Mobilidade Urbana

A mobilidade urbana no Brasil, assim como em diversos outros países, não constitui um problema recente e apresenta-se como um cenário de crise contínua, intensificado desde o início do século XX. O crescimento populacional e industrial, aliado ao aumento expressivo de veículos automotores individuais — que historicamente têm recebido prioridade em relação a outros meios de locomoção, como o transporte coletivo e as alternativas não motorizadas — amplia as desigualdades no acesso ao espaço urbano. Nesse contexto, indivíduos de menor renda e baixo status socioeconômico são diretamente prejudicados em seus deslocamentos diários, o que tem motivado reações e manifestações da sociedade civil em defesa de melhorias nas condições de mobilidade urbana (RODRIGUES, 2016).

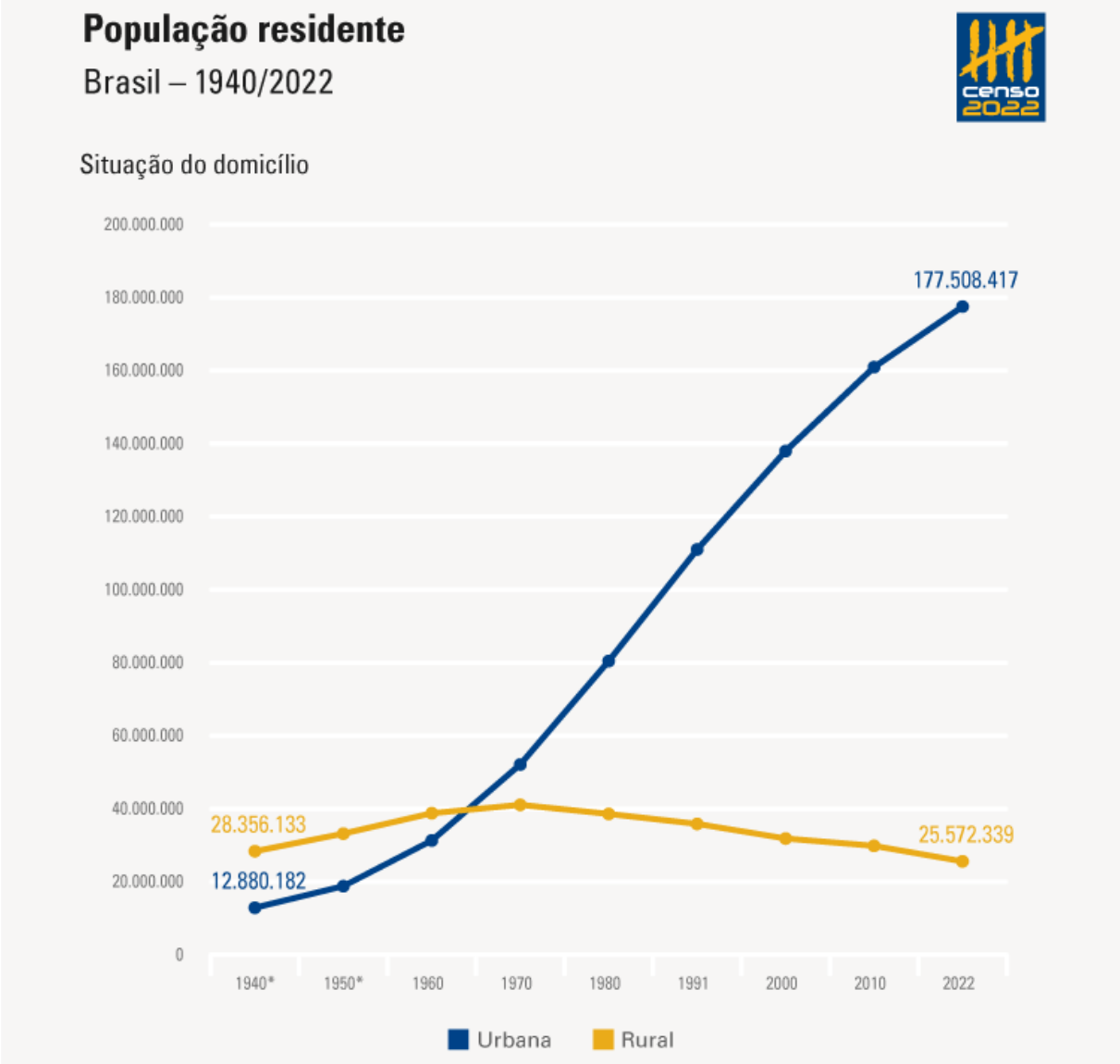
O deslocamento diário constitui uma necessidade comum a todos, porém as condições em que esse deslocamento se realiza — como frequência, motivos, fluidez, conforto, segurança e meio de transporte utilizado — dependem de fatores individuais, familiares e territoriais. A mobilidade urbana pode, portanto, ser compreendida como o conjunto de facilidades e capacidades relacionadas ao transporte de pessoas e mercadorias no espaço urbano. Estudos clássicos na área de transporte indicam que a situação socioeconômica exerce influência direta sobre a mobilidade cotidiana, visto que indivíduos com maior renda tendem a realizar deslocamentos com maior facilidade, acesso e frequência. Isso se deve, em grande parte, ao uso do automóvel particular, o que evidencia desigualdades estruturais e contraria o princípio de igualdade no uso dos espaços públicos de circulação das cidades (RODRIGUES, 2016).

A busca por um sistema de mobilidade mais igualitário, financeiramente sustentável e que não exclua populações de baixa renda deve ser responsabilidade direta dos gestores públicos. Para avançar nessa direção, é necessário superar diversos desafios, como a falta de integração entre as políticas de desenvolvimento urbano e metropolitano e o planejamento dos sistemas de mobilidade; a ausência de políticas contínuas de financiamento e investimento em infraestrutura de transporte público; a carência de medidas de racionalização do uso de veículos individuais motorizados e de compensação pelas externalidades negativas que estes geram; além do impacto

do envelhecimento populacional nas condições de deslocamento e nos custos do transporte coletivo. Soma-se a isso a necessidade de revisão do modelo regressivo de financiamento atualmente vigente no Brasil, entre outros entraves.

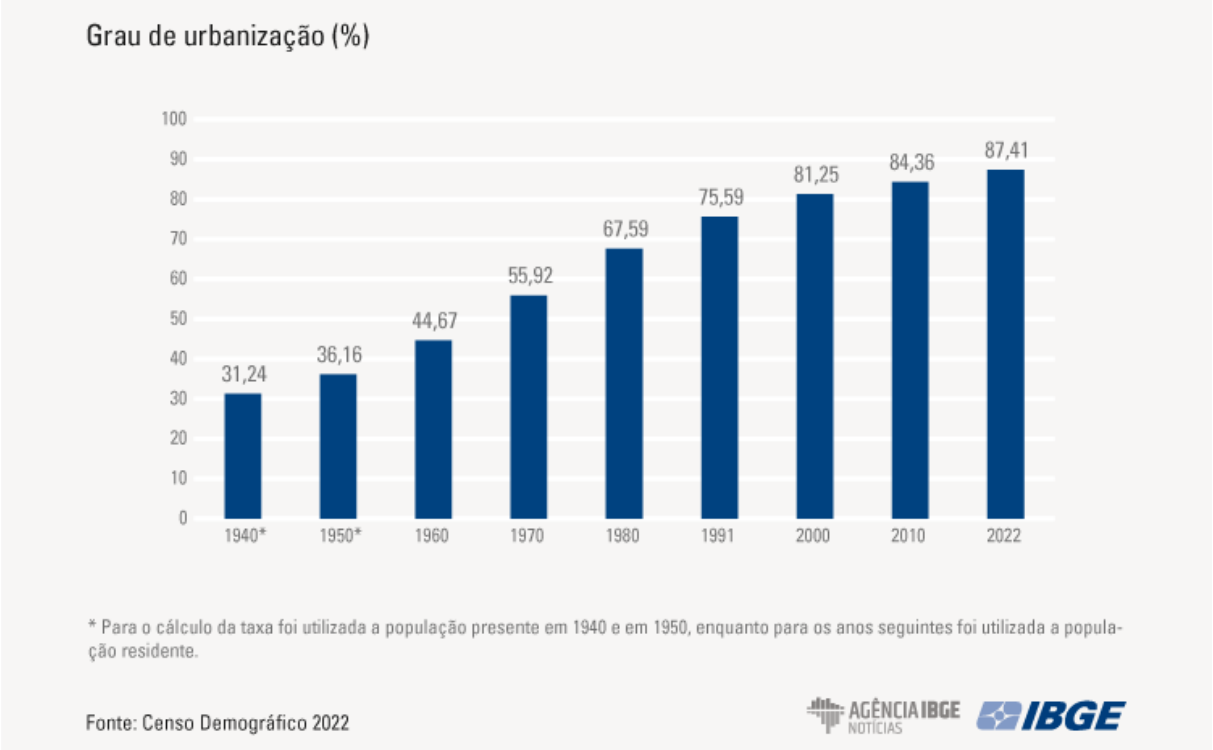
Há pouco mais de quatro décadas, a maior parte da população brasileira ainda residia em áreas rurais, onde a demanda por transporte de massa era limitada, dado o número reduzido de centros urbanos. Atualmente, contudo, cerca de 85% da população encontra-se em áreas urbanas, com 36 cidades que superam 500 mil habitantes e aproximadamente 40 regiões metropolitanas estabelecidas. Nessas regiões vivem mais de 80 milhões de brasileiros — quase 45% da população nacional —, cenário que intensifica a necessidade de políticas públicas eficientes e inclusivas voltadas à mobilidade urbana (DE CARVALHO, 2016). A Figura 1 ilustra o crescimento populacional brasileiro e a Figura 2 apresenta a evolução do grau de urbanização.

Figura 1 – Censo da População Brasileira a Viver em Áreas Urbanas e Rurais



Fonte: (BRASIL, 2023)

Figura 2 – Grau de Urbanização



Fonte: (BRASIL, 2023)

Com a criação da lei que instituiu as diretrizes da Política de Mobilidade Urbana brasileira, torna-se necessário investigar os avanços e as barreiras observadas em sua referência original: a Comunidade Europeia (CE). Reduzir os impactos do transporte e alcançar o desenvolvimento urbano sustentável têm motivado a CE a investir em pesquisas e projetos que culminaram na recomendação para que as cidades adotassem o *Sustainable Urban Mobility Plan (SUMP)*. Embora muitas delas tenham implementado essa proposta, verifica-se que, dez anos depois, os avanços efetivos na redução dos congestionamentos e das emissões atmosféricas ainda são limitados.

Uma revisão sistemática composta por 37 estudos sobre o SUMP apresentado por (MACHADO; PICCININI, 2018), evidenciando a evolução da política de mobilidade na CE, os guias metodológicos para sua elaboração, as barreiras à implementação e as recomendações para avaliação dos planos. A revisão apontou que a metodologia do SUMP foi adotada também em países fora da CE, como Brasil, México e Índia, sugerindo uma hegemonia no planejamento da mobilidade. Observou-se que a efetiva implantação do SUMP depende, além de uma avaliação contínua, do enfrentamento dos desafios decorrentes das decisões políticas sobre as técnicas adotadas, da redução das desigualdades socioespaciais, da integração entre os níveis de governo, dos projetos setoriais, dos modos de transporte e das medidas propostas.

Ao buscar soluções para os inevitáveis percalços da vida urbana, propõe-se que, em vez

de iniciar tal exame por meio da leitura de estatísticas, aproximemo-nos da cidade através da leitura de sua paisagem — física e humana. Com efeito, ao circularmos preferencialmente na condição de pedestre ou ciclista, a fim de dispor de tempo e calma necessários ao exercício da percepção, o que se verá é uma sequência de paisagens. Conduzidos pelas vias e demais espaços públicos, a leitura da cidade de São Paulo revela um descaso com o sítio natural e sua topografia, contradizendo a potencialidade perdida de amplos horizontes propiciados pelos declives; ou, ao contrário, a paisagem pode exaltar os rasgos de céu e o sol aquecendo as calçadas.

Estranha-se, ainda, o que ocorreu com suas grandes várzeas — a do Tietê e a do Pinheiros — ambas marcadas por amplos meandros e cursos de rios lentos e variáveis, decorrentes da pequena declividade (6%) que percorrem no município. Em vez de um território que acolha os rios e suas matas, tornando-os parte do uso urbano, observam-se hoje moradias sujeitas à inundação e edifícios que poderiam estar em locais mais adequados. A paisagem urbana, além de revelar volumes construídos e intensidades de ocupação do solo, evidencia também o fluxo de pessoas e veículos, bem como as condições das estruturas físicas que sustentam tais movimentos.

Por esse motivo, o segundo passo para a apreensão de uma cidade consiste no levantamento do relevo do sítio e de suas infraestruturas físicas — redes viárias e demais malhas, como trilhos, redes aéreas para distribuição de energia, sistemas de abastecimento de água e coleta de esgoto. Ao descrever tais infraestruturas, torna-se indispensável explicar o processo histórico que conduziu à ocupação do solo urbano (WILHEIM, 2013).

2.1.1.1 Aplicação Móvel no Contexto de Mobilidade

O rápido desenvolvimento da tecnologia gera um enorme potencial para transformar os conceitos de mobilidade urbana (FANG, 2015). A mobilidade, sustentada pelo sistema de transporte urbano, é dinâmica, influenciada pela sociedade e capaz de impactar diretamente a capacidade de escolha das pessoas, à medida que expande ou limita suas opções de deslocamento — seja pelo custo, pelo tempo de viagem ou por outros fatores (KUNIEDA; GAUTHIER, 2007). Na visão de (GOWRISHANKAR; STERN; WORK, 2014), melhorias na mobilidade podem tanto auxiliar no crescimento das cidades quanto elevar a qualidade de vida em escala global.

Nesse cenário, diversos aplicativos de mobilidade urbana têm sido desenvolvidos e utilizados de forma crescente com o objetivo de facilitar o cotidiano dos usuários (CHAVES; STEINMACHER; VIEIRA, 2011); (CHEN et al., 2014), ainda que apresentem propostas e serviços distintos. Entre eles, destacam-se aplicativos para automóveis, aplicativos de táxi, aplicativos voltados ao transporte público, aplicativos para pedestres e aplicativos para bicicletas (FRANÇOZO; MELLO, 2016).

A tecnologia pode ser compreendida como diferentes formas de expressão da identidade, atuando como meio, determinante ou consequência de uma identidade pessoal formada a partir de experiências prévias (CARTER; GROVER, 2015). Na primeira perspectiva — tecnologia

como meio —, esta atua estimulando a confiança, funcionando como instrumento de verificação da identidade e de projeção do indivíduo para os outros. Na segunda perspectiva — tecnologia como determinante —, "a TI pode influenciar a identidade dos indivíduos nas posições sociais que ocupam" (p. 938). Por fim, a tecnologia como consequência refere-se ao envolvimento da TI nas percepções de continuidade, em que a identidade passa a refletir significados construídos nas relações por meio do comportamento (FLORIDI, 2010); (STETS; BIGA, 2003).

Dessa forma, as tecnologias móveis transformaram o relacionamento entre o ser humano e os dispositivos tecnológicos ao possibilitar que esses recursos o acompanhem em qualquer lugar, abrindo espaço para demandas antes inviáveis (CASTRO; TEDESCO, 2014). Assim, aplicativos de mobilidade urbana facilitam a rotina dos usuários ao executar tarefas como traçar rotas, exibir informações de tráfego, pedágios e pontos de referência, bem como horários estimados de chegada de transporte público (ALMEIDA et al., 2016).

E (SILVA; SANTOS, 2014) investigou como aplicativos contribuem para facilitar o cotidiano das pessoas em distintos contextos urbanos, especialmente no âmbito da mobilidade. O estudo identificou que os principais benefícios e fatores de adesão incluem agilidade nos serviços, otimização dos deslocamentos — reduzindo o tempo gasto — e maior controle do processo de viagem, uma vez que os usuários passam a dispor de mais informações sobre como, onde e quando chegar ao destino desejado.

2.1.1.2 Mobilidade Urbana Em Tempo Real

Nos últimos anos, os estudos sobre mobilidade urbana avançaram significativamente, com aumento na produção acadêmica e diversificação de abordagens. Essas perspectivas integram dimensões urbanas, ambientais e de inclusão socioespacial, evidenciando a complexidade e a pluralidade de conceitos que configuram novos olhares sobre o tema. A mobilidade urbana consolida-se, assim, como um pilar central da urbanização contemporânea, constituindo um componente fundamental do cotidiano das cidades em escala global. Mais do que um simples deslocamento físico, a mobilidade urbana representa um capital específico marcado pelo fluxo contínuo de pessoas, mercadorias e informações, cuja compreensão se amplia ao incorporar aspectos essenciais para a organização e estruturação do espaço urbano (ALVIM; IZAGA; CLAPS, 2024).

Essa urbanização informatizada favorece o monitoramento de sistemas em tempo real, bem como sua administração e melhoria contínua. No entanto, o conceito de *smart cities* não se limita ao uso de tecnologia, envolvendo também governança, infraestrutura e capital humano e social, que, em conjunto, visam ao desenvolvimento econômico e sustentável das cidades (ANDRADE; GALVÃ et al., 2016). Para que uma cidade seja considerada inteligente, é necessário compreender suas necessidades e de que forma elas podem ser atendidas. Nesse processo, a tecnologia pode atuar como suporte, "tornando os dados da vida urbana tangíveis por meio da criação e execução de projetos voltados para sua captura, tratamento e disponibilização em tempo

real" (WEISS, 2013).

2.2 Conceitos de Sistemas de Informação Geográfica e Geolocalização

Os Sistemas de Informação Geográfica (SIG) são ferramentas que permitem coletar, armazenar, analisar e representar dados georreferenciados, enquanto a geolocalização é a capacidade de identificar a posição de um objeto ou pessoa na superfície terrestre em tempo real. Segundo (CÂMARA; CASANOVA; MAGALHÃES, 1996) A implementação de SIGs requer integrar conhecimentos de diversas áreas da Ciência da Computação e de disciplinas relacionadas ao processamento de certos tipos específicos de dados.

A necessidade de identificar diferentes "camadas" de dados em uma série de mapas e, posteriormente, analisá-los e relacioná-los por meio da sobreposição é uma ideia muito anterior ao surgimento dos atuais Sistemas de Informação Geográfica. A evolução dos sistemas de informação não ocorreu como um evento isolado, mas acompanha a própria história da humanidade e o avanço das Geociências. Essa trajetória é perceptível desde os primeiros cálculos desenvolvidos pelos sumérios até os supercomputadores modernos; das representações cartográficas da Terra elaboradas por Ptolomeu até as imagens hiperespectrais de satélite; das referências espaciais obtidas pelo astrolábio ao Sistema de Posicionamento Global por Satélite (GPS); e dos registros de dados em mapas portulanos aos programas do tipo CAD *(Computer Aided Design)* (BOLFE; MATIAS; FERREIRA, 2008).

2.2.1 Tecnologias Para Mapeamento e Navegação (Google Maps API, OpenStreetMap)

Atualmente, a internet permite o acesso a diferentes formatos de informação, como textos, áudios, imagens e vídeos, todos consumidos de forma direta e interativa pelos usuários. O caráter multimídia da rede, aliado ao aumento da velocidade de transmissão de dados, ao aprimoramento do processamento computacional e à evolução dos algoritmos de compactação, contribui para a ampla disponibilização e circulação de grandes volumes de dados na Web.

Nesse contexto, as geotecnologias passaram a ocupar maior relevância no ambiente digital. Entre as diversas ferramentas disponíveis, destacam-se aquelas voltadas à criação, manipulação, visualização e edição de mapas, que têm despertado especial interesse. Os chamados *Web Maps* configuram-se como uma tecnologia amplamente utilizada, permitindo o desenvolvimento de mapas interativos, multiescalares, com controle de camadas, inclusão de conteúdos adicionais, animações e participação ativa dos usuários. Tais recursos superam as limitações do mapeamento em mídia impressa, ampliando o potencial de acesso e disseminação da informação geoespacial (OLIVEIRA; NETO; SANTOS, 2010).

2.2.1.1 Google Maps API

O *Google Maps API* é um conjunto de ferramentas da plataforma Google Maps que permite integrar mapas, geolocalização e serviços de navegação em aplicativos e sites. Na Web de hoje, as soluções de mapeamento são um ingrediente natural. Nós as usamos para ver a localização de coisas, procurar o endereço de um local, obter direções de trajeto e realizar inúmeras outras tarefas. A maioria das informações tem uma localização e, se algo tem uma localização, pode ser exibido em um mapa. Existem várias soluções de mapeamento, incluindo *Yahoo*! *Maps* e *Bing Maps*, mas a mais popular é o Google Maps. De fato, segundo o site *Programmableweb.com*, é o mais popular na Internet. De acordo com as estatísticas de maio de 2010 do site, 43% por cento de todas as combinações *("mashups")* utilizam a *Google Maps API*. Em comparação, a segunda API de mapeamento mais popular era a do *Flickr*, com 11%, e a terceira era a *VirtualEarth (Bing Maps)*, com 3%. Aplicações e sites que combinam dados ou funcionalidades de duas ou mais fontes são comumente chamados de mashups. Os mashups estão se tornando cada vez mais populares e revolucionaram a forma como a informação é usada e visualizada. As soluções de mapeamento são um ingrediente importante em muitos desses *mashups*. A *Google Maps API* permite aproveitar o poder do *Google Maps* para usar em suas próprias aplicações e exibir seus próprios dados (ou de outros) de forma eficiente e útil (SVENNERBERG, 2010).

2.2.1.2 *OpenStreetMap*

OpenStreetMap ou OSM é um projeto colaborativo que fornece mapas digitais gratuitos e abertos, criados e mantidos por voluntários em todo o mundo. Os mapas sempre foram uma fonte de poder. Os exércitos os usaram para obter vantagem militar, os comerciantes os usaram para encontrar a rota mais curta entre locais de fornecimento e mercados, e piratas fictícios os usaram para encontrar tesouros enterrados. Aqueles que possuem um mapa têm uma vantagem sobre aqueles que não têm. Não é de se admirar, então, que os mapas e as informações usadas para criá-los tenham sido fortemente protegidos por empresas e governos que os possuem. Muitas agências governamentais e comerciais de mapeamento cobram taxas altas para usar seus dados, impõem restrições rigorosas sobre o que você pode fazer com essas informações e podem até fazer você pagar para usar mapas que você mesmo desenhou. O projeto OSM busca mudar isso, dando a todos o seu próprio mapa, gratuitamente, e para usar da forma que quiserem (BENNETT, 2010).

2.2.2 Tráfego e Monitoramento em Tempo Real — Uso de Sensores e APIs de Trânsito

O monitoramento de tráfego em tempo real é um dos pilares da mobilidade urbana inteligente. Ele combina sensores físicos instalados em vias e veículos com APIs de trânsito que coletam, processam e disponibilizam dados atualizados sobre condições de tráfego, acidentes e

congestionamentos.

Diante desse cenário, a busca por soluções inovadoras e eficientes para o controle do tráfego urbano torna-se fundamental. A utilização de tecnologias da informação e comunicação, aliada a algoritmos de inteligência artificial, que permitem a criação de sistemas inteligentes capazes de monitorar o fluxo de veículos em tempo real, otimizar a sinalização semafórica e adaptar-se às condições dinâmicas do trânsito. Diversos estudos, demonstram a eficácia de sistemas inteligentes de controle de tráfego na redução do congestionamento e na melhoria da fluidez do trânsito (CHILORA, 2024).

O aumento populacional e a crescente demanda por serviços de transporte têm resultado em congestionamentos, atrasos e uma sobrecarga nas infraestruturas existentes, impactando diretamente a qualidade de vida dos cidadãos. Além disso, a imprevisibilidade dos horários de transporte e a falta de informação em tempo real dificultam o planejamento eficiente por parte dos usuários. Esses problemas são ainda mais críticos em áreas que dependem de transporte coletivo, como no contexto universitário, onde a pontualidade e a segurança são fatores essenciais para a rotina acadêmica e pessoal. Diante dessa realidade, torna-se fundamental explorar soluções tecnológicas que possam otimizar o gerenciamento de frotas e a comunicação entre os veículos e os usuários, melhorando a eficiência e a confiabilidade do transporte (GUIMARÃES, 2025).

2.3 Inteligência Artificial na Otimização do Tráfego Urbano

A Inteligência Artificial (IA) se refere a criação de máquinas, com a capacidade de pensar e agir como os seres humanos. São softwares que são capazes de abstrair, deduzir e aprender ideias. Dessa forma, tem como proposito ajudar nas tarefas do cotidiano, modernização e avançar nas pesquisas científicas (CABRAL; NUNES, 2021).

A IA consegue assemelhar a capacidade humana ligada a inteligência, como por exemplo a habilidade de análise para a tomada de decisão e o raciocínio. Sendo também um campo da ciência, com o intuito de desenvolver, estudar e utilizar máquinas para conseguir realizar, de maneira autônoma, atividades humanas. Alguns benefícios da IA são, auxiliar nos processos de análise, melhorando a tomada de decisão; aumento na velocidade no tratamento de informações; aumento da automação, criando maior velocidade no tratamento das informações; reduz riscos, erros e custos operacionais, entre outros (EFAGUNDES, 2019b).

A engenharia de transporte, segundo a resolução 1096/2017 do Conselho Federal de Engenharia e Agronomia (CONFEA), é responsável pela infraestrutura de transporte. O profissional faz o planejamento da construção e da manutenção da infraestrutura viária e de terminais rodoviários, ferroviários, portuários e aeroportuários. Nas cidades, atua na viabilização da mobilidade urbana, cuidando da sinalização viária, da gestão e do planejamento do transporte urbano (MACHADO; SILVA, 2022).

O profissional faz o planejamento da construção e da manutenção da infraestrutura viária e de terminais rodoviários, ferroviários, portuários e aeroportuários. Nas cidades, atua na viabilização da mobilidade urbana, cuidando da sinalização viária, da gestão e do planejamento do transporte urbano.

Os problemas no trânsito causados pela má gestão urbana geram a população estresse, acidentes e perda de tempo no trânsito. Além disso, os constantes congestionamentos, devido ao grande número de veículos nas vias, afetam a rotina dos cidadãos. Conforme um estudo realizado pelo Instituto Brasileiro de Economia da Fundação Getúlio Vargas (FGV), o custo adicional gerado pelo tempo perdido pela população nos deslocamentos para o trabalho para as cidades é de 62,1 bilhões de reais (EFAGUNDES, 2020).

Dessa forma, novas obras de infraestrutura que exploram o uso do território possuem limitações. Construir mais vias e estradas, em locais estratégicos, nos quais tenha um maior fluxo de veículos é cada vez menos possível. Portanto, é necessário técnicas que aumentam a eficiência das infraestruturas já existentes (JR, 2015).

Ao utilizar a AI, é possível melhorar a mobilidade dos cidadãos em uma cidade, otimizar os modais de transporte e melhorar o espaço público. Alguns aplicativos móveis já auxiliam na otimização do tráfego urbano, como o *Waze* e o *Google Maps* (EFAGUNDES, 2019a).

O uso de forma eficiente da inteligência artificial resolve os problemas de transporte, principalmente no gerenciamento de tráfego, segurança no trânsito, transporte público e mobilidade urbana, resultando na melhoria da mobilidade humana. As técnicas de inteligência artificial estão, praticamente, dominadas e aptas a resolver a maioria dos problemas de mobilidade nas cidades. O grande desafio é a obtenção dos dados, tanto dos órgãos do governo como das empresas privadas que operam os diversos modais nas cidades (EFAGUNDES, 2019a).

2.4 Desenvolvimento de Aplicativos Móveis

Desenvolvimento de aplicativos móveis é uma área fascinante e em constante evolução, que combina criatividade, tecnologia e experiência do usuário. Processo de criação de softwares que rodam em dispositivos móveis (*smartphones, tablets, smartwatches*). Pode envolver diferentes plataformas: *Android, iOS* e até soluções híbridas que funcionam em ambas.

A evolução da tecnologia dos aparelhos celulares permitiu oferecer ao usuário recursos que vão muito além da realização de uma chamada ou do envio de uma mensagem. As melhorias de hardware dos aparelhos celulares permitiram o desenvolvimento de sistemas operacionais mais avançados. Com sistemas operacionais mais avançados foi possível desenvolver aplicativos melhores, com cada vez mais recursos e serviços ao usuário. Os acessos a serviços de instituições financeiras e redes sociais, por exemplo, podem ser facilitados pelo uso de aplicativos que são executados em aparelho celular. Devido a esta evolução, um aparelho celular se transformou em

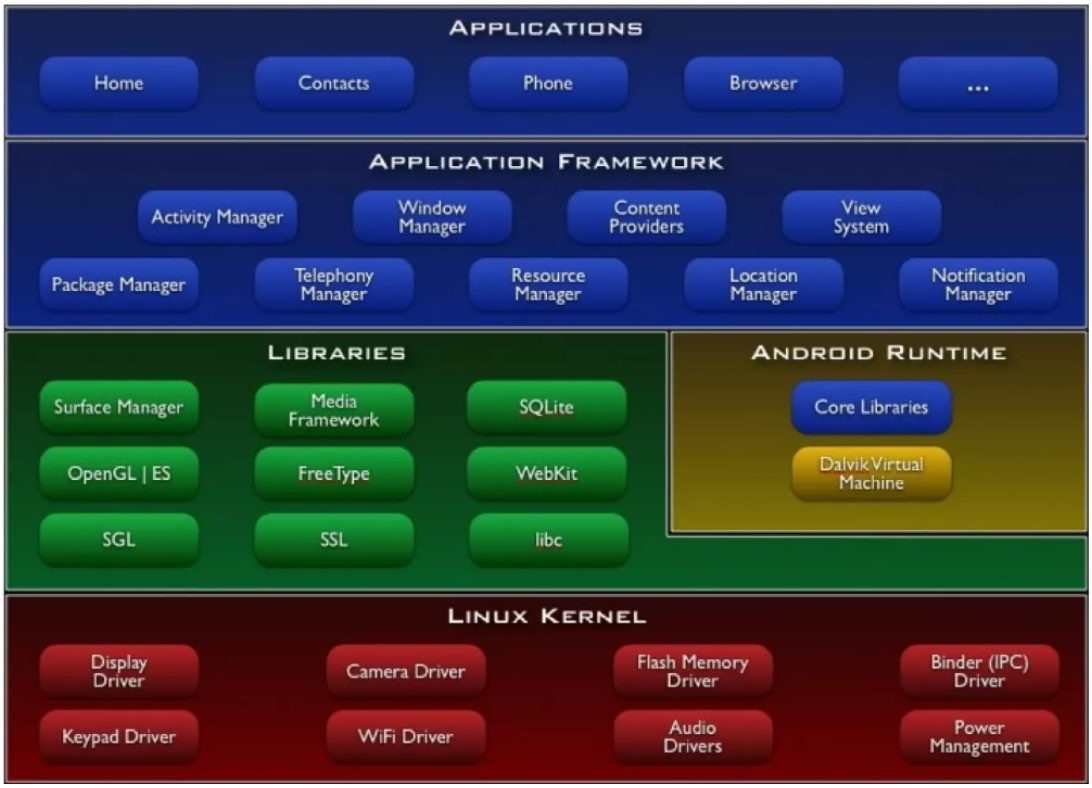
uma oportunidade de entretenimento, acesso à informação e solução de problemas, integrando-se assim ao cotidiano das pessoas e facilitando diversas tarefas do dia a dia (SILVA; SANTOS, 2014).

2.4.0.1 Arquitetura de Sistemas Móveis (*Android/iOS*)

Um sistema computacional moderno é composto por vários componentes: um ou vários processadores, disco rígido, impressoras, teclado, monitor, entre outros. Os sistemas operacionais são softwares complexos e responsáveis pelo gerenciamento dos componentes fundamentais para o funcionamento de um computador. Além de gerenciar esses componentes, eles fornecem uma interface gráfica que facilita a vida do usuário, tornando sua interação com o hardware mais simples. Os sistemas operacionais são classificados em vários tipos, como: Servidor, Grande Porte, Embarcado, Tempo Real, entre outros (TANENBAUM, 2009).

O Android é um sistema operacional móvel *open source*, desenvolvido inicialmente pelo Google, e possui uma arquitetura baseada na versão 2.6 do *kernel Linux*, responsável pelo controle das principais tarefas do sistema, como segurança, gerenciamento de memória, gerenciamento de processos, pilha de rede e modelo de *drivers* (GOOGLEINC, 2011), assim como representado abaixo na Figura 3.

Figura 3 – Arquitetura do Android



Fonte: (GOOGLEINC, 2011)

iOS é a abreviação de *iPhone Operating System*, desenvolvido pela Apple. O sistema

foi baseado no sistema operacional *macOS X* e projetado para atender às necessidades dos aparelhos móveis da *Apple*. O sistema realiza uma abstração entre a comunicação do *hardware* e o aplicativo. A arquitetura do *iOS* é composta por quatro camadas (APPLEINC, 2011), assim como representado abaixo na Figura 4.

Figura 4 – Arquitetura do iOS

Fonte: (APPLEINC, 2011)

A Tabela 1, expõe para melhor compreensão de cada camada, permite visualizar suas funções de forma organizada e comparativa, mostrando como cada camada da aplicação possui responsabilidades distintas, porém interligadas, no fluxo de dados e no processamento.

Camada	**Função Principal**
Cocoa Touch	UI e interação
Media	Áudio, vídeo e gráficos
Core Services	Lógica e serviços essenciais
Core OS	Kernel, drivers e segurança

Tabela 1 – Funções de cada Camada do iOS, (Fonte: Elaborada pelo Autor, 2025.)

2.4.1 Frameworks e Possíveis Linguagens a Serem Utilizadas (Flutter, React Native e Kotlin)

A utilização de *frameworks* baseados em linguagem de padrões para apoiar a reengenharia de sistemas legados é uma linha de estudo em aberto, que pode facilitar a migração de sistemas legados e pode evitar os problemas criados quando da utilização de conversores de linguagem e

quanto à carência de ferramentas voltadas para tal migração. Além disso, o produto obtido após o processo de reengenharia, utilizando *framework*, possui boa qualidade, considerando que o *framework* tenha sido validado durante sua construção. Essa característica do produto é difícil de ser alcançada quando se utilizam conversores de linguagem, como declarado por (TEREKHOV; VERHOEF, 2002). Apresentamos um estudo comparativo entre *flutter, React Native e Kotlin* para desenvolvimento mobile na Fabela 2.

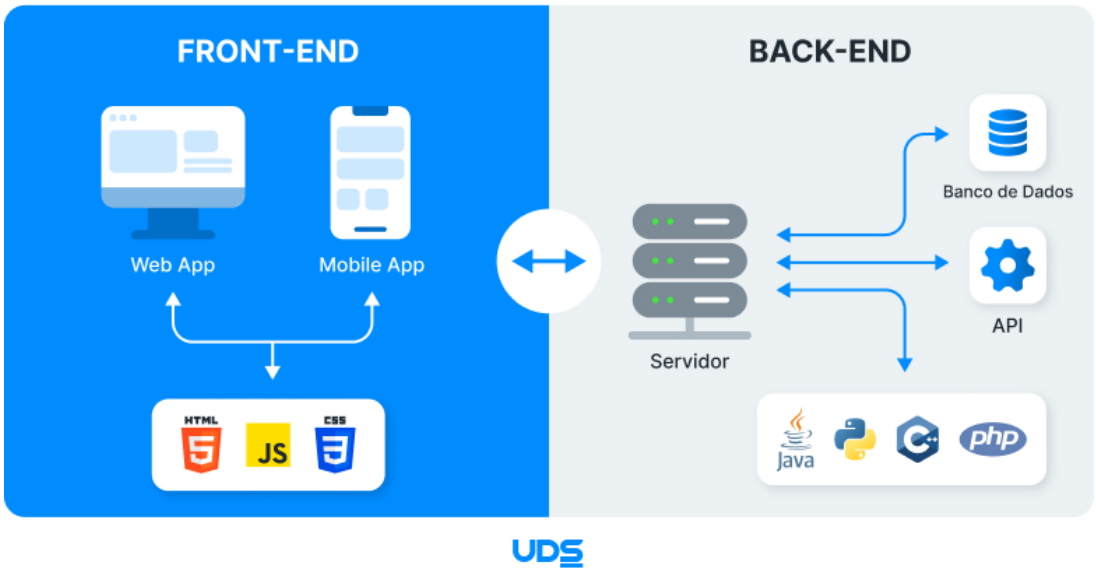
Característica	Flutter	React Native	Kotlin (Android Nativo)
Linguagem	Dart	JavaScript / TypeScript	Kotlin
Plataforma	Multiplataforma	Multiplataforma	Android (nativo)
Desempenho	Alto (engine própria)	Médio-alto (bridge nativo)	Máximo (nativo)
Interface (UI)	Própria (widgets Flutter)	Nativa (via bridge)	Nativa (XML / Jetpack Compose)
Hot Reload / Live Reload	Sim	Sim	Sim (com Compose)
Suporte / Manutenção	Google	Meta (Facebook)	Google / JetBrains

Tabela 2 – Comparativo entre Flutter, React Native e Kotlin para Desenvolvimento Mobile, (Fonte: Elaborada pelo Autor, 2015).

2.5 Conceitos de *Back-end* e Integração de APIs

O *back-end* representa a lógica e o processamento da aplicação, incluindo autenticação, persistência de dados, controle de sessões e integração com sistemas externos (RAJABOV, 2023). Um componente central do *back-end* moderno é a implementação de APIs, que funcionam como pontos de acesso a funcionalidades específicas do sistema e permitem a comunicação entre diferentes componentes e aplicações. Podemos verificar na Figura 5 a diferença entre Front-End e Back-End.

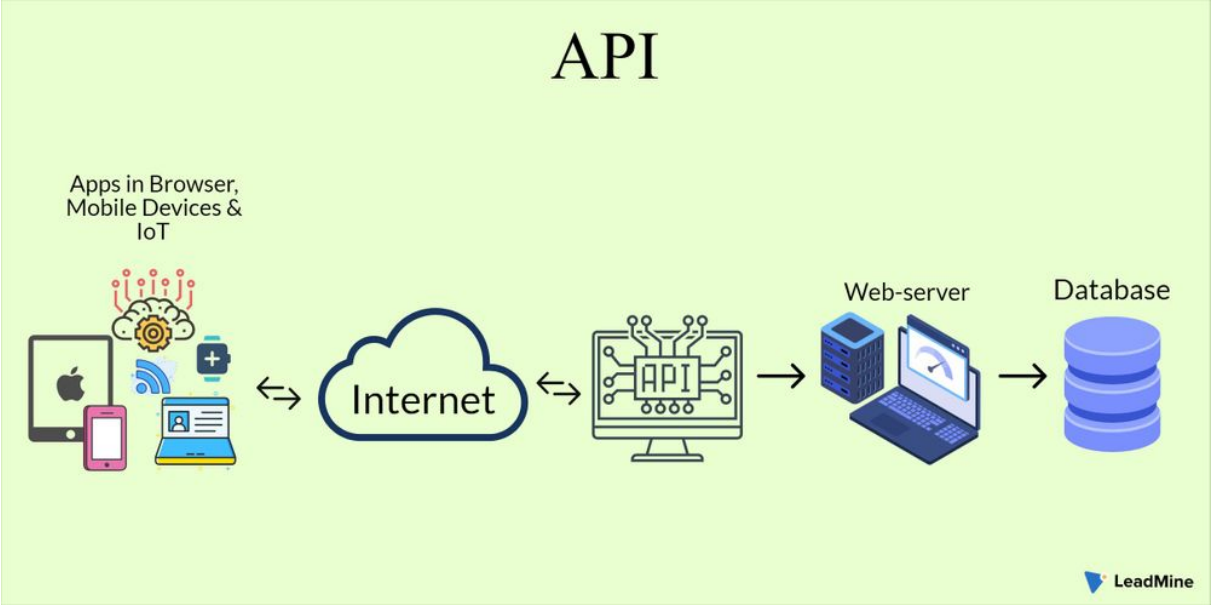
Figura 5 – Comparativo entre *Front-end* e *Back-end*.



Fonte: (TECNOLOGIA, 2024)

API é um contrato que define como sistemas conversam entre si. As APIs tornaram-se elementos fundamentais no desenvolvimento de soluções tecnológicas, permitindo a integração eficiente entre sistemas e facilitando a interoperabilidade entre diferentes plataformas. As APIs não apenas otimizam processos internos das organizações, mas também desempenham um papel estratégico ao possibilitar novas formas de monetização e criação de valor. Elas funcionam como uma ponte invisível para o usuário final, conectando serviços e garantindo a comunicação segura e estruturada entre aplicações e bancos de dados (JUNIOR, 2024). Conformem demostrado o funcionamento da API na Figura 6.

Figura 6 – Funcionamento de uma API.



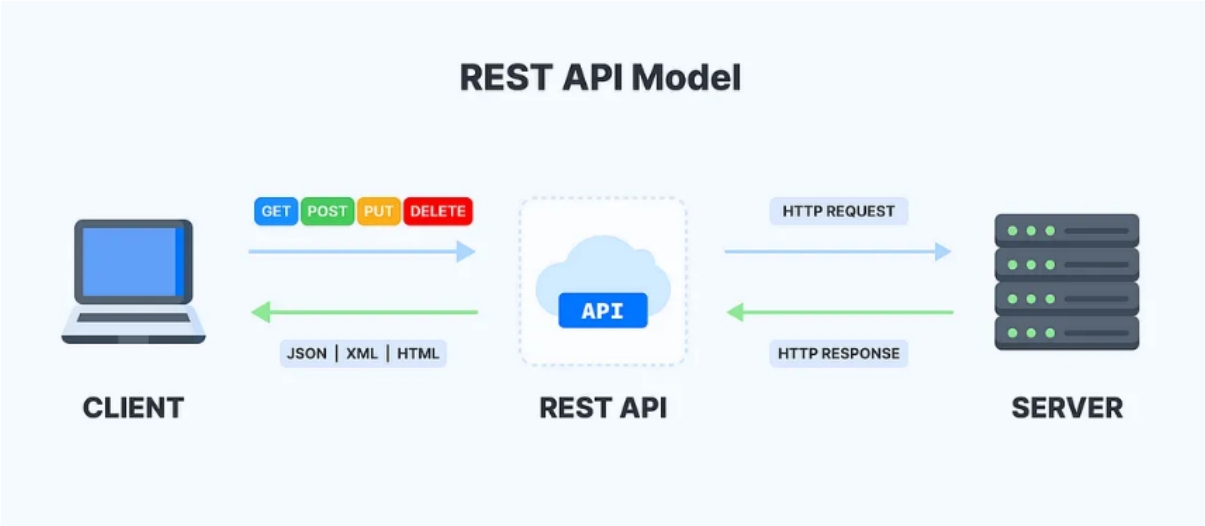
Fonte: (LEADMINE, 2021)

2.5.0.1 REST

A arquitetura REST (*Representational State Transfer*) é um estilo de arquitetura para sistemas distribuídos que define princípios simples e padronizados para comunicação entre cliente e servidor usando o protocolo HTTP. Não é uma linguagem ou tecnologia, mas um modelo arquitetural criado por *Roy Fielding* em sua tese de doutorado no ano 2000, amplamente utilizado por sua simplicidade e aderência ao protocolo HTTP. Nesse modelo, os recursos são apresentados por meio de URLs, enquanto métodos HTTP como GET, POST, PUT e DELETE são empregados para a manipulação. Sua leveza e compatibilidade com sistemas heterogêneos o tornaram a principal escolha no desenvolvimento de APIs modernas, favorecendo escalabilidade e integração entre diferentes plataformas. As APIs RESTful utilizando métodos HTTP, como GET, POST, PUT, DELETE e PATCH, permitindo operações padronizadas para leitura, criação, atualização e remoção de recursos. Esse modelo promove interoperabilidade, consistência e escalabilidade, sendo uma prática consolidada no desenvolvimento de sistemas web e distribuídos

(FIELDING, 2000). A Fgura 7 explica resumidamente para nós o funcionamento de uma API REST.

Figura 7 – Funcionamento de uma API REST.

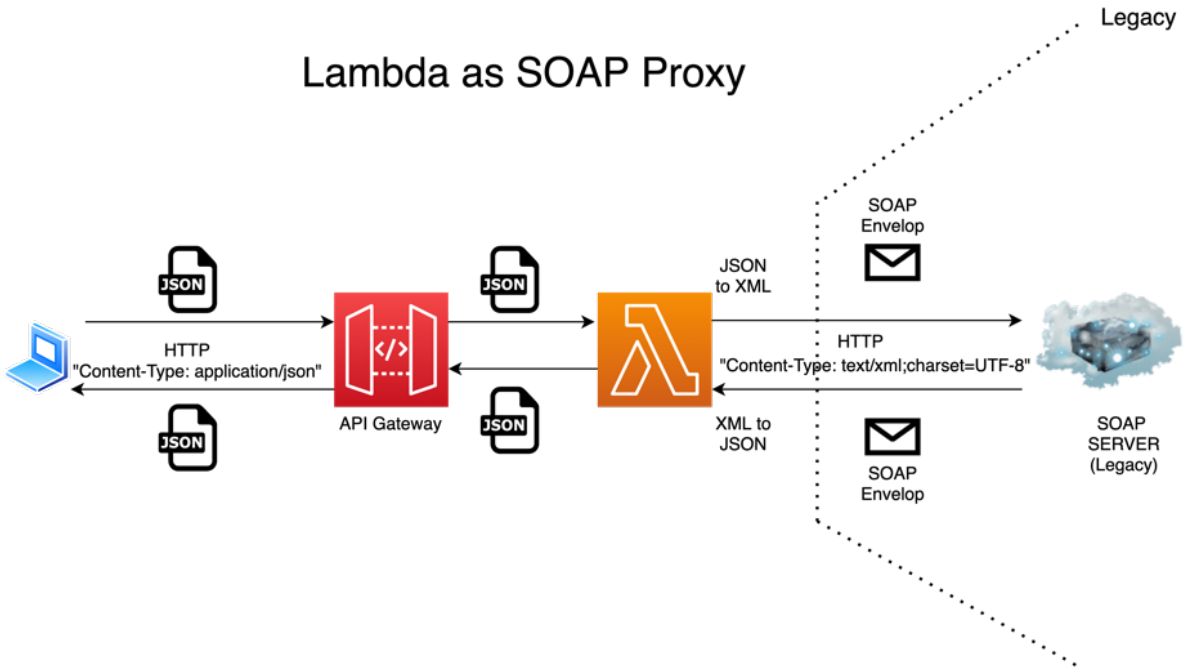


Fonte: (LEFUNDES, 2024)

2.5.0.2 SOAP

O Simple Object Access Protocol (SOAP) é um protocolo baseado em XML, padronizado pela W3C, que garante forte tipagem, suporte a transações complexas e alto nível de segurança. Apesar de ser mais rígido que o REST, o SOAP é amplamente empregado em domínios que exigem confiabilidade e segurança, como sistemas financeiros e governamentais. Embora mais complexo, o SOAP continua sendo uma escolha relevante em contextos nos quais a padronização e a robustez das transações são fatores determinantes (GOMES; JR, 2009). Abaixo temos a Figura 8 representando o diagrama a seguir representa a arquitetura para um SOAP Proxy Server usando os serviços da AWS.

Figura 8 – SOAP Proxy Server.

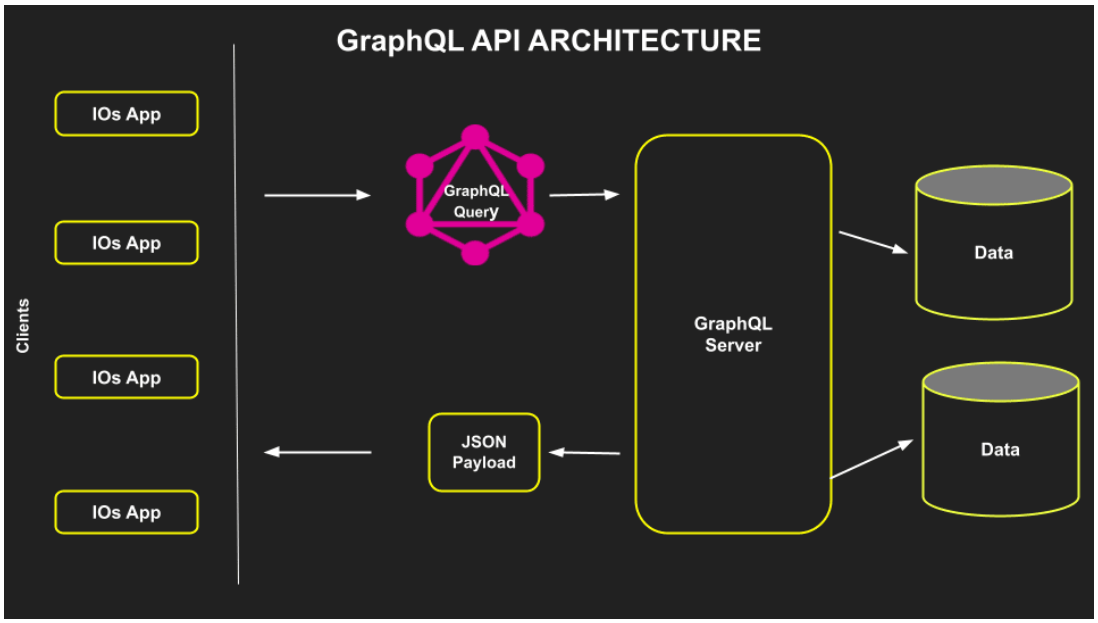


Fonte: (ABIB, 2022)

2.5.0.3 GraphQL

O GraphQL, desenvolvido pelo Facebook, surgiu como uma alternativa para a construção de APIs, oferecendo maior flexibilidade em relação a tecnologias tradicionais, como REST e SOAP. Essa tecnologia permite que o cliente especifique exatamente quais informações deseja receber em uma única requisição, tornando o processo de comunicação entre cliente e servidor mais eficiente. O GraphQL tem se mostrado especialmente útil em aplicações modernas, incluindo sistemas móveis e aplicações com interfaces ricas, nas quais a agilidade na obtenção de dados é um fator importante (RIBEIRO, 2019). A figura 9 representa para nós a Arquitetura GraphQL API, resumindo assim o seu funcionamento.

Figura 9 – Arquitetura GraphQL API.



Fonte: (ESEME, 2025)

2.5.0.4 Integração de APIs

Integrar uma API significa conectar um sistema ao outro, permitindo que compartilhem dados ou funcionalidades. A Fabela 3, resume o conceito de back-end e integraçãO de APIs.

Conceito	O que é	Papel principal
Backend	Parte lógica e invisível do sistema	Processar dados, aplicar regras de negócio e se comunicar com o banco de dados
API	Interface de comunicação entre sistemas	Permitir integração e troca de dados
Integração de APIs	Conexão entre diferentes sistemas via APIs	Fazer com que aplicações se comuniquem de forma automatizada

Tabela 3 – Conceitos de Back-end e Integração de APIs, (Fonte: Elaborada pelo Autor, 2015)

2.5.1 Algoritmo K-nearest Neighbors KNN

2.6 Algoritmo K-Vizinhos Mais Próximos (KNN)

O algoritmo *K-vizinhos mais próximos* (KNN) (COVER; HART, 1967) é um método não paramétrico simples e amplamente utilizado para tarefas de classificação e regressão. Trata-se de um algoritmo baseado em instâncias — também conhecido como *aprendizado lento* — que não realiza o treinamento explícito de um modelo antes da etapa de predição. Em vez disso, o

processo de aprendizagem ocorre apenas no momento da classificação, quando novas amostras são apresentadas para serem classificadas (KOTSIANTIS et al., 2007).

O princípio fundamental do KNN consiste em comparar uma amostra de teste com um conjunto de k amostras vizinhas do conjunto de treinamento, considerando um espaço de atributos. A classe da amostra é então determinada a partir da categoria predominante entre esses k vizinhos mais próximos. A seleção dos vizinhos é normalmente realizada por meio de uma métrica de distância, sendo a distância euclidiana uma das mais utilizadas (ALTMAN, 1992). Assim, a predição é obtida por votação majoritária das amostras vizinhas (KOTSIANTIS et al., 2007); (WITTEN et al., 2017).

O valor de k exerce grande influência sobre o desempenho do modelo: valores muito altos podem incorporar vizinhos distantes e ruidosos, gerando resultados imprecisos; já valores muito baixos podem tornar o modelo instável e mais suscetível a *overfitting*. Portanto, a escolha adequada de k é essencial para cada aplicação, garantindo equilíbrio entre vieses e variância do modelo (FRIEDMAN, 2009).

O K-vizinho mais próximo, também conhecido como KNN, é uma das formas mais simples de algoritmo de ML supervisionado , usado tanto para problemas de classificação quanto de regressão. O KNN é considerado um algoritmo não paramétrico, o que significa que não são feitas suposições sobre os dados subjacentes (COVER; HART, 1967).

Matematicamente, a distância entre os pontos geralmente é calculada por meio da distância euclidiana, definida pela Equação 2.1:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

onde, x um ponto em um espaço de características (feature space) e y outro ponto no mesmo espaço e ambos representam vetores de atributos.

No contexto deste trabalho, o KNN é utilizado para recomendar postos de combustíveis considerando dois atributos principais:

- Distância geográfica entre o usuário e os postos;
- Nível de congestionamento nas vias de acesso.

Assim, o algoritmo poderá indicar ao usuário o posto mais viável em termos de deslocamento, combinado proximidade e fluidez do tráfego.

O algoritmo KNN funciona com base na proximidade similar, utilizando cálculos de distância. As seguintes etapas são utilizadas no desenvolvimento de um modelo KNN: Determine o número de vizinhos mais próximos, também conhecido como K. Por exemplo, se K = 3. A seguir, descrevemos o funcionamento do método KNN (K-Nearest Neighbors) aplicado ao

conjunto de pontos do grafo, seguindo as cinco etapas formais do algoritmo. Aqui, cada ponto é conectado aos seus três vizinhos mais próximos segundo a distância euclidiana.

Etapa 1: Por exemplo, se K = 3, três dos pontos mais próximos com base no cálculo de distância serão escolhidos para determinar onde uma instância será atribuída (em um problema de classificação). Selecionar K pode ser desafiador em um algoritmo KNN. Escolher um K pequeno significa uma influência maior no resultado. Por outro lado, escolher um K maior pode levar a um limite de decisão mais suave com menor variância, mas maior viés. Uma abordagem para selecionar K é treinar um modelo com vários K vizinhos, como 1, 2 ... etc., para ver qual K resultaria na maior precisão de teste usando uma técnica de validação cruzada discutida no capítulo de validação de modelo. Antes de prosseguir para a próxima etapa, certifique-se de padronizar os dados.

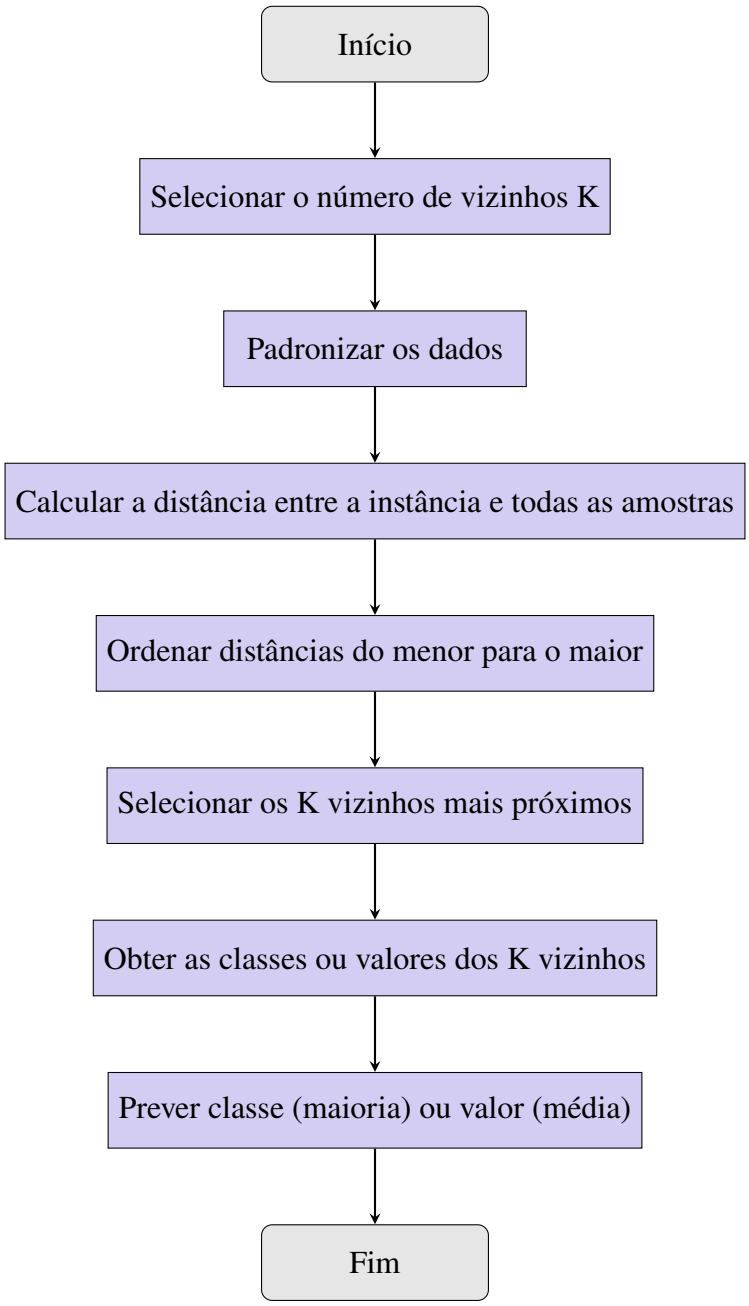
Etapa 2: Use uma função de distância, como a euclidiana, para calcular a distância entre cada instância e todas as amostras de treinamento.

Etapa 3: Em seguida, classifique a distância do menor para o maior. Em seguida, escolha o(s) vizinho(s) mais próximo(s) com base no número de vizinhos mais próximos (K) selecionado na Etapa 1. Em outras palavras, escolha K vizinhos com a menor distância.

Etapa 4: Obtenha a categoria (classificação) ou valor numérico (regressão) dos vizinhos mais próximos obtidos na Etapa 3.

Etapa 5: Use a média (em problemas de regressão) ou a maioria (em problemas de classificação) dos vizinhos mais próximos para prever um valor ou uma classe de uma instância.

Figura 10 – Fluxograma do algoritmo KNN



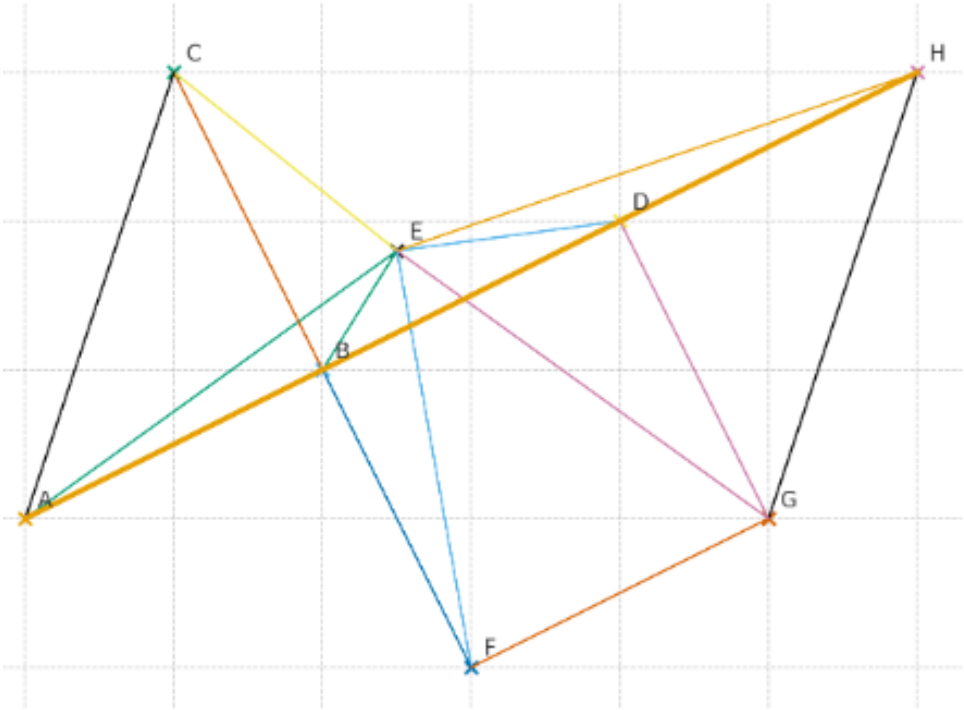
Fonte: Elaborada pelo Autor, 2025.

Neste trabalho, exploramos uma ideologia que demonstrar que, embora os algoritmos tradicionais de determinação de rotas — como o cálculo do menor caminho em grafos ponderados — identifiquem trajetos geometricamente curtos entre dois pontos, isso não garante que tais caminhos sejam os mais eficientes ou práticos no contexto real. Em especial, analisamos o percurso entre os pontos A e H, representada na Figura 11, o grafo estudado e mostramos que a noção de menor distância pode divergir significativamente da noção de melhor rota.

Dessa forma, mesmo que visualmente ou ideologicamente conforme representa a Figura

11 que o caminho A → B → D → H, por exemplo, seja o menor em termos métricos, na prática ele pode ser inviável em determinados momentos. Uma rota alternativa — ainda que mais longa geometricamente — pode apresentar melhor fluidez, menor tempo de deslocamento e maior segurança operacional. Assim, demonstramos que a tomada de decisão baseada exclusivamente em distância ignora o caráter dinâmico e complexo das redes de tráfego.

Figura 11 – Representação Gráfica do Grafo.



Fonte: Elaborada pelo Autor, 2025.

2.7 Trabalhos Relacionados

Dentre os trabalhos relacionados, o estudo de (DOOLAN; MUNTEAN, 2016), propõe um sistemas de transporte ecológico e inteligente (EcoTrec), que é uma solução de roteamento veicular baseada em VANET para redução das emissões de carbono. O EcoTrec foi avaliado em diversos cenários, variando a taxa de penetração e a taxa de conformidade, em diferentes tipos de mapa, incluindo mapas de dimensões urbanos e rurais, e simulações de tráfego, abordando situações em dias e horários diferentes. O algoritmo EcoTrec proposto, se implementado, poderia melhorar muito o meio ambiente, reduzindo as emissões de gases e aumentando o bem-estar geral da sociedade, diminuindo o tempo perdido devido ao congestionamento do tráfego (DOOLAN; MUNTEAN, 2016).

Já (MENESES; LEANDRO; LOUREIRO, 2003) trouxe no seu trabalho como objetivo de estudo principal apresentar uma discussão teórica e prática sobre a definição e o uso de indicadores de desempenho para avaliar objetivos de sistemas de gestão do tráfego urbano.

Inicialmente, são apresentados conceitos básicos sobre a definição de indicadores de desempenho para aferir a eficiência e a eficácia da gestão do tráfego urbano. Em seguida, são descritos os principais objetivos gerenciais do sistema CTA de Fortaleza e seus respectivos indicadores de desempenho, incluindo formulação matemática e exemplos de aplicação prática, implementados numa base SIG, o que permitiu agregar o atributo espacial e facilitar a compreensão dos resultados obtidos com a determinação destes indicadores.

Neste trabalho (DOURADO; CAMPOS, 2007) apresenta um estudo sobre técnicas utilizadas para o gerenciamento da demanda de tráfego em vias urbanas com enfoque para utilização da informação em tempo real para o usuário. Procurou-se, assim, descrever as diferentes técnicas e sistemas utilizados para mostrar a aplicabilidade dos mesmos no gerenciamento da demanda em locais onde os sistemas eletrônicos de controle de tráfego possibilitam a obtenção de dados do tráfego em tempo real para gerar informações importantes para o usuário das vias.

Visando contribuir para o monitoramento em tempo real do tráfego urbano em cidades brasileiras, (LOUREIRO; GUEDES; FIGUEIREDO, 2025) explora uma visão trabalhista que apresenta uma solução inteligente que utiliza imagens de câmeras de monitoramento urbanas para treinar modelos inteligentes de contagem e classificação de veículos. O modelo YOLOv11 Medium obteve mAP experimental de 0,7372 a 94 FPS, e foi implantado em um dashboard interativo. Este trabalho também propõe um benchmark para a tarefa de detecção de veículos e contribui com estratégias adaptativas e responsivas para a mobilidade urbana inteligente em cidades brasileiras.

Uma solução para sistemas de transporte inteligentes que coleta informações em tempo real, detecta e gerência o congestionamento é proposto em (BRENNAND et al., 2015). O sistema proposto utiliza um conjunto de RSU distribuídas na cidade com capacidade de obter uma cobertura de comunicação de todas as ruas. Cada RSU é responsável por gerenciar e detectar congestionamentos de veículos em sua área de cobertura. O sistema proposto utiliza uma mensagem de controle que, periodicamente, realiza a atualização de todas as rotas do veículos para que os mesmos não encontrem congestionamentos. Entretanto, o sistema proposto não gerência o congestionamento quando o mesmo acontece, já que as rotas serão atualizadas na fase de atualização de rotas.

E (SILVA, 2025) apresenta uma abordagem para a coleta e processamento de dados em tempo real do tráfego urbano, com o objetivo de ajustar dinamicamente os tempos dos semáforos com base nas condições atuais do tráfego. A metodologia proposta utiliza a biblioteca OpenCV em um ambiente Python para capturar imagens do tráfego, que são processadas por uma rede neural YOLO para extrair informações, como os tipos e a quantidade de veículos que passam nas vias. Em seguida, foi desenvolvido um algoritmo para ajustar os tempos dos semáforos dinamicamente, de acordo com as condições de tráfego em tempo real. A eficácia do sistema foi avaliada através de um simulador contendo um cruzamento com quatro vias e por meio de simulações que compararam um sistema de semáforo dinâmico com um sistema estático. Os

resultados da simulação demonstraram que o sistema dinâmico reduziu o tempo médio de espera dos veículos nas interseções e aumentou o número de veículos que atravessam a interseção por unidade de tempo, destacando a importância da ciência de dados em tempo real na criação de sistemas de controle de tráfego inteligentes, capazes de atender às demandas de uma infraestrutura urbana moderna e eficiente.

Outra solução para a aquisição em tempo real de informações sobre veículos é proposto em (PAN et al., 2012). A solução proposta, projetada de maneira centralizada, obtém informações sobre a localização geográfica dos veículos, suas velocidades e direção do percurso para detectar congestionamentos. Quando o veículo se aproxima de um congestionamento detectado, o sistema calcula uma nova rota baseada em dois algoritmos diferentes, que são: Dynamic Shortest Path (DSP), onde as rotas são calculadas combinando o menor caminho e o menor tempo de viagem e Random k Shortest Paths (RkSP), onde k rotas são calculadas e aleatoriamente o sistema escolhe uma rota para o veículo. O objetivo da escolha aleatória entre k rotas é não causar novos engarrafamentos em outros pontos da cidade. O sistema proposto em (PAN et al., 2012) possui a mesma desvantagem do sistema proposto em (BRENNAND et al., 2015), onde as novas rotas são calculadas apenas na fase de atualização de rotas.

(THOMÉ et al., 2020) Afirma que os dados coletados por sensores, câmeras, redes sociais e aplicativos podem contribuir para a detecção automática de eventos atípicos no trânsito. A natureza heterogênea dessas fontes de dados traz como vantagem a redundância de informação, aumentando o grau de confiabilidade dos eventos detectados, então sugere no seu trabalho uma proposta de um arcabouço para detecção de eventos anômalos e emissão de alertas em tempo real, com uma interface capaz de integrar fontes de dados heterogêneas. Para isso, ao receber os eventos de uma via, o arcabouço os organiza em séries temporais diárias. Em seguida, é aplicada uma técnica de clusterização nessas séries temporais, a fim de criar um histórico padrão que possibilite detectar anomalias e emitir alertas em tempo real. Para validar o arcabouço, foram implementadas interfaces para dados disponibilizados pela Prefeitura de Vitória (ES), provenientes da plataforma Waze e do Twitter, além de um conjunto de algoritmos de clusterização e detecção de anomalias. Utilizando dados reais da cidade, os resultados mostraram que o arcabouço proposto é escalável e que os alertas podem auxiliar os gestores na tomada de decisões.

O presente trabalho diferencia-se ao aplicar o especificamente para recomendação de postos de combustível, integrando dados de geolocalização e tráfego em tempo real em um aplicativo móvel, o que representa uma contribuição original para a área.

3

Referencial Tecnológico.

O presente capítulo descreve as principais ferramentas, tecnologias e recursos computacionais empregados no desenvolvimento da aplicação móvel de apoio à mobilidade urbana, cujo objetivo é direcionar o usuário ao posto de combustível mais próximo, considerando níveis de congestionamento em tempo real.

Serão abordados os aspectos técnicos que fundamentam a construção do sistema, incluindo as linguagens de programação utilizadas no desenvolvimento da aplicação, bem como os frameworks e bibliotecas que otimizam e aceleram o processo de implementação.

Além disso, serão discutidas as tecnologias de geolocalização e os serviços de mapas e rotas integrados à aplicação, que possibilitam o cálculo de trajetos eficientes e a análise do tráfego urbano. Também serão apresentados os bancos de dados responsáveis pelo armazenamento e consulta das informações de localização e dos estabelecimentos, bem como os ambientes de desenvolvimento e plataformas que fornecem a infraestrutura necessária para o desenvolvimento, testes e implantação da solução.

Por fim, este capítulo busca oferecer uma visão abrangente das tecnologias que sustentam a aplicação proposta, evidenciando como sua integração possibilita uma experiência dinâmica e em tempo real para o usuário.

3.1 Linguagens e *Frameworks* Utilizadas no Desenvolvimento da Aplicação

O desenvolvimento da aplicação móvel de apoio à mobilidade urbana foi baseado em linguagens e frameworks que permitem a criação de soluções modernas, responsivas e eficientes. Essas tecnologias proporcionam uma integração eficaz entre os componentes de interface, lógica de negócio e serviços externos de geolocalização e tráfego.

3.1.1 CSS3.

O CSS3 é a linguagem utilizada para definir a aparência visual e o layout de interfaces em aplicações web e móveis híbridas. No desenvolvimento da aplicação de apoio à mobilidade urbana, o CSS3 desempenha papel essencial na criação de interfaces responsivas e intuitivas, garantindo que o sistema se adapte adequadamente a diferentes tamanhos de tela, como smartphones e tablets. Recursos como transições, animações, gradientes e sombras permitem construir um design moderno e agradável, enquanto as mídia queries asseguram uma experiência visual consistente independentemente do dispositivo ou resolução de tela. Dessa forma, o CSS3 contribui para a usabilidade e eficiência da aplicação, tornando a interação do usuário mais fluida e agradável durante a busca pelo posto de combustível mais próximo e com menor congestionamento.

CSS, é a linguagem usada para estilizar páginas web, controlando a aparência visual e o layout. O CSS3 a terceira versão desta linguagem e representa um marco em termos de possibilidade visuais e interatividade. Com o CSS3, os desenvolvedores ganharam a ferramentas poderosas para criar design atraentes e modernos. Ele introduziu recursos como gradientes, sombras, transições, animações e o suporte a fontes personalizadas por meio de @font-face. Essas funcionalidades permitem que sites sejam mais dinâmicos agradáveis visualmente e eficientes. Outro avanço crucial foi o suporte aprimorado ao design responsivo. Com propriedades coo média queries, o CSS3 tornou possível adptar layouts para diferentes dispositivos e resolução de rela, uma necessidade em um mundo onde smartphones, tablets e desktops coexistem como plataforma de acesso à web (LUCCHESI, 2024). A Fabela 4, mostra a aplicação do CSS em diferentes tipos de aplicativos móveis.

Tipo de Aplicativo	**Uso de CSS**	**Observações**
Aplicação Web / PWA	Sim	Utiliza HTML, CSS e JavaScript diretamente no front-end.
Ionic / Cordova / Capacitor	Sim	Frameworks híbridos baseados em tecnologias web (HTML, CSS e JS).
React Native	Parcial	Utiliza uma sintaxe inspirada no CSS para definir estilos via JavaScript.
Flutter	Não	Usa widgets estilizados por código Dart, sem suporte a CSS.
Android Nativo (Java/Kotlin)	Não	Interface definida via arquivos XML, sem uso de CSS.
iOS Nativo (Swift / SwiftUI)	Não	Interface e estilos definidos por código Swift ou arquivos de interface.

Tabela 4 – Uso do CSS em Diferentes Tipos de Aplicativos Móveis, (Fonte: Elaborada pelo Autor, 2015).

3.1.2 HTML5.

HTML é a sigla em inglês para HyperText Markup Language, que, em português, significa linguagem para marcação de hipertexto. O conceito de hipertexto admite um sem-número de considerações e discussões que fogem ao escopo deste livro. Para o bom entendimento das definições, podemos resumir hipertexto como todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos da web. O que torna possível a construção de hipertextos são os links, presentes nas páginas dos sites que estamos acostumados a visitar quando entramos na internet (SILVA, 2008).

O **HTML5** é a linguagem de marcação utilizada para estruturar o conteúdo de aplicações web e móveis híbridas. Na aplicação móvel de apoio à mobilidade urbana, HTML5 é empregado para organizar os elementos da interface, incluindo botões, mapas, menus e painéis de informações sobre postos de combustível e níveis de congestionamento em tempo real. Entre seus recursos avançados, destacam-se a **geolocation** e o **canvas**. O recurso de **geolocation** é essencial para determinar a posição do usuário e calcular rotas até o posto de combustível mais próximo, considerando o tráfego atual. Já o **canvas** permite a exibição de gráficos e indicadores visuais de congestionamento, tornando a interface mais interativa e informativa. Combinado a CSS3 e JavaScript, o HTML5 possibilita criar uma aplicação híbrida e responsiva, que se adapta a diferentes tamanhos de tela, proporcionando uma experiência consistente e intuitiva em smartphones e tablets, e permitindo que os usuários tomem decisões rápidas sobre o melhor trajeto para abastecimento. Temos abaixo na Tabela 5, os principais recursos do HTML5 aplicando à aplicação móvel.

Recurso	**Aplicação no Projeto**
Estruturação de Conteúdo	Organiza elementos da interface, como mapas, menus, botões e painéis de informações sobre postos e tráfego.
Geolocation	Permite determinar a posição do usuário e calcular rotas até o posto de combustível mais próximo, considerando o congestionamento em tempo real.
Canvas	Exibe gráficos, indicadores de congestionamento e visualizações interativas para facilitar a tomada de decisão do usuário.
Formulários e Inputs	Coleta informações do usuário, como preferências de rota ou tipo de combustível desejado.
Multimídia (Audio/Video)	Possibilita alertas sonoros ou instruções visuais em tempo real sobre rotas ou trânsito.

Tabela 5 – Principais Recursos do HTML5 Aplicados à Aplicação Móvel, (Fonte: Elaborada pelo Autor, 2015).

A vantagem de construir uma aplicação utilizando os recursos do HTML5 é sem sombra de dúvida a mobilidade na plataforma de desenvolvimento, utilizando a mesma aplicação nos mais diversos dispositivos que acessam a internet. Esta mobilidade muitas vezes precisa de algumas

adaptações (principalmente em dispositivos móveis) para que sua usabilidade seja mantida. Já é possível encontrar no mercado alguns frameworks (conjunto de funcionalidades que abstraem problemas comuns à diversas aplicações) que trazem boas implementações principalmente para dispositivos móveis, ao qual se destaca o SenchaTouch e o JQuery Mobile. Deve-se finalmente existir acima de tudo um cuidado quanto à utilização dos novos recursos, de maneira que os mesmos não comprometam a navegação para os usuários que possuem navegadores incompatíveis (SCHROEDER, 2012).

3.1.3 Django

(CABECUDO; KAPLAN-MOSS, 2010) diz que quando se dispõe de boas ferramentas de desenvolvimento web, o processo torna-se envolvente e criativo; mas, quando essas ferramentas faltam, ele pode se transformar em uma sequência entediante de ações repetitivas. O Django possibilita concentrar-se nos momentos agradáveis do trabalho — a parte essencial de um aplicativo web — reduzindo a rotina ao mínimo. Para alcançar esse objetivo, ele oferece um conjunto de ferramentas comprovadas para desenvolvimento web: uma abstração de alto nível, instrumentos para executar rapidamente tarefas comuns de programação web e convenções claras que permitem resolver problemas frequentes. Tudo isso permite que você avance rapidamente no desenvolvimento dos seus aplicativos e reduza ao mínimo a necessidade de configurar componentes comuns.

A espinha dorsal é Django para backend, com atualizações em tempo real via WebSockets, ingestão contínua de dados e um motor de roteamento que entende o trânsito como ele está agora. Há referências de dashboards em tempo real com Django Channels, Celery e integrações como MQTT/Kafka que mostram esse stack funcionando no mundo real. No ecossistema brasileiro, há plataformas que monitoram mobilidade em tempo real (como análise de GPS e métricas de velocidade), o que valida a abordagem de dados dinâmicos para sua solução. E a experiência dos apps populares (Moovit, Google Maps, CittaMobi, etc.) mostra que o valor está em combinar fontes e oferecer rotas práticas e acessíveis. Representamos na Tabela 6, o fluxo de funcionamento do aplicativo com o django.

Etapa	Descrição
Usuário abre o app	Django fornece a rota inicial via API, retornando configurações e estado inicial do sistema.
Dados de GPS chegam	O endpoint do Django Rest Framework (DRF) recebe os dados de posição; uma task Celery é acionada para processamento assíncrono.
Processamento no servidor	Celery limpa, agrega e atualiza índices de tráfego com base nos dados recebidos.
Banco PostGIS atualizado	As geometrias e tabelas espaciais são atualizadas e o nível de congestionamento é recalculado por segmento viário.
Envio de alerta via WebSocket	Django Channels emite mensagens para usuários conectados na região afetada, notificando sobre congestionamento.
Motor de rotas	O serviço de roteamento lê os novos pesos dos segmentos e calcula uma rota otimizada com menor congestionamento.

Tabela 6 – Fluxo de funcionamento do Aplicativo com Django, (Fonte: Elaborada pelo Autor, 2015).

3.1.4 Python

(LUTZ, 2001) argumenta que as interfaces de sistema do Python abrangem domínios de aplicação, mas nos próximos quatro capítulos, a maioria dos nossos exemplos se enquadra na categoria ferramentas de sistema — programas às vezes chamados de utilitários de linha de comando, scripts de shell, ou alguma variação desses termos. Independentemente do nome, você provavelmente já está familiarizado com esse tipo de script; eles realizam tarefas como processar arquivos em um diretório, executar scripts de teste, e assim por diante. Esses programas, historicamente, foram escritos em linguagens de shell não portáveis e sintaticamente obscuras, como arquivos de lote do DOS, csh e awk.

Mesmo nesse domínio relativamente simples, porém, alguns dos melhores atributos do Python brilham intensamente. Por exemplo, a facilidade de uso do Python e sua extensa biblioteca embutida tornam simples (e até divertido) usar ferramentas avançadas de sistema, como threads, signals, forks, sockets, e similares; tais ferramentas são muito menos acessíveis sob a sintaxe obscura das linguagens de shell e os ciclos lentos de desenvolvimento das linguagens compiladas. O suporte do Python para conceitos como clareza de código e programação orientada a objetos também nos ajuda a escrever ferramentas de shell que podem ser lidas, mantidas e reutilizadas. Quando se utiliza Python, não há necessidade de começar cada novo script do zero (LUTZ, 2001). Podemos constatar na Tabela 7, as possíveis contribuições do python na nossa aplicação.

Camada	**Função do Python**
Backend	Recebe a localização do usuário, consulta bancos de dados de postos, processa dados de trânsito em tempo real e retorna rotas otimizadas.
Processamento de Dados	Algoritmos de roteamento, análise de congestionamento e previsão de tráfego usando machine learning (NumPy, scikit-learn, TensorFlow).
Integração	Consome APIs externas de mapas e trânsito, atualiza rotas em tempo real e integra informações adicionais, como clima ou eventos de tráfego.
Aplicação Móvel	Desenvolvimento de aplicativos híbridos ou nativos simples usando ionic e angular, integrando a lógica do backend e atualização de dados.

Tabela 7 – Contribuição do Python na Aplicação, (Fonte: Elaborada pelo Autor, 2015).

Ao aplicarmos a mesma lógica apresentada no texto de (LUTZ, 2001) — a de usar uma linguagem de programação clara, simples e com boas bibliotecas para criar ferramentas de sistema — ao desenvolvimento de uma aplicação móvel para mobilidade urbana, o raciocínio fica conforme a Tabela 8, apresentada abaixo:

Etapa / Função	**Descrição**	**Como Python contribui**
Coleta de dados em tempo real	Obter localização do usuário, trânsito, rotas e dados dos postos.	Python facilita chamadas à API, manipulação de dados e atualização contínua com bibliotecas simples como `requests` e `asyncio`.
Processamento e análise	Calcular os postos mais próximos, tempo de chegada e evitar rotas congestionadas.	A clareza e simplicidade do Python ajudam a escrever algoritmos reutilizáveis e fáceis de manter.
Uso de ferramentas avançadas	Atualização constante, trabalho paralelo e processamento assíncrono.	Python oferece `threads`, `sockets` e processos leves, tornando o sistema ágil sem complicação.
Integração com sistemas externos	Conectar-se a serviços de mapas, bancos de dados e APIs de trânsito.	Python possui bibliotecas robustas para comunicação entre sistemas e suporte nativo para web services.
Backend do aplicativo	Processa dados, calcula rotas e envia informações ao app móvel.	Frameworks como `Overpass API` e `Django` permitem criar APIs rápidas, limpas e escaláveis.
Atualizações em tempo real	Enviar notificações ou atualizações instantâneas ao app.	WebSockets permitem comunicação contínua e em tempo real entre servidor Python e o aplicativo.
Aplicativo móvel (front-end)	Interface que mostra rotas, postos e tempos estimados.	Python realiza a lógica pesada no servidor; o app apenas consome os dados.
Manutenção e reutilização	Melhorias contínuas no sistema sem reescrever tudo do zero.	A clareza do código Python facilita expansão, reaproveitamento e manutenção.

Tabela 8 – Abordagem de Desenvolvimento do Aplicativo, (Fonte: Elaborada pelo Autor, 2025).

3.1.5 JavaScript

Autores como (FLANAGAN; MATILAINEN, 2007) dizem que o JavaScript é a linguagem de programação da Web. A ampla maioria dos sites modernos usa JavaScript e todos os navegadores modernos – em computadores de mesa, consoles de jogos, tablets e smartphones – incluem interpretadores JavaScript, tornando-a a linguagem de programação mais onipresente da história. JavaScript faz parte da tríade de tecnologias que todos os desenvolvedores Web devem conhecer: HTML, para especificar o conteúdo de páginas Web; CSS, para especificar a apresentação dessas páginas; e JavaScript, para especificar o comportamento delas. Este livro o ajudará a dominar a linguagem. Na verdade, o nome “JavaScript” é um pouco enganoso. A não ser pela semelhança sintática superficial, JavaScript é completamente diferente da linguagem de programação Java. E JavaScript já deixou para trás suas raízes como linguagem de script há muito tempo, tornando-se uma linguagem de uso geral robusta e eficiente.

O desenvolvimento da aplicação móvel destinada ao apoio à mobilidade urbana, com foco no direcionamento ao posto de combustível mais próximo com menor congestionamento em tempo real, incorpora o JavaScript como elemento central da camada cliente. A escolha

dessa linguagem fundamenta-se na sua ampla aceitação no ecossistema de desenvolvimento móvel multiplataforma, sobretudo em frameworks como React Native e Ionic, que permitem a construção de interfaces interativas e responsivas, compatíveis com dispositivos Android e iOS. No contexto da aplicação, o JavaScript viabiliza a aquisição contínua de dados de geolocalização do usuário por meio das APIs nativas do sistema operacional. Tais informações são transmitidas periodicamente ao backend implementado em Django Rest Framework (DRF), que processa os dados recebidos, integra informações sobre postos de combustível e calcula o nível de congestionamento nos segmentos viários, utilizando algoritmos de roteamento com base em pesos variáveis provenientes do PostGIS. A Tabela 9, trás os resultados da contribuição do javascript na aplicação.

Camada	**Função do JavaScript**
Frontend	Criação de interface interativa, exibição de mapas, seleção de postos de combustível e cálculo de rotas.
Backend	Processamento de dados de trânsito, cálculo de rotas otimizadas considerando distância e congestionamento em tempo real.
Integração	Consumo de APIs de trânsito e mapas, atualização dinâmica de rotas e alertas em tempo real.
Aplicação híbrida	Código único para Android e iOS, acesso a sensores do celular (GPS, acelerômetro) e envio de notificações.

Tabela 9 – Contribuição do JavaScript na Aplicação, (Fonte: Elaborada pelo Autor, 2015).

3.1.6 Node.js

Node.js é um interpretador de código JavaScript que funciona do lado do servidor. Seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade (como um servidor web), com códigos capazes de manipular dezenas de milhares de conexões simultâneas, numa única máquina física. O Node.js é baseado no interpretador V8 JavaScript Engine (interpretador de JavaScript open source implementado pelo Google em C++ e utilizado pelo Chrome). Foi criado por Ryan Dahl em 2009, e seu desenvolvimento é mantido pela empresa Joyent, onde Dahl trabalha (PAULINO, 2018).

Node.js é o motor de alta performance e comunicação em tempo real da aplicação, garantindo que o usuário receba a rota mais rápida e informações de trânsito atualizadas instantaneamente, mesmo com muitos usuários conectados simultaneamente. O Node.js desempenha um papel central no back-end da aplicação móvel de apoio à mobilidade urbana, sendo responsável por gerenciar e processar dados em tempo real para fornecer ao usuário informações precisas sobre os postos de combustível mais próximos com menor congestionamento. A contribuição do node.js pode ser verificada na Tabela 10.

Camada	**Função do Node.js**
Backend	Criação de APIs REST/GraphQL que recebem a localização do usuário, processam requisições simultâneas e retornam os postos de combustível otimizados.
Tempo real	Atualização dinâmica das rotas via WebSockets/Sockets, enviando alertas instantâneos sobre congestionamento ou mudanças na rota mais eficiente.
Integração	Consome APIs externas de mapas, trânsito e postos de combustível, agregando dados de diferentes fontes para fornecer informações atualizadas.
Frontend móvel	Suporte a apps híbridos (React Native, Ionic), envio de notificações push e sincronização de dados em tempo real com o backend.

Tabela 10 – Contribuição do Node.js na Aplicação, (Fonte: Elaborada pelo Autor, 2015).

3.2 Serviços de Geolocalização e Mapas.

A geolocalização basicamente consiste em obter do usuário as suas posições geográficas, basicamente os navegadores podem obter a posição de três formas: **Localização por IP:** método mais comum, através de serviços disponíveis se consulta de onde determinado endereço de IP possivelmente está acessando. Possui uma grande margem de erro, geralmente sendo preciso a nível de região e estado. **Triangulação GPRS:** A posição é obtida através da triangulação das antenas de celulares próximas, bem mais preciso que a localização por IP, pode precisar o bairro em que o usuário está. **GPS:** Método maxeciso (utilizando satélites), tem uma margem de erro média de 10 metros da localização exata do usuário. O desenvolvedor não escolhe que maneira irá utilizar (pois inclusive podem ser implementadas outras), o único controle que se tem é de solicitar "alta precisão", porém deve se ter em mente que o dispositivo que o usuário está utilizando talvez não suporte o mesmo. (SCHROEDER, 2012). Os principais serviços de geolocalização e mapas são representado Tabela 11.

Serviço	**Função**	**Utilidade no Projeto**
Google Maps API	Fornece mapas interativos, geocodificação, rotas e informações de trânsito em tempo real	Exibir mapas da cidade, calcular rotas e indicar trânsito em tempo real
OpenStreetMap	Plataforma aberta com dados geográficos colaborativos	Criar mapas personalizados e rotas alternativas sem custo de licenciamento
TomTom Traffic API	Dados precisos sobre congestionamento e condições de tráfego	Permitir a escolha de rotas com menor tráfego e alertar sobre congestionamentos
APIs de Geolocalização nativas (Android/iOS)	Acessar a posição do dispositivo diretamente	Determinar a localização exata do usuário e integrar com mapas e rotas

Tabela 11 – Principais Serviços de Geolocalização e Mapas (Fonte: Elaborada pelo Autor, 2015).

A integração desses serviços possibilita que aplicações móveis ofereçam funcionalidades de navegação precisas, alertas sobre congestionamentos e sugestões de rotas eficientes, garantindo maior mobilidade e otimização do tempo do usuário.

3.2.1 APIDOG Plataforma Utilizada para Teste da API da Aplicação.

Para (OLIVEIRA, 2025), uma **API** é um conjunto de definições, protocolos e ferramentas que permite a comunicação e integração entre diferentes sistemas de software. Ela define como os componentes de software devem interagir, especificando métodos, parâmetros e formatos de dados que podem ser utilizados para solicitar e fornecer funcionalidades entre aplicações. As APIs possibilitam que desenvolvedores criem programas modulares e escaláveis, permitindo que sistemas distintos compartilhem dados e serviços de forma padronizada e eficiente. Elas são amplamente utilizadas em ambientes web, móveis e corporativos, garantindo interoperabilidade entre plataformas e redução do esforço de desenvolvimento. Se APIs são essenciais no mundo digital interconectado de hoje. Elas servem como pontes, permitindo que diferentes sistemas de software se comuniquem e compartilhem dados de forma eficaz. Desde integrações com redes sociais até transações financeiras complexas, as APIs desempenham um papel fundamental. Mas desenvolver uma API eficiente não se resume a escrever código; envolve escolher o framework certo que alinhe às necessidades do seu projeto.

3.2.1.1 Por Que Usar Frameworks de Desenvolvimento de API?

Usar um framework de desenvolvimento de API pode simplificar significativamente o processo de criação de APIs robustas, escaláveis e seguras. Esses frameworks fornecem componentes e ferramentas pré-construídos, reduzindo a quantidade de código repetitivo que você precisa escrever. Isso significa que você pode se concentrar mais na lógica de negócios e

menos em tarefas repetitivas. Além disso, frameworks costumam vir com as melhores práticas integradas para segurança, desempenho e manutenibilidade, garantindo que sua API seja de alta qualidade (OLIVEIRA, 2025).

O **APIDOG** é uma plataforma de APIs que fornece dados em tempo real para aplicações, permitindo a integração de serviços externos de maneira rápida e eficiente. Na aplicação móvel de apoio à mobilidade urbana, o APIDOG pode ser utilizado para acessar informações sobre tráfego, localização de postos de combustível e outros dados georreferenciados essenciais para o cálculo de rotas mais eficientes. E a utilização do APIDOG, combinada a tecnologias como **HTML5**, **CSS3** e **JavaScript**, permite criar uma aplicação híbrida responsiva, com interface intuitiva, visualização de mapas e interação em tempo real, proporcionando ao usuário informações confiáveis e ações rápidas para o planejamento de sua rota. Testar sua API é um passo crucial no processo de desenvolvimento para garantir que funcione corretamente e atenda a todos os requisitos. O Apidog simplifica os testes de API com sua interface amigável e recursos poderosos.(OLIVEIRA, 2025).

Através da integração com o APIDOG, a aplicação é capaz de:

- Obter dados atualizados sobre o trânsito em tempo real, permitindo calcular rotas com menor congestionamento;
- Localizar postos de combustível próximos à posição atual do usuário;
- Integrar informações de preços e disponibilidade de combustível (quando disponíveis), otimizando a experiência do usuário;
- Fornecer alertas dinâmicos e recomendações de rotas alternativas, garantindo mobilidade eficiente.

A Figura 12 mostra para API utilizada criar e testar funcionamento da nossa APIs aplicada ao nosso aplicativo, nesse caso Overpass API.

Figura 12 – APIdog Usa-se Para Testar a API do Aplicativo

Fonte: (OLIVEIRA, 2025)

3.2.2 Ambiente de Desenvolvimento

O ambiente de de desenvolvimento ou a IDE a utilizarmos é o VSCode e o Git/GitHub pois, juntos e combinados eles resultam no sucesso de controle de versão da nossa aplicação móvel de apoio à mobilidade urbana, direcionando o usuário ao posto de combustível mais próximo com menor congestionamento em tempo real.

Para o desenvolvimento da aplicação móvel de apoio à mobilidade urbana, que direciona o usuário ao posto de combustível mais próximo com menor congestionamento em tempo real, utilizou-se o **VSCode** como IDE principal, juntamente com **Git e GitHub** para controle de versão e colaboração.

3.2.2.1 VSCode

O VS Code é uma IDE moderna e leve, utilizada para escrever, depurar e testar o código da aplicação. Suas principais contribuições para o projeto incluem:

- Suporte a linguagens como JavaScript, Node.js, Python, HTML, CSS e frameworks móveis (Ionic);
- Extensões para integração com bancos de dados (MongoDB, Firebase, PostgreSQL);
- Terminal integrado para execução de servidores, scripts e comandos Git;
- Depuração em tempo real, fundamental para rotas dinâmicas e dados de trânsito;
- Interface customizável e leve, permitindo produtividade em projetos multiplataforma.

O VSCode contribui de forma decisiva para o desenvolvimento desse aplicativo, funcionando como uma plataforma central onde todas as etapas do projeto podem ser realizadas de maneira integrada, rápida e organizada. A primeira contribuição é a sua capacidade de unificar em um único ambiente o desenvolvimento do backend, escrito em Python, e do aplicativo móvel, desenvolvido em Ionic Angular, Flutter ou React Native. Isso evita a fragmentação do trabalho e permite que o programador mantenha foco, eficiência e continuidade ao longo de todo o processo. O VS Code facilita a escrita do código através do IntelliSense, que fornece sugestões inteligentes, reduz erros e aumenta a produtividade na implementação dos algoritmos responsáveis por calcular rotas, analisar congestionamentos e comunicar com APIs externas.

3.2.2.2 Git e GitHub

O Git e o GitHub serão utilizados para controle de versão e colaboração no desenvolvimento da aplicação. Suas principais contribuições incluem:

- Versionamento do código-fonte, permitindo registrar alterações e reverter mudanças se necessário;
- Colaboração entre múltiplos desenvolvedores, integrando funcionalidades como backend, frontend e banco de dados;
- Integração com VS Code para commits, push, pull e branches diretamente da IDE;
- Facilita gerenciamento de versões da aplicação durante testes e implantação em produção.

Na Tabela 12, temos descritos os benefícios da combinação da IDE VSCode e Git/GitHub na aplicação.

Benefício	**Descrição**
Desenvolvimento eficiente	O VS Code permite codificação, teste e depuração em um único ambiente, agilizando o desenvolvimento do backend, frontend e integração com o banco de dados.
Segurança e rastreabilidade do código	Git/GitHub mantém histórico completo das alterações, permitindo reverter mudanças e acompanhar evoluções do projeto.
Colaboração simplificada	Vários desenvolvedores podem trabalhar em paralelo, integrar funcionalidades e gerenciar branches de forma organizada.
Integração contínua	Facilita automação de testes, deploy e integração de novas funcionalidades, garantindo consistência da aplicação.

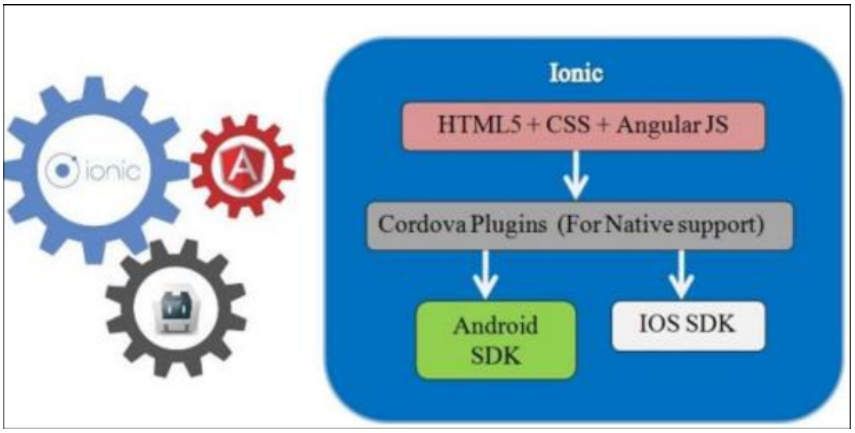
Tabela 12 – Benefícios Combinados da IDE VSCode e Git/GitHub no Aplicação. (Fonte: Elaborada pelo Autor, 2015).

3.3 *Ionic Framework* com Angular para Desenvolvimento de Aplicativos Híbridos

O framework Ionic é uma ferramenta multiplataforma de código aberto que permite o desenvolvimento de aplicações híbridas que funcionam em diferentes plataformas móveis, como Android, iOS e Windows, sem muito esforço e aumentando a facilidade de desenvolvimento. O framework Ionic aprimora a aparência e a experiência do usuário da aplicação híbrida com o uso de HTML5 e CSS. O Angular é um framework de código aberto para aplicações web, usado no desenvolvimento de aplicações baseadas na web, enquanto o Apache Cordova permite que os desenvolvedores criem aplicações móveis usando JavaScript, CSS e HTML, em vez de usar APIs (Interfaces de Programação de Aplicativos) específicas da plataforma, como as do iOS, Android ou Windows Phone (WARANASHIWAR; UKEY, 2018).

O Ionic fornece aos usuários todos os componentes e funcionalidades usados no desenvolvimento nativo para dispositivos móveis – os SDKs (Kits de Desenvolvimento de Software). Os desenvolvedores podem criar suas aplicações usando as ferramentas e os exemplos de código fornecidos pela documentação e pelo site de ajuda do framework Ionic. A instalação requer NodeJS e npm, que é o gerenciador de pacotes padrão para NodeJS, independentemente do ambiente Windows, Linux ou Macintosh. O Ionic Creator é uma plataforma usada para construir interfaces de usuário arrastando e soltando componentes prontos. Foi implementado no design da aparência final do aplicativo. O Ionic View facilita muito a execução e o compartilhamento de aplicativos desenvolvidos em vários dispositivos usando uma interface simples e comum. Antes de serem disponibilizados na loja oficial, os aplicativos são compartilhados no Ionic Market, mediante o pagamento de uma taxa para download. Além disso, o Ionic oferece suporte ao usuário confiável e documentação com informações úteis. O Ionic Lab permite testar aplicativos recém-desenvolvidos em Android e iOS (WARANASHIWAR; UKEY, 2018). Na Figura 26 abaixo temos a arquitetura do Ionic.

Figura 13 – Arquiterura Ionic.



Fonte: (WARANASHIWAR; UKEY, 2018).

3.4 KNN: Principais Componentes

Algorithm 1 Compute KNN Score

Input: stations (lista de postos), k número de vizinhos, include_traffic, max_distance_m
Output: ranking de postos com pontuações KNN

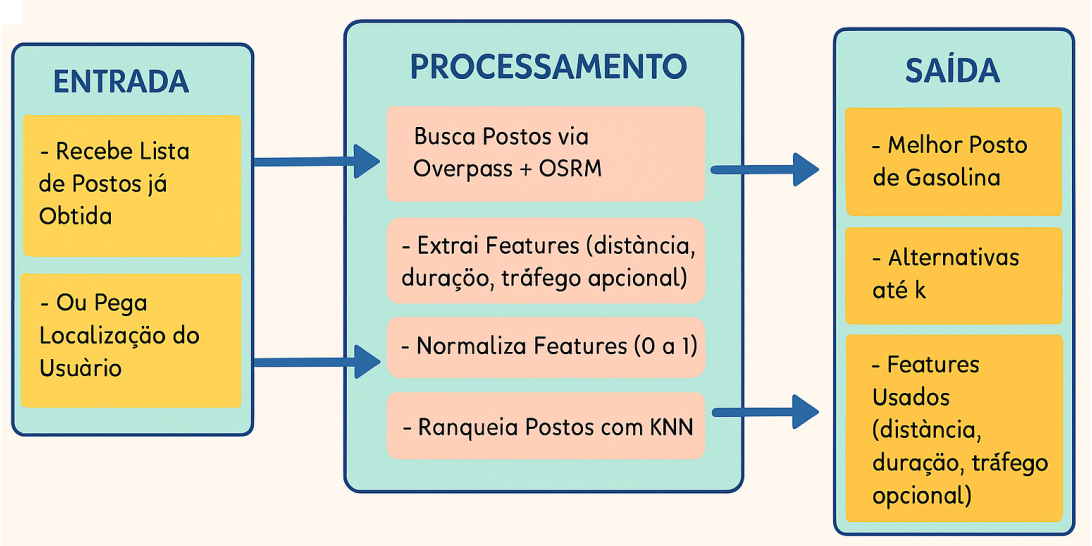
if *stations está vazio* **then**
  return lista vazia
end
filtered_stations ← postos com distance_m ≤ max_distance_m
if *filtered_stations está vazio* **then**
  return lista vazia
end
feature_list ← lista vazia
foreach *station em filtered_stations* **do**
  features ← extract_features_from_station(station, include_traffic) adicionar features em feature_list
end
feature_names ← nomes das features X ← matriz com valores das features
X_{min} ← mínimo por coluna X_{max} ← máximo por coluna
Normalizar:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min} + 10^{-9}}$$

if *include_traffic* **then**
  weights ← [0.5, 0.4, 0.1]
end
else
  weights ← [0.5, 0.5]
end
scores ← $X_{norm} \cdot weights$
ranked_indices ← ordenação crescente de scores
results ← lista vazia
for *idx em ranked_indices* **do**
  station ← filtered_stations[idx] features ← extract_features_from_station(station)
  adicionar em results:
    osm_id, nome, lat, lon, distance_m, duration_s,
    traffic_factor, score, rank, tags
end
return results

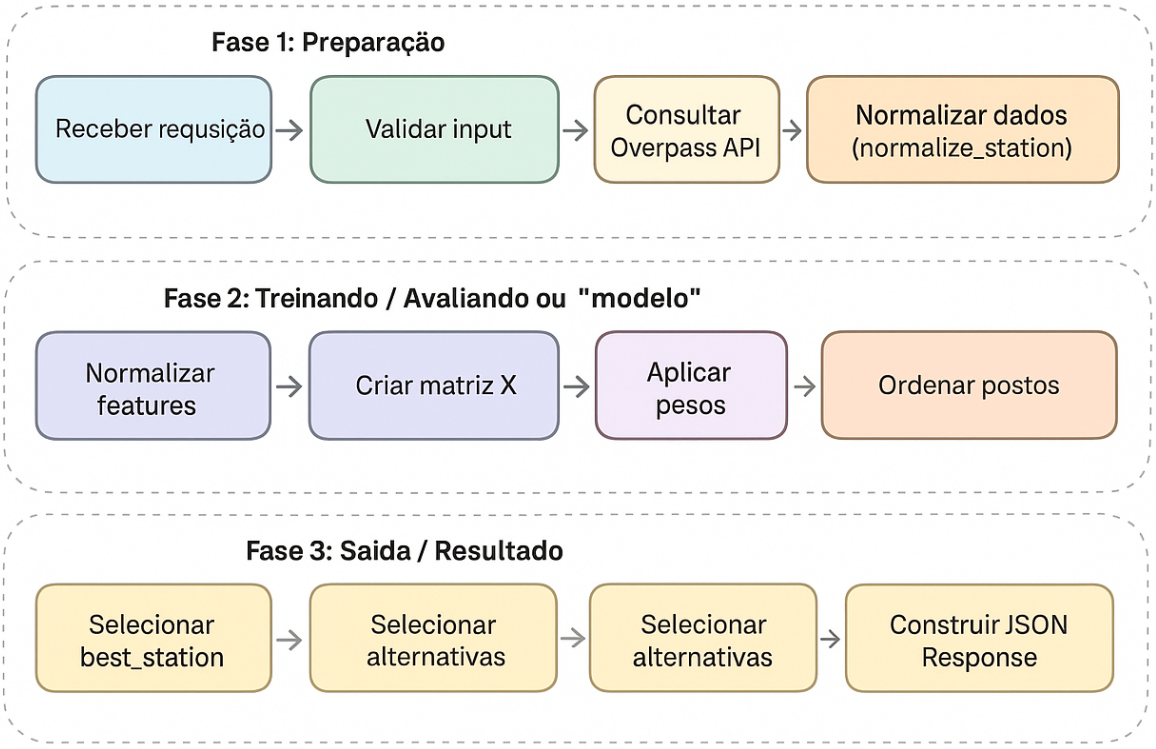
A seguir temos uma Figura 14 onde ilustramos o modelo do funcionamento do KNN na aplicação e a outra Figura 15 que representa as fases do mesmo algoritmo de aprendizado.

Figura 14 – Modelo do Funcionamento do KNN na Aplicação.



Fonte: Elaborda pelo Autor, 2025.

Figura 15 – Fases do Funcionamento do KNN na Aplicação.



Fonte: Elaborda pelo Autor, 2025.

4 Metodologia

A primeira etapa deste trabalho consiste na pesquisa bibliográfica, a qual tem como objetivo reunir, analisar e discutir os principais conceitos relacionados à mobilidade urbana, aos aplicativos móveis e ao uso de algoritmos de aprendizado de máquina, em especial o K-Nearest Neighbors (KNN).

4.1 Metodologia da Pesquisa Teórica (Revisão Bibliográfica)

De acordo com (GIL, 2008), "a pesquisa bibliográfica é desenvolvida a partir de material já elaborado, constituído principalmente de livros e artigos científicos". Esse tipo de pesquisa possibilita ao pesquisador identificar o estado da arte sobre determinado tema, bem como lacunas existentes que podem ser exploradas pelo estudo.

Para a construção do referencial teórico, foram consultadas bases de dados científicas como Google Acadêmico e SciELO, priorizando publicações dos últimos dez anos, mas também incluindo obras clássicas, como a de (COVER; HART, 1967), que introduziu o KNN.

Os critérios de seleção das referências incluem:

1. Relevância para os temas de mobilidade urbana, cidades inteligentes e transporte sustentável;
2. Aplicações de aprendizado de máquina em mobilidade e sistemas de recomendação;
3. Estudos específicos sobre o algoritmo KNN e suas aplicações em problemas de classificação e recomendação;
4. Fontes reconhecidas academicamente, evitando materiais de caráter não científico.

Além de artigos científicos, serão utilizados livros de referência, como (ULIAN, 2022) (2016) sobre mobilidade urbana e (RUSSELL; NORVIG, 1995) sobre inteligência artificial, a fim

de garantir um embasamento sólido e atualizado. Essa etapa teórica serviu como alicerce para a formulação do problema de pesquisa, definição dos objetivos e fundamentação da metodologia prática, assegurando que o desenvolvimento do aplicativo esteja alinhado com o conhecimento científico já consolidado.

4.1.1 Metodologia da Pesquisa Prática (Desenvolvimento do Aplicativo)

Após a revisão bibliográfica, será conduzida a pesquisa prática com foco no desenvolvimento e validação do aplicativo móvel. Esta etapa tem natureza aplicada e experimental, pois visa construir uma solução tecnológica e testá-la em cenários simulados e reais.

1. Etapas da pesquisa prática:
    - Definição de requisitos funcionais (localização via GPS, recomendação de postos, exibição em mapa) e não funcionais (usabilidade, responsividade, baixo consumo de dados).
2. Coleta e Integração de Dados:
    - Posição do usuário obtida via GPS do dispositivo;
    - Dados de tráfego em tempo real obtidos por meio de APIs como Google Maps API, Here Maps ou OpenStreetMap;
3. Implementação do Algoritmo KNN:
    - Definição dos atributos: distância geográfica e nível de congestionamento;
    - Normalização dos dados para garantir comparabilidade;
    - Execução do KNN para identificar o posto mais viável considerando o número k de vizinhos.
4. Desenvolvimento do Aplicativo Móvel:
    - Linguagem e framework: Ionic Angular (JavaScript) para compatibilidade Android/iOS;
    - Backend opcional em Node.js para centralizar requisições;
    - Interface gráfica projetada com foco em usabilidade e simplicidade.
5. Testes e Validação:
    - Testes funcionais em dispositivos móveis;
    - Questionários com usuários para avaliação da usabilidade.

4.2 Levantamento de Requisitos

O levantamento de requisitos é a etapa inicial do processo de desenvolvimento da aplicação móvel proposta, sendo fundamental para identificar as necessidades dos futuros usuários e compreender o funcionamento esperado do sistema. Para isso, foram utilizados métodos qualitativos e quantitativos, por meio da aplicação de questionários, com o objetivo de compreender as principais dificuldades enfrentadas na busca por postos de combustíveis próximos e com menor nível de congestionamento.

Essa etapa permitiu identificar problemas recorrentes, como falta de informações com um padrão de precisão atualizadas sobre o elevado fluxo de tráfego, o que contribui para dificuldade de comparar opções disponíveis nas proximidades e ineficiência no deslocamento devido à escolha de rotas inadequadas. Com base nesses dados, tornou-se possível definir os requisitos funcionais e não funcionais do sistema, assegurando que a aplicação atenda às necessidades reais dos usuários e integre de forma eficiente o algoritmo KNN para recomendação dos postos mais adequados.

4.3 Coleta de Requisitos

4.3.1 Introdução à Coleta de Requisitos

A etapa de coleta de requisitos é fundamental no processo de desenvolvimento de software, pois permite compreender as necessidades dos usuários, identificar o escopo da aplicação e estabelecer as funcionalidades essenciais do sistema. No presente trabalho, cujo objetivo é desenvolver uma aplicação móvel de apoio à mobilidade urbana utilizando o algoritmo *K-Nearest Neighbors* (KNN) para sugerir postos de combustíveis mais próximos e com menor nível de congestionamento, foram adotadas diversas técnicas e ferramentas que garantem precisão e confiabilidade às informações levantadas.

A coleta foi conduzida considerando o perfil dos potenciais usuários, o contexto de mobilidade urbana e os aspectos técnicos necessários para a implementação do algoritmo KNN. As técnicas aplicadas permitiram a identificação de requisitos funcionais, não funcionais e de dados relevantes para o funcionamento do sistema.

4.3.2 Técnicas e Ferramentas Utilizadas

Para garantir uma análise completa das necessidades do público-alvo e das demandas do sistema, utilizou-se uma combinação de técnicas qualitativas e quantitativas conforme descrito nas subseções a seguir.

4.3.2.1 Questionários Online

Com o intuito de obter dados quantitativos, foi elaborado um formulário no Google Forms, divulgado em grupos de motoristas, redes sociais e comunidades de mobilidade. O questionário coletou informações como:

- frequência de abastecimento;
- importância atribuída a fatores como preço, proximidade e congestionamento;
- uso de aplicativos de navegação;
- interesse em sistemas de recomendação de rotas.

As respostas permitiram estimar o perfil do usuário e validar a relevância da aplicação proposta.

4.3.2.2 Observação Direta

Foi realizada observação direta em diferentes horários e regiões da cidade com o objetivo de verificar o comportamento real dos motoristas diante da oferta de postos e da dinâmica do trânsito. Essa técnica possibilitou identificar aspectos que não são facilmente relatados pelos usuários, tais como:

- dificuldade de acesso e saída de determinados postos;
- influência de cruzamentos e semáforos no fluxo de veículos;
- pontos críticos de congestionamento recorrente.

Essas observações auxiliaram na definição de critérios adicionais para o algoritmo KNN que vão além das coordenadas geográficas.

4.3.2.3 Análise de Documentos

Para a extração de dados na Análise Documental é necessário que o pesquisador assuma uma posição ativa na pesquisa e na produção do conhecimento, seguindo passos, quais sejam: como selecionar o material; analisar; organizar e categorizar; ler e reler; sistematizar; descontruir e reconstruir; entre outros (ALVES et al., 2021). Foram analisados relatórios públicos de mobilidade urbana, trânsito e além de informações relacionadas a sistemas de navegação existentes, como o Google Maps e o Waze. A análise documental serviu para:

- identificar padrões de congestionamento;

- compreender metodologias existentes para classificação de tráfego;
- apoiar parâmetros do modelo KNN, como distância, vizinhos mais próximos e métricas de similaridade.

4.3.2.4 Prototipação de Interfaces

A prototipação foi realizada no Canva, onde foram desenhadas telas da aplicação móvel com o objetivo de validar requisitos funcionais junto aos usuários. Os protótipos contemplaram:

- tela de busca por postos;
- postos de combustíveis recomendados.

A prototipação permitiu ajustes na experiência do usuário antes da implementação.

4.3.3 Resultados da Coleta de Requisitos

A partir das técnicas aplicadas, foi possível identificar um conjunto robusto de requisitos funcionais e não funcionais, além de requisitos de dados essenciais para a correta operação do algoritmo KNN. Os resultados incluem:

- funcionalidades principais da aplicação;
- critérios utilizados pelo algoritmo para sugerir postos;
- necessidades reais do usuário em contextos urbanos;
- expectativas quanto à interface e à usabilidade;
- restrições tecnológicas e operacionais.

A coleta demonstrou que existe demanda por uma solução que una informações de localização, tráfego e recomendação inteligente, justificando a relevância e aplicação prática deste trabalho.

4.3.4 Requisitos Funcionais (RF)

Os requisitos funcionais descrevem as funcionalidades que o sistema deve obrigatoriamente executar. A seguir, apresentam-se os requisitos funcionais da aplicação proposta:

- **RF01 – Identificação da localização do usuário:** A aplicação deve obter a localização atual do usuário utilizando serviços de geolocalização.

- **RF02 – Listagem dos postos de combustíveis:** A aplicação deve apresentar uma lista dos postos disponíveis nas proximidades.
- **RF03 – Coleta de dados de congestionamento:** A aplicação deve integrar dados de tráfego e fluxo de veículos por meio de APIs externas ou base interna.
- **RF04 – Processamento dos dados pelo algoritmo KNN:** O sistema deve utilizar o algoritmo KNN para classificar e recomendar os postos com menor congestionamento considerando proximidade e fluxo.
- **RF05 – Ordenação dos postos recomendados:** A aplicação deve ordenar os postos com base em critérios como proximidade e nível de congestionamento.
- **RF06 – Exibir informações detalhadas do posto:** A aplicação deve mostrar dados como endereço, horário de funcionamento, distância e nível estimado de movimentação.
- **RF7 – Avaliação e feedback do usuário:** O sistema deve permitir que os usuários avaliem a recomendação e forneçam feedback.

4.3.5 Requisitos Não Funcionais (RNF)

Os requisitos não funcionais estabelecem como o sistema deve operar, definindo restrições e qualidades esperadas. Seguem os requisitos não funcionais do sistema:

- **RNF01 – Desempenho:** O aplicativo deve processar o algoritmo KNN e exibir recomendações em tempo inferior a 3 segundos.
- **RNF02 – Usabilidade:** A interface deve ser intuitiva e adequada para uso simples e rápido.
- **RNF03 – Confiabilidade dos dados:** A aplicação deve utilizar APIs confiáveis que forneçam dados atualizados de localização e tráfego.
- **RNF04 – Compatibilidade:** O sistema deve ser compatível com dispositivos Android (e iOS, caso implementado).
- **RNF05 – Segurança:** Os dados de localização do usuário devem ser protegidos seguindo boas práticas de privacidade.
- **RNF06 – Disponibilidade:** O sistema deve estar disponível 24 horas por dia, dependendo da disponibilidade das APIs integradas.
- **RNF07 – Escalabilidade:** A aplicação deve permitir a adição de novos dados e funcionalidades sem perda de desempenho.
- **RNF08 – Manutenibilidade:** O código deve ser modularizado para facilitar correções, melhorias e atualizações.

- **RNF09 – Portabilidade:** O aplicativo deve ser facilmente adaptável para outras regiões urbanas mediante substituição dos dados de entrada.
- **RNF10 – Precisão do modelo KNN:** A aplicação deve apresentar um nível mínimo aceitável de acurácia do algoritmo, definido nos testes.

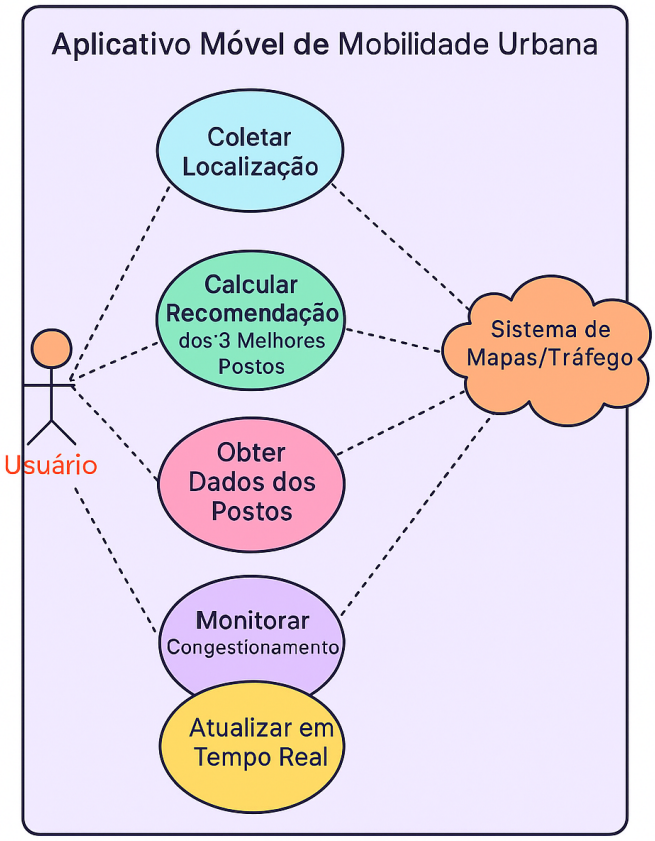
4.4 Modelagem de Requisito

Houve a necessidade de modelar os requisitos do aplicativo baseando-se em diagramas de casos de uso para dar estrutura aos requisitos funcionais da aplicação móvel de apoio à mobilidade urbana. O mesmo diagrama oferece a oportunidade de observar modo específico as interações entre os usuários e o aplicativo, simplificando as funcionalidades essenciais. O caso representa o funcionamento da aplicação ao ser executo pelo usuário. No entanto, o diagrama de caso de uso são relevantes para entender de modo detalhado o sistema e o que o mesmo deve realizar do ponto de vista do usuário final.

A utilização dos diagramas traz diversas vantagens ao processo de desenvolvimento. Em primeiro lugar, eles fornecem uma representação visual intuitiva das funcionalidades da aplicação, o que facilita a comunicação entre desenvolvedores, orientadores e demais envolvidos no projeto. Além disso, contribuem para a identificação de possíveis lacunas, redundâncias ou inconsistências nos requisitos inicialmente levantados, permitindo ajustes antes da fase de implementação. Durante essa etapa de modelagem, foram identificados os principais atores da aplicação: Motoristas, Sistema de Geolocalização e Servidor de Processamento KNN.

Cada ator está associado a um conjunto específico de interações, de acordo com suas responsabilidades e necessidades dentro do sistema. Essa característica é especialmente útil para integração dos diagramas ao texto do TCC e para a reprodução em diferentes mídias. As figuras apresentadas ao longo do trabalho ilustram as relações entre os atores e o sistema, possibilitando a visualização das funcionalidades principais, como indicar os postos de combustíveis mais próximos, calcular rotas com menor congestionamento e processar recomendações utilizando o algoritmo KNN. A Figura 16 a baixo representa os diagramas de uso dos requisitos funcionais e a Figura 17 os requisitos não funcionais as relações entre os atores e a aplicação, permitindo a visualização das atividades que cada etapa pode realizar.

Figura 16 – Diagrama de Uso Requisitos Funcionais



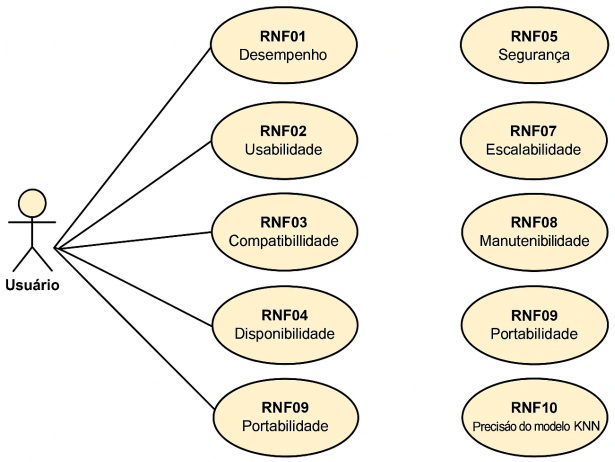
Fonte: Elaborada pelo Autor, 2025.

Descrição dos Casos de Uso

- **Coletar Localização:** O sistema coleta a localização atual do usuário para iniciar o processo de recomendação.
- **Calcular Recomendação dos 3 Melhores Postos:** O aplicativo utiliza algoritmos (como KNN) para identificar os três postos mais adequados com base na proximidade, congestionamento e dados obtidos.
- **Obter Dados dos Postos:** O sistema acessa informações como preços, abastecimento e disponibilidade.
- **Monitorar Congestionamento:** O aplicativo monitora o trânsito em tempo real utilizando dados de mapas/tráfego.
- **Atualizar em Tempo Real:** As informações são atualizadas dinamicamente conforme o usuário se desloca ou os dados mudam.

- **Integração com Sistema de Mapas/Tráfego:** O aplicativo depende de um serviço externo de mapas para obter dados de trânsito.

Figura 17 – Diagrama de Uso Requisitos Não Funcionais



Fonte: Elaborada pelo Autor, 2025.

Descrição dos Requisitos Não Funcionais

- **RNF01 — Desempenho:** O sistema deve responder rapidamente às solicitações do usuário.
- **RNF02 — Usabilidade:** A interface deve ser simples, clara e intuitiva.
- **RNF03 — Compatibilidade:** O aplicativo deve funcionar nos principais sistemas operacionais móveis.
- **RNF04 — Disponibilidade:** O sistema deve estar disponível para uso na maior parte do tempo.
- **RNF05 — Segurança:** Os dados trafegados e armazenados devem ser protegidos.
- **RNF07 — Escalabilidade:** O sistema deve comportar um aumento no número de usuários sem perda de desempenho.
- **RNF08 — Manutenibilidade:** A aplicação deve permitir fácil atualização e correções.
- **RNF09 — Portabilidade:** O sistema deve ser facilmente adaptável a diferentes plataformas.
- **RNF10 — Precisão do modelo KNN:** O algoritmo de recomendação deve atingir boa precisão na seleção dos três melhores postos.

4.5 Arquitetura do Sistema

A aplicação móvel de apoio à mobilidade urbana desenvolvida neste trabalho adota a arquitetura MVC (Model–View–Controller) como base de organização estrutural. Esse padrão foi escolhido por proporcionar uma clara separação de responsabilidades, favorecendo a manutenção, a escalabilidade e a reutilização dos componentes do sistema. Além disso, a abordagem MVC facilita a integração entre a interface do usuário, os dados manipulados pela aplicação e a lógica de controle responsável pelo processamento das informações.

No contexto da solução proposta, o Model é responsável pelo gerenciamento dos dados associados aos postos de combustíveis, incluindo informações de localização, níveis de congestionamento, distância do usuário e demais atributos relevantes. Nesta camada também se encontra a implementação do algoritmo KNN, empregado para identificar os postos mais próximos e com menor índice de congestionamento. Assim, o Model concentra tanto as estruturas de dados quanto os métodos de cálculo e de acesso às APIs externas (como mapas e serviços de geolocalização).

A View corresponde à interface gráfica apresentada ao usuário, composta por a lista de recomendações de postos, indicadores de congestionamento e elementos de interação como botões e filtros. Essa camada não contém lógica de negócio, sendo responsável apenas por exibir os resultados fornecidos pelo Controller e por receber ações do usuário.

O Controller atua como intermediário entre a View e o Model. Ele interpreta as ações realizadas pelo usuário (por exemplo, solicitar a busca por postos próximos), aciona os métodos apropriados do Model — incluindo a execução do algoritmo KNN — e retorna à View os dados já processados para apresentação. Desse modo, o Controller coordena o fluxo da aplicação e garante que as regras de negócio sejam aplicadas corretamente.

A utilização do padrão MVC mostrou-se adequada para a proposta deste trabalho, pois possibilita uma estrutura organizada, modular e facilmente compreensível, ao mesmo tempo em que suporta a integração eficiente entre o cálculo de proximidade via KNN e a interface móvel voltada ao usuário final.

5

Apresentação e Análise de Resultados

Neste capítulo, são apresentados os resultados alcançados durante o desenvolvimento da aplicação móvel de apoio à mobilidade urbana. A análise dos dados coletados e o design das interfaces do aplicativo demonstram como cada componente foi estruturado para assegurar acessibilidade, usabilidade e eficiência na utilização do aplicativo “Mobilidade Inteligente”.

A aplicação utiliza o algoritmo K-Nearest Neighbors (KNN) para processar informações de localização e níveis de congestionamento, permitindo ao usuário identificar os postos de combustíveis mais próximos que oferecem menor impacto no tráfego. Os resultados obtidos evidenciam que a integração entre o modelo de recomendação e a interface gráfica proporciona uma experiência prática e intuitiva, favorecendo a tomada de decisão rápida durante os deslocamentos urbanos.

O design das telas foi concebido com foco na simplicidade e clareza, garantindo que motoristas possam acessar as recomendações de forma ágil, mesmo em situações de trânsito intenso. Além disso, a aplicação foi estruturada para oferecer feedback visual e interativo, reforçando a confiabilidade das sugestões geradas pelo algoritmo.

Assim, os resultados demonstram que o sistema cumpre seu objetivo principal: apoiar a mobilidade urbana por meio de uma solução tecnológica que combina análise de dados, inteligência computacional e design centrado no usuário.

5.1 Interpretação dos Dados Coletados

A coleta e análise de dados foram essenciais para compreender os padrões de deslocamento dos usuários e os desafios enfrentados na busca por postos de combustíveis em áreas urbanas congestionadas. A interpretação desses dados oferece um panorama sobre o comportamento dos motoristas, suas necessidades de abastecimento e as condições de tráfego que influenciam suas

escolhas.

Esse processo permitiu identificar variáveis relevantes — como localização geográfica, intensidade do congestionamento e distância até os postos — que serviram de base para o treinamento do algoritmo KNN. A análise revelou não apenas os pontos críticos de mobilidade, mas também sugestões para aprimorar o sistema, tornando as recomendações mais precisas e alinhadas às expectativas dos usuários.

Assim, a integração entre coleta de dados, análise e aplicação do algoritmo de recomendação fortalece a proposta da aplicação móvel, garantindo maior eficiência na mobilidade urbana e suporte prático para decisões rápidas durante os deslocamentos.

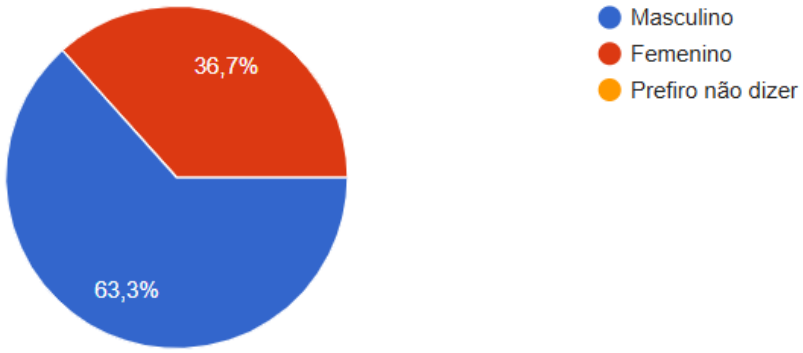
5.2 Perfil dos Participantes

Do núcleo definido para a pesquisa, participaram 36 (trinta e seis) pessoas. Na fase inicial, a aplicação foi testada por usuários comuns; porém, o público-alvo final corresponde a motoristas de diferentes modalidades de transporte, cuja locomoção depende diretamente do abastecimento com combustíveis. A amostra contemplou indivíduos com perfis variados, permitindo uma visão ampla sobre a percepção e uso do sistema.

No que diz respeito ao gênero dos participantes, 63,3% eram homens e 36,7% mulheres, conforme apresentado na Figura 18. Quanto à faixa etária, observou-se que 41,7% possuíam entre 18 e 25 anos, 41,7% entre 26 e 35 anos, 5,6% entre 36 e 45 anos, e 11,1% tinham 46 anos ou mais, como ilustrado na Figura 19. Esses dados demonstram predominância de um público jovem adulto, característico de usuários que possuem maior familiaridade com tecnologias digitais e soluções móveis.

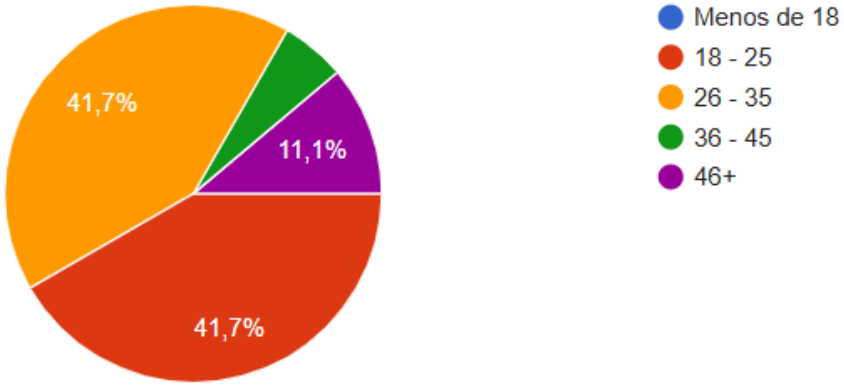
Em relação à frequência com que dirigem, verificou-se que 55,6% dos respondentes afirmaram não conduzir veículos com regularidade, enquanto 25% relataram dirigir diariamente e 19,4% dirigem esporadicamente, conforme expresso na Figura 20. Essa diversidade no perfil de utilização veicular contribui para uma avaliação mais realista do sistema, contemplando tanto motoristas frequentes quanto ocasionais.

Figura 18 – Gráfico Relacionado Sexo dos Participantes



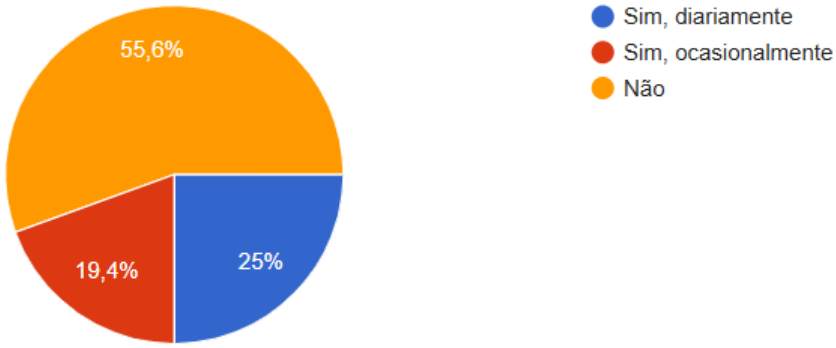
Fonte: Adaptado do GoogleForms, 2025.

Figura 19 – Gráfico Relacionado a Idade dos Participantes



Fonte: Adaptado do GoogleForms, 2025.

Figura 20 – Gráfico Relacionado a Frequência de Dirigir



Fonte: Adaptado do GoogleForms, 2025.

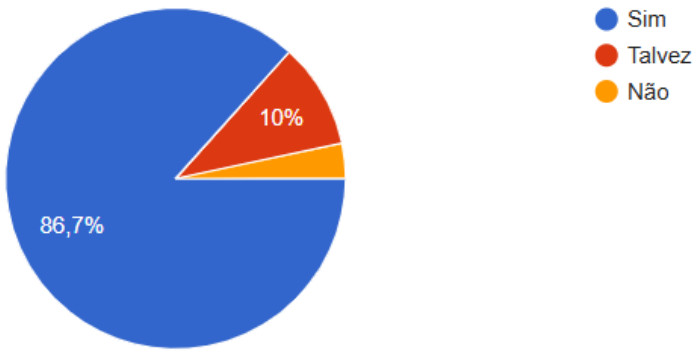
5.3 Utilidade e Interesse a Aplicação

Os resultados da pesquisa de intenção de uso da aplicação demonstram sua alta aceitação e relevância prática. Dos participantes, 86,7% afirmaram ter interesse em utilizar a aplicação, enquanto 10% se mostraram indecisos e apenas 3,3% declararam não ter interesse. Esse cenário evidencia que a aplicação atende a uma necessidade real e é percebida como útil pela ampla maioria dos potenciais usuários, ver Figura 21.

A elevada taxa de interesse sugere que a aplicação possui grande potencial de adoção e impacto positivo, reforçando sua pertinência como solução tecnológica. O grupo dos indecisos representa uma oportunidade estratégica: com ajustes na comunicação dos benefícios, melhorias na experiência do usuário ou oferta de testes práticos, há possibilidade concreta de ampliar ainda mais a taxa de aceitação. Já o percentual reduzido de rejeição confirma que a resistência ao uso é mínima e não compromete a viabilidade do projeto.

Assim, os dados da pesquisa sustentam que a aplicação não apenas é viável, mas também necessária para o público-alvo, validando sua utilidade e justificando o investimento em seu desenvolvimento e implementação.

Figura 21 – Gráfico Relacionado ao Interesse da Aplicação



Fonte: Adaptado do GoogleForms, 2025.

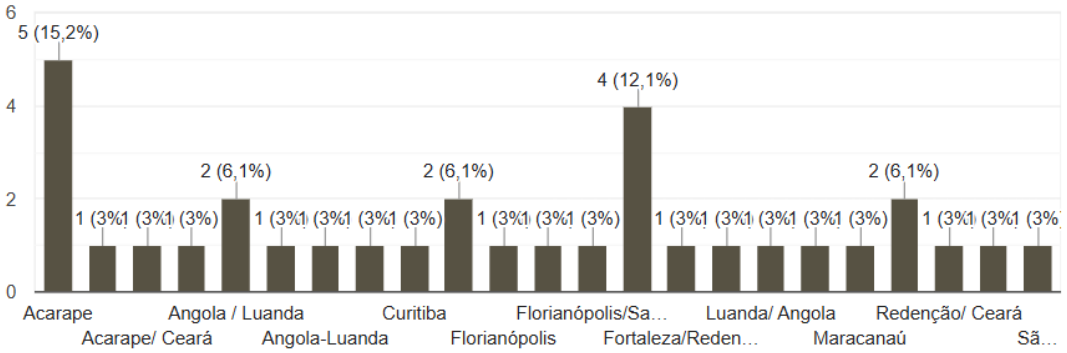
5.4 Funcionamento da Aplicação em Diferentes Locais

A Figura 22 apresenta os locais mais recorrentes indicados pelo sistema. Observa-se que o maior volume de recomendações ocorreu para o município de Acarape, representando 15,2% (5 registros) de todas as indicações retornadas. Esse comportamento pode estar associado ao fato de a localidade ser um dos pontos centrais dos testes, refletindo maior densidade de consultas realizadas próximos a esta região.

Em segundo nível de recorrência aparecem Fortaleza/Redenção, com 12,1% (4 registros), seguida por outros polos como Angola/Luanda, Curitiba e Redenção/Ceará, todos com percentual

de 6,1% (2 registros cada). Os demais retornos apresentam distribuição uniforme, cerca de 3% (1 registro por local), demonstrando que, embora haja concentração inicial em alguns municípios específicos, o algoritmo conseguiu realizar recomendações diversificadas, abrangendo múltiplas regiões.

Figura 22 – Gráfico Relacionado ao Funcionamento da Aplicação em Diferentes Locais

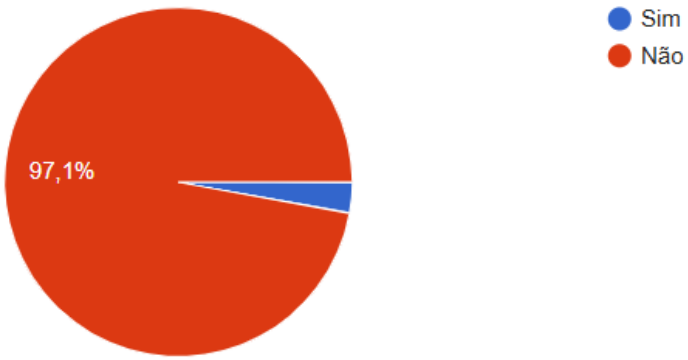


Fonte: Adaptado do GoogleForms, 2025.

Além do desempenho do algoritmo KNN na recomendação de postos de combustíveis, foi aplicado um questionário com o objetivo de analisar a percepção dos usuários em relação à usabilidade e funcionalidade da aplicação. A Figura 23 apresenta o resultado obtido, onde os participantes foram questionados se consideraram útil o uso do sistema para apoio à mobilidade urbana.

Os dados mostram uma predominância expressiva de avaliações negativas, sendo que 97,1% dos usuários responderam “Não”, indicando que, no estado atual, a aplicação ainda não atende plenamente às expectativas ou necessidades práticas do público. Apenas 2,9% afirmaram ter considerado o sistema útil, o que evidencia que sua aceitação ainda é baixa e reforça a necessidade de melhorias.

Figura 23 – Gráfico Relacionado ao Funcionamento da Aplicação em Diferentes Locais



Fonte: Adaptado do GoogleForms, 2025.

Esse resultado pode ser interpretado sob diferentes perspectivas. Primeiramente, a aplicação encontra-se em fase inicial de desenvolvimento, o que implica limitações funcionais previstas, especialmente considerando que a base de dados utilizada ainda é restrita e depende da disponibilidade de informações externas para geração das recomendações. Além disso, a baixa aceitação pode estar relacionada à experiência de uso, abrangência geográfica reduzida ou ausência de dados mais atualizados sobre rotas e congestionamentos, dificultando o funcionamento pleno desde o primeiro uso.

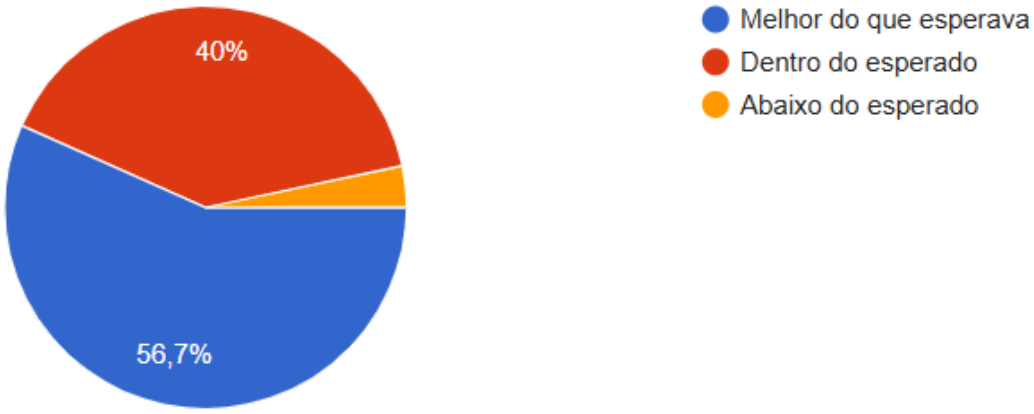
5.5 Funcionalidades Essenciais da Aplicação

A avaliação da funcionalidade da aplicação revelou resultados altamente positivos. Do total de participantes, 56,7% afirmaram que a aplicação atendeu mais do que o esperado, enquanto 40% consideraram que o desempenho esteve dentro do esperado e apenas 3,3% relataram desempenho abaixo do esperado. Esses números demonstram que a aplicação não apenas cumpre sua proposta inicial, mas em grande parte dos casos supera as expectativas dos usuários, evidenciando sua eficácia e valor agregado, constatar a Figura 24.

O elevado percentual de respostas acima do esperado indica que a aplicação possui recursos diferenciados e funcionalidades que surpreendem positivamente o público-alvo, reforçando sua relevância como solução tecnológica. O grupo que avaliou dentro do esperado confirma que a aplicação é consistente e confiável, atendendo às necessidades básicas previstas. Já o percentual reduzido de insatisfação (3,3%) mostra que eventuais ajustes são pontuais e não comprometem a percepção geral de qualidade.

Assim, os dados sustentam que a aplicação apresenta alto nível de funcionalidade e desempenho, validando sua proposta de inovação e justificando sua implementação. Além disso, o fato de superar expectativas em mais da metade dos casos reforça sua capacidade de gerar impacto positivo e diferencial competitivo no contexto em que será aplicada.

Figura 24 – Gráfico Relacionado a Funcionalidade da Aplicação

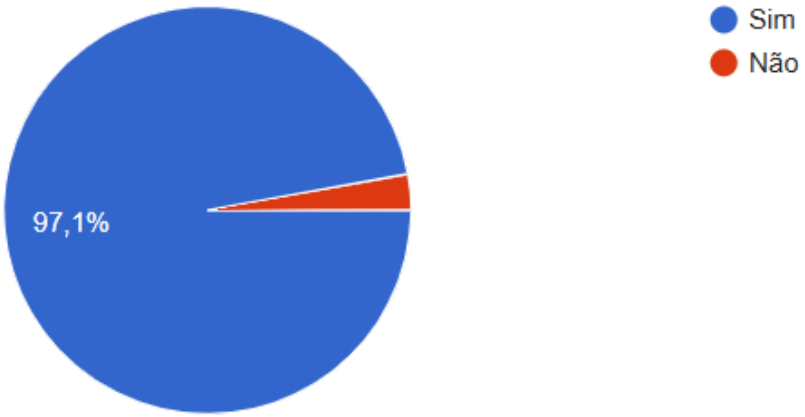


Fonte: Adaptado do GoogleForms, 2025.

5.6 Avaliação Quanto À Capacidade do Aplicativo em Localizar Postos de Combustíveis

Outro ponto analisado no questionário aplicado aos usuários foi a eficiência da aplicação em encontrar postos de combustíveis próximos com base nos dados fornecidos e no funcionamento do algoritmo de recomendação. A Figura 25 ilustra claramente que a aceitação foi majoritariamente positiva, apresentando 97,1% de respostas “Sim”, enquanto apenas 2,9% dos participantes relataram não ter conseguido localizar postos utilizando o sistema.

Figura 25 – Gráfico Relacionado a Funcionalidade da Aplicação



Fonte: Adaptado do GoogleForms, 2025.

Esse resultado representa um avanço significativo no contexto da proposta, evidenciando que a aplicação cumpriu com êxito uma de suas funções essenciais: identificar e retornar

postos dentro do raio configurado de consulta. Mesmo diante de limitações estruturais de dados, especialmente em regiões com menor registro no mapa, o sistema demonstrou capacidade de rastreamento e retorno de alternativas reais para abastecimento.

A baixa taxa de respostas negativas indica que eventuais falhas de localização ocorreram de maneira pontual e possivelmente associadas a áreas com menor densidade de postos cadastrados, problemas momentâneos de conectividade ou inconsistência temporária das APIs externas. Ainda assim, o desempenho geral do sistema pode ser considerado satisfatório nesta etapa experimental.

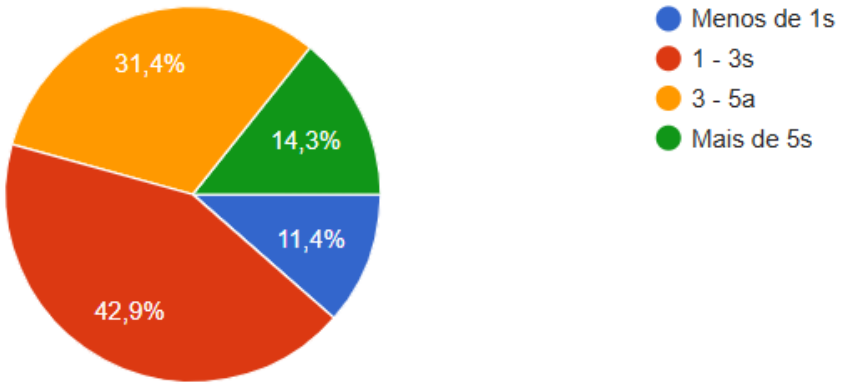
5.7 Tempo de Resposta para Localização e Recomendação dos Postos

Também foi avaliado o tempo necessário para que o sistema processasse a requisição e retornasse sugestões de postos próximos com menor índice de congestionamento. Conforme apresentado na Figura 26, a maioria dos usuários relatou um período de resposta entre 1 e 3 segundos, representando 42,9% das respostas. Esse intervalo pode ser considerado satisfatório dentro do contexto de aplicações móveis baseadas em geolocalização, demonstrando bom desempenho no processamento computacional e na comunicação com as APIs envolvidas.

Outro grupo significativo de respostas (31,4%) indicou um tempo entre 3 e 5 segundos, sugerindo que parte das requisições exigiu maior carregamento de dados, possivelmente devido à distância dos pontos consultados, variações na conexão do usuário ou maior volume de processamento requerido pelo algoritmo KNN. Mesmo assim, esses valores permanecem dentro de uma faixa aceitável para o uso prático.

Vale ressaltar que 14,3% dos usuários informaram tempo superior a 5 segundos, o que indica que o sistema pode enfrentar lentidão em alguns cenários específicos, o que pode estar associado à qualidade da rede móvel, número reduzido de postos na região ou limitações temporárias de consulta à base de dados externa. Já 11,4% reportaram resposta inferior a 1 segundo, um indicativo de que quando os dados são favoráveis e próximos, o aplicativo é capaz de responder de forma extremamente ágil.

Figura 26 – Gráfico Relacionado a Funcionalidade da Aplicação



Fonte: Adaptado do GoogleForms, 2025.

Os resultados demonstram que o aplicativo apresenta desempenho positivo e eficiente, com respostas majoritárias abaixo de 5 segundos, o que torna o uso viável e fluido na prática. Ainda assim, o tempo de espera registrado acima desse intervalo aponta para a possibilidade de futuras otimizações, como processamento mais leve das requisições, cache de resultados e ampliação do banco de dados local para reduzir consultas externas.

5.8 Desafios da Implementação da Aplicação na versão Beta com Base aos Feedback dos Usuários

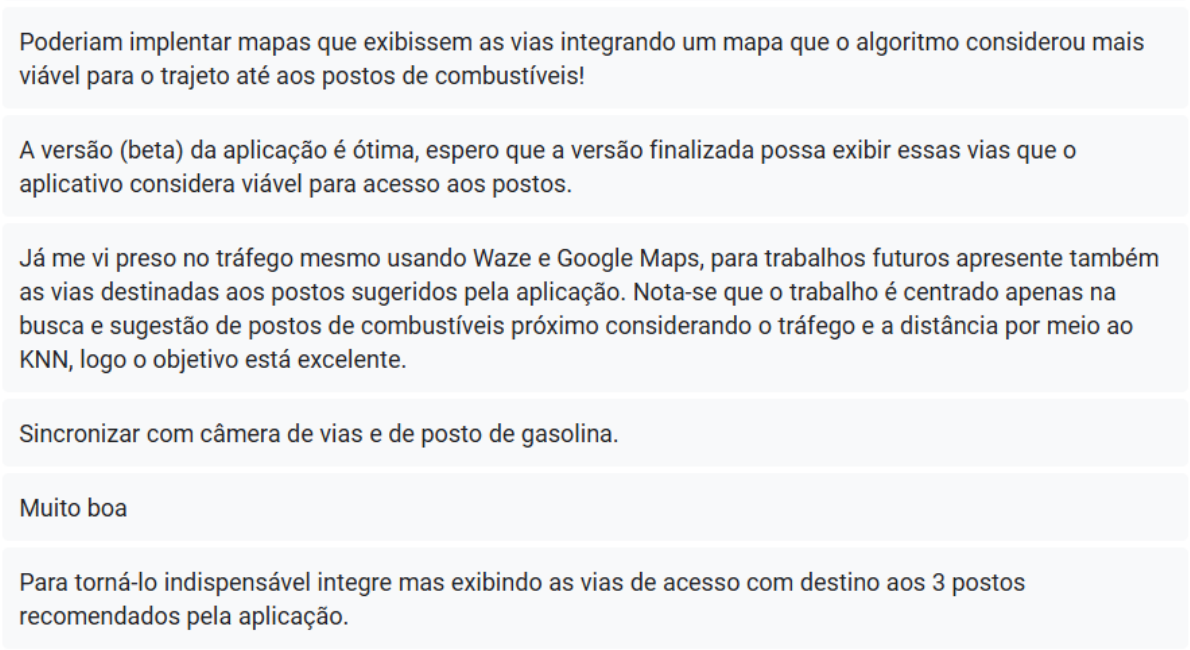
Durante os testes da versão beta da aplicação, foram coletados comentários dos usuários que evidenciam tanto a utilidade quanto os pontos de melhoria necessários. A seguir, sintetizam-se os principais desafios identificados:

- **Validação da utilidade e inovação:** garantir que o algoritmo KNN continue entregando resultados confiáveis em cenários de tráfego dinâmico, mantendo a relevância das recomendações.
- **Integração com rotas e mapas:** implementar recursos que permitam ao usuário visualizar rotas diretas até os postos recomendados, evitando a necessidade de consultas externas em outros aplicativos.
- **Visualização das vias e trajetos otimizados:** desenvolver uma interface gráfica que represente visualmente os cálculos do KNN, exibindo não apenas os postos sugeridos, mas também os caminhos considerados mais viáveis.
- **Sincronização com câmeras de trânsito e postos:** viabilizar a integração com sistemas externos de monitoramento, considerando aspectos técnicos, legais e de infraestrutura.

- **Exibição das vias de acesso aos postos recomendados:** ampliar a experiência do usuário ao mostrar as vias de acesso aos três postos sugeridos, aumentando a eficiência e tornando o aplicativo indispensável no apoio à mobilidade urbana.

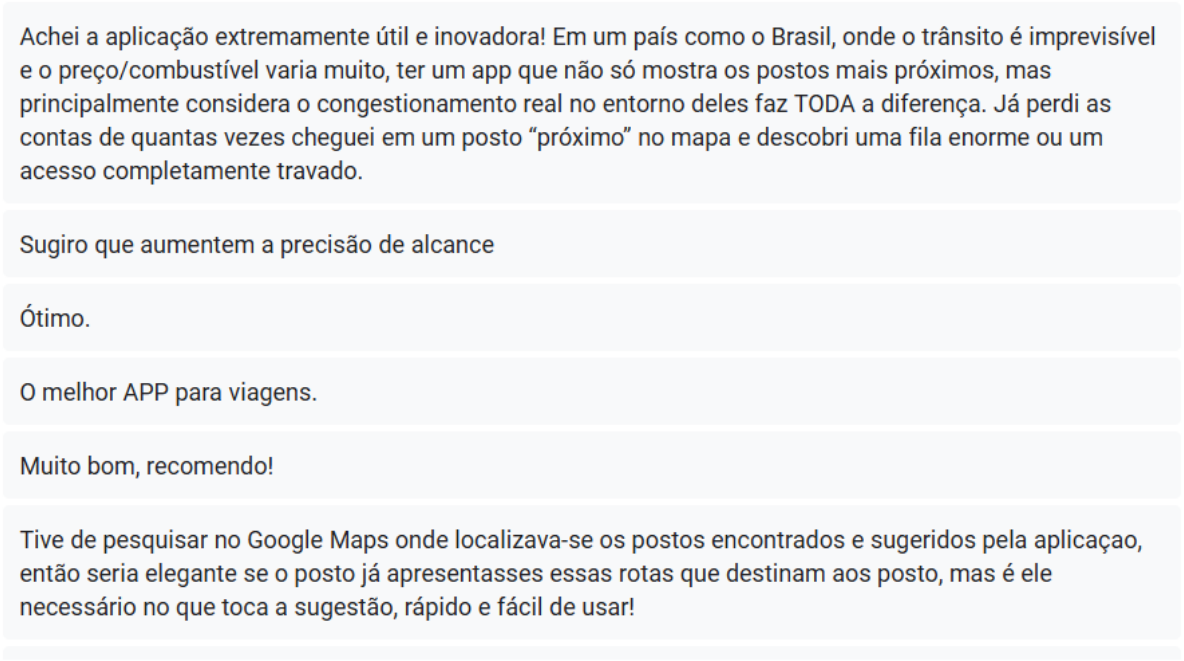
Esses desafios representam tanto limitações atuais quanto oportunidades de evolução futura, podendo ser explorados em trabalhos posteriores para ampliar a robustez e a aplicabilidade da solução. Segue abaixo as Figuras 27 e 28, mostrando os comentário realizados após o uso da versão beta da aplicação.

Figura 27 – Comentário e Sugestões dos Usuários



Fonte: Adaptada do GoogleForms, 2025.

Figura 28 – Comentário e Sugestões dos Usuários



Fonte: Adaptado do GoogleForms, 2025.

5.8.1 Síntese dos Desafios

Os comentários dos usuários da versão beta evidenciam que a aplicação já é percebida como útil e inovadora, mas também apontam desafios técnicos importantes para sua evolução. Entre eles destacam-se: a necessidade de integração com mapas e rotas, a visualização gráfica dos trajetos otimizados, o acesso a dados externos como câmeras de trânsito e postos, além da manutenção da precisão do algoritmo KNN em cenários dinâmicos de mobilidade urbana.

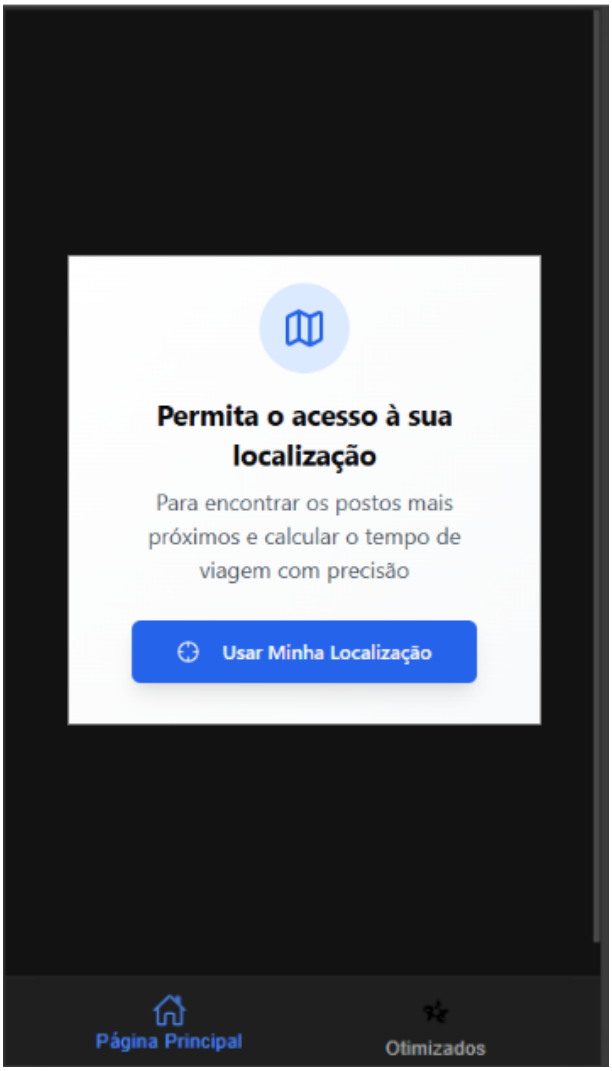
Esses pontos representam tanto limitações atuais quanto oportunidades de evolução futura, podendo ser explorados em trabalhos posteriores para ampliar a robustez, a eficiência e a aplicabilidade da solução proposta.

5.9 Tela Principal da Aplicação

A aplicação móvel de apoio à mobilidade urbana foi projetada para oferecer uma experiência de uso prática e acessível aos usuários que buscam otimizar seus deslocamentos. Diferente de sistemas que exigem registro ou autenticação, o aplicativo solicita apenas a permissão de acesso à localização do usuário, garantindo simplicidade no uso e maior agilidade no processo de consulta. Como ilustrado na Figura 29, a interface apresenta um design limpo e intuitivo, permitindo que o usuário, ao conceder acesso à sua localização, receba sugestões dos postos de

combustíveis mais próximos com menor nível de congestionamento. Essa abordagem elimina barreiras de entrada, tornando o sistema mais inclusivo e acessível, ao mesmo tempo em que assegura a proteção das informações pessoais, já que não há necessidade de armazenamento de dados sensíveis.

Figura 29 – Tela Inicial da Aplicação



Fonte: Elaborada pelo Autor, 2025.

A integração do algoritmo KNN (K-Nearest Neighbors) possibilita que o aplicativo ofereça recomendações precisas e contextualizadas, considerando tanto a proximidade geográfica quanto as condições de tráfego. Dessa forma, o sistema garante usabilidade, eficiência e apoio direto à mobilidade urbana, atendendo às necessidades dos usuários de forma segura e prática.

5.10 Tela tela de Busca da Aplicação

A tela de busca da aplicação móvel de apoio à mobilidade urbana foi desenvolvida para oferecer uma experiência simples e objetiva ao usuário. Como ilustrado na Figura 30, a interface apresenta um campo central de pesquisa que, ao ser acionado, utiliza a permissão de localização previamente concedida pelo usuário para identificar sua posição atual. A partir desse ponto, o sistema aplica o algoritmo KNN (K-Nearest Neighbors) para sugerir os postos de combustíveis mais próximos, considerando não apenas a distância geográfica, mas também o nível de congestionamento das vias de acesso.

O design da tela privilegia a clareza visual e a usabilidade, com informação de "carregando estações"e dois botões de ação destacados, permitindo que o usuário voltar a tela dos resultados e outro para visualizar a tela de Processamento de dados, exibindo a IA processando os dados, essa interação simplificada elimina etapas burocráticas, tornando o processo de busca ágil e acessível a diferentes perfis de motoristas.

Figura 30 – Tela tela de Busca daAplicação

Fonte: Elaborada pelo Autor, 2025.

5.11 Tela de Processamento de Dados da Aplicação

A tela de processamento da aplicação móvel de apoio à mobilidade urbana foi projetada para atuar como o núcleo funcional do sistema, responsável por receber os dados de localização do usuário e otimizar a busca pelos postos de combustíveis. Como ilustrado na Figura 31, ao conceder a permissão de acesso à localização, o aplicativo coleta as informações geográficas e de tráfego em tempo real, encaminhando-as para o módulo de cálculo baseado no algoritmo KNN (K-Nearest Neighbors).

O algoritmo, previamente treinado com k = 3 vizinhos, realiza a análise comparativa entre a posição atual do usuário e os pontos de interesse cadastrados (postos de combustíveis), considerando tanto a proximidade espacial quanto o nível de congestionamento das vias de acesso. A tela, portanto, não apenas recebe os dados, mas também orquestra o envio das informações para o KNN, que retorna como resultado os três postos mais adequados ao contexto do deslocamento.

Visualmente, a interface apresenta uma estrutura clara e objetiva, exibindo os resultados em formato de lista ou mapa interativo, permitindo ao usuário identificar rapidamente as opções sugeridas. Essa abordagem garante eficiência na tomada de decisão, reduzindo o tempo de busca e promovendo maior fluidez na mobilidade urbana.

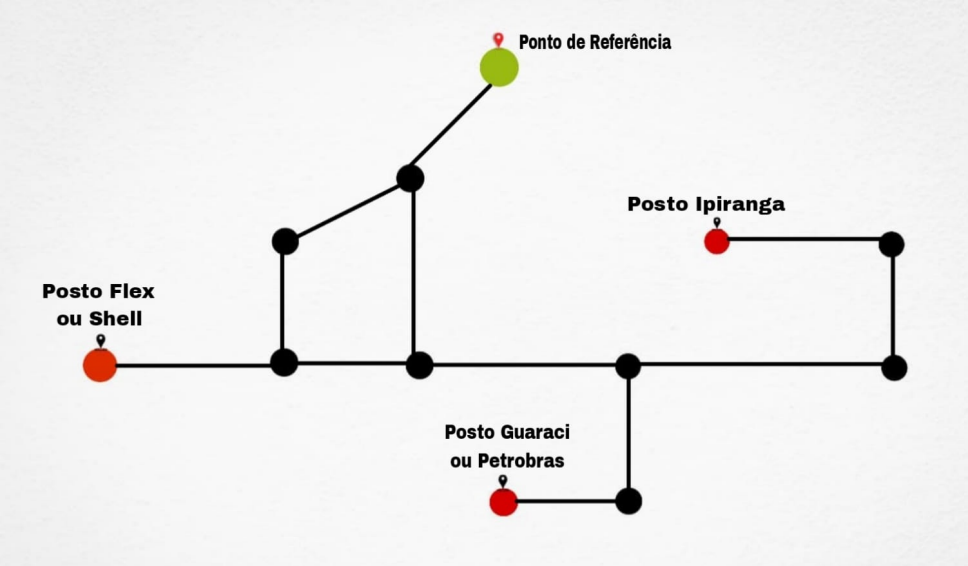
Figura 31 – Tela de Processamento de Dados da Aplicação

Fonte: Elaborada pelo Autor, 2025.

5.12 Representação da Localização dos Postos de Combustíveis Sugeridos no Mapa em Grafos

Diante da inexistência de registros de postos de combustíveis na região de Acarape/Redenção na API do OpenStreetMap, a aplicação não pôde identificar estabelecimentos próximos com base na localização real do usuário. Para possibilitar a avaliação do correto funcionamento da aplicação, adotou-se uma busca simulada utilizando como referência as coordenadas geográficas da cidade de Fortaleza (Latitude: -3.71839, Longitude: -38.5434). Essa abordagem permitiu testar os módulos responsáveis pela identificação dos postos mais próximos e pela análise do nível de congestionamento nas vias adjacentes. A Figura 32 apresenta a disposição espacial dos postos de combustíveis retornados pela aplicação a partir dessa simulação.

Figura 32 – Localizações dos Postos de Combustíveis Sugeridos pela Aplicação.



Fonte: Elaborada pelo Autor, 2025.

5.13 Tela dos Resultados e Sugestão do Postos

A tela de exibição dos resultados constitui a etapa final da interação entre o usuário e a aplicação móvel de apoio à mobilidade urbana. Como ilustrado nas Figuras 33 e 34, após o processamento realizado pelo algoritmo KNN (K-Nearest Neighbors), treinado com k = 3 vizinhos, o sistema apresenta ao usuário os três postos de combustíveis recomendados de acordo com sua localização atual e o nível de congestionamento das vias de acesso.

A interface foi projetada para ser clara, objetiva e funcional, exibindo os postos em formato de lista ou mapa interativo. Cada recomendação é acompanhada de informações relevantes, como distância aproximada, tempo estimado de deslocamento e nível de tráfego, permitindo que o

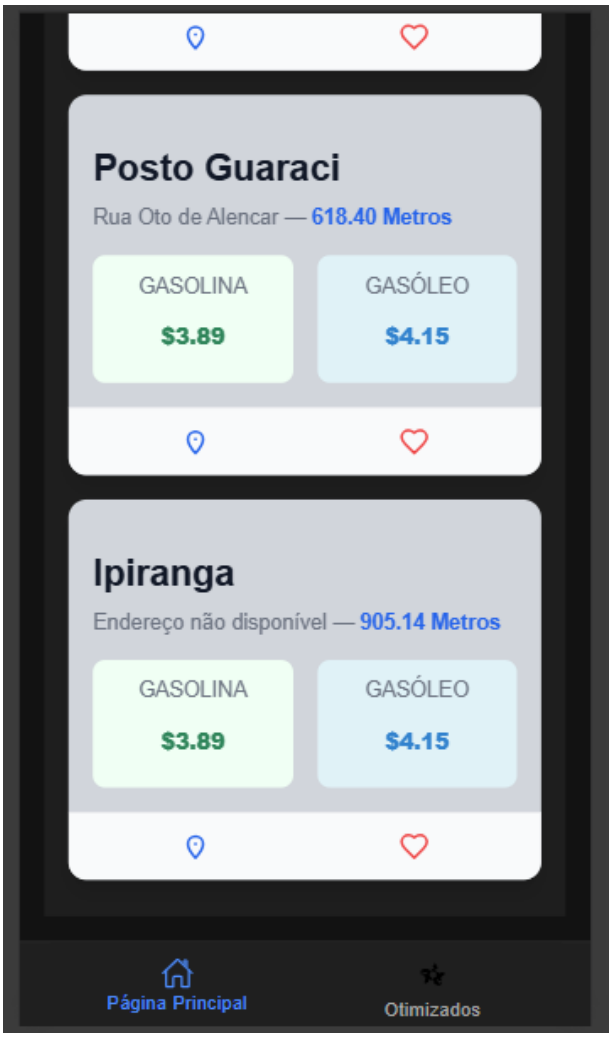
usuário compare rapidamente as opções disponíveis. O uso de ícones e cores diferenciadas facilita a interpretação visual, destacando o posto mais próximo ou aquele com menor congestionamento.

Essa tela desempenha papel essencial na usabilidade e eficiência do sistema, pois traduz os cálculos complexos do algoritmo em informações acessíveis e práticas para o usuário. Além disso, estabelece um fluxo de interação transparente: o usuário fornece sua localização, o sistema processa os dados e retorna recomendações otimizadas.

Figura 33 – Tela dos Resultados e Sugestão do Postos

Fonte: Elaborada pelo Autor, 2025.

Figura 34 – Tela dos Resultados e Sugestão do Postos



Fonte: Elaborada pelo Autor, 2025.

A aplicação móvel de apoio à mobilidade urbana apresenta um conjunto de telas projetadas para garantir simplicidade, eficiência e usabilidade. A tela inicial solicita apenas a permissão de localização, eliminando a necessidade de cadastro e tornando o acesso rápido e acessível. Em seguida, a tela de busca permite que o usuário acione o sistema para localizar postos de combustíveis próximos, utilizando dados de tráfego em tempo real. A tela de processamento recebe essas informações e as encaminha ao algoritmo KNN, treinado com três vizinhos, que realiza o cálculo e otimiza a seleção dos postos mais adequados. Logo, a tela de exibição dos resultados apresenta ao usuário os três postos recomendados, em formato de lista ou mapa interativo, acompanhados de informações como distância em metros, endereço se disponível, os preços de dois tipo de combustível tudo para facilitar a tomada de decisão.

6

Conclusões e Trabalhos Futuros

Conclusão

As atividades deste trabalho iniciaram-se com uma pesquisa bibliográfica voltada à identificação dos principais conceitos relacionados à mobilidade urbana, sistemas inteligentes de transporte e aplicações móveis. Foram consultadas bases como Google Acadêmico e SciELO, priorizando estudos recentes e alinhados ao uso de tecnologias de localização e algoritmos de recomendação.

Esse levantamento permitiu compreender como dados em tempo real podem contribuir para a otimização do deslocamento urbano, reduzindo o tempo de procura por postos de combustíveis e evitando rotas congestionadas. A aplicação desenvolvida torna-se relevante ao oferecer suporte direto ao motorista, auxiliando na tomada de decisão durante o trânsito e possibilitando maior fluidez no deslocamento. Dessa forma, além de propor uma solução tecnológica, este trabalho contribui para a melhoria na experiência do usuário no ambiente urbano, potencializando a eficiência no uso das vias e favorecendo uma mobilidade mais inteligente e acessível à população.

Com base nesse referencial teórico, avançou-se para a etapa prática, envolvendo a definição dos requisitos funcionais e não funcionais do aplicativo, a coleta e integração de dados via GPS e APIs de mapas, e a implementação da aplicação utilizando Angular, HTML, CSS, JavaScript e Python. O algoritmo KNN foi incorporado para sugerir postos de combustível próximos considerando menor congestionamento. A solução foi testada em dispositivos móveis e validada por meio de testes funcionais e questionários de usabilidade, verificando adequação, precisão e facilidade de uso.

Metodologia Utilizada

A metodologia adotada combinou uma pesquisa teórica (revisão bibliográfica) para construção do embasamento conceitual e identificação das tecnologias adequadas, com uma pesquisa prática de natureza aplicada, voltada ao desenvolvimento, integração de APIs e testes do aplicativo móvel. A fase teórica guiou a definição dos requisitos do sistema, enquanto a fase experimental permitiu implementar a solução, testar o algoritmo KNN em cenários simulados e validar a usabilidade por meio de avaliações com usuários.

Limitações do Trabalho

Apesar dos resultados positivos, algumas limitações podem comprometer a credibilidade ou generalização do trabalho. A principal limitação refere-se à dependência de APIs externas, como Google Maps e Here Maps, cujos dados podem sofrer restrições, custos ou variações de disponibilidade. Além disso, os testes foram realizados com um número reduzido de usuários e em ambientes controlados, o que limita a avaliação da performance do algoritmo KNN em situações reais de tráfego urbano complexo. Também não foram exploradas diferenças regionais de infraestrutura, que podem influenciar a precisão da recomendação.

Trabalhos Futuros

Como trabalhos futuros, recomenda-se ampliar a base de testes com um número maior e mais diverso de usuários, integrar novas fontes de dados de tráfego em tempo real e explorar algoritmos alternativos, como redes neurais ou métodos híbridos de recomendação e para exibição das vias encontradas do algoritmo no map integrar também o Google Maps API. Também é pertinente incorporar avaliações contínuas de desempenho, adicionar funcionalidades como histórico de deslocamentos e otimização de rotas, além de implementar versões mais leves do aplicativo para uso eficiente em regiões com baixa conectividade. Tais melhorias podem fortalecer a robustez da solução e ampliar seu impacto na mobilidade urbana.

Referências

ABIB, D. *Enterprise Solution Architect, FSI – LATAM*. <https://aws.amazon.com/pt/blogs/aws-brasil/modernize-seu-servidor-soap-legado-usando-o-amazon-api-gateway-e-o-aws-lambda/>, 2022. Acesso em: 13 de nov. 2025. Disponível em: <https://aws.amazon.com/pt/>. Citado na página 40.

ALMEIDA, R. et al. Quando a tecnologia apoia a mobilidade urbana: Uma avaliação sobre a experiência do usuário com aplicações móveis. In: *Proceedings of the XV Brazilian Symposium on Human Factors in Computer Systems (IHC 2016). Brazilian Society of Computation-SBC, Porto Alegre, Brazil*. [S.l.: s.n.], 2016. Acesso em: 25 set. 2025. Citado na página 30.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, Taylor & Francis, v. 46, n. 3, p. 175–185, 1992. Acesso em: 25 set. 2025. Citado na página 42.

ALVES, L. H. et al. Análise documental e sua contribuição no desenvolvimento da pesquisa científica. *Cadernos da FUCAMP*, v. 20, n. 43, 2021. Acesso em: 14 de nov. 2025. Citado na página 67.

ALVIM, A. T. B.; IZAGA, F. G. d.; CLAPS, R. F. Mobilidade urbana em perspectiva: novos olhares sobre as dinâmicas da cidade contemporânea. *Cadernos Metrópole*, SciELO Brasil, v. 26, n. 60, p. 413–421, 2024. Acesso em: 12 de nov. 2025. Citado na página 30.

ANDRADE, J. N.; GALVÃ, D. C. et al. O conceito de smart cities aliado à mobilidade urbana. *Revista Hum@ nae*, v. 10, n. 1, 2016. Acesso em: 12 de nov. 2025. Citado na página 30.

APPLEINC. *ios overview*. <http://developer.apple.com/library/ios/referencelibrary/GettingStarted/URLiPhoneOSOverview/index.html//appleref/doc/uid/TP40007592>, 2011. Acesso em: 12 nov. 2025. Disponível em: <https://developer.apple.com/>. Citado na página 36.

BACKES, S. et al. Hubbard band versus oxygen vacancy states in the correlated electron metal srvo 3. *Physical Review B*, APS, v. 94, n. 24, p. 241110, 2016. Acesso em: 25 set. 2025. Citado na página 24.

BAPTISTA, C. et al. Logística e mobilidade: um modelo que promova a mobilidade de veículos de grande porte em vias urbanas e rodoviárias. São Paulo, 2024. Acesso em: 17 de out. 2025. Citado na página 16.

BATTY, M. *Inventing future cities*. [S.l.]: MIT press, 2018. Citado na página 21.

BENNETT, J. *OpenStreetMap*. [S.l.]: Packt Publishing Ltd, 2010. Acesso em: 12 de nov. 2025. Citado na página 32.

BOLFE, É. L.; MATIAS, L. F.; FERREIRA, M. C. Sistemas de informação geográfica: uma abordagem contextualizada na história. *Geografia*, v. 33, n. 1, p. 69–88, 2008. Acesso em: 12 de nov. 2025. Citado na página 31.

BRASIL, I. Censo demográfico 2022. *Dados nacionais. Fundaçao Instituto Brasileiro de Geografia e Estatıstica. Brasil*, 2023. Acesso em: 13 de nov. 2025. Citado 2 vezes nas páginas 27 e 28.

BRENNAND, C. A. et al. An intelligent transportation system for detection and control of congested roads in urban centers. In: IEEE. *2015 IEEE Symposium on Computers and Communication (ISCC)*. [S.l.], 2015. p. 663–668. Acesso em: 25 set. 2025. Citado 2 vezes nas páginas 46 e 47.

CABECUDO, A.; KAPLAN-MOSS, J. Django. , 2010. Acesso em: 13 de nov. 2025. Citado na página 51.

CABRAL, J. A.; NUNES, R. S. A inteligência artificial no departamento de recursos humanos: um estudo de caso sobre a ia no processo de recrutamento e seleção. 269, 2021. Acesso em: 25 set. 2025. Citado na página 33.

CÂMARA, G.; CASANOVA, M. A.; MAGALHÃES, G. C. Anatomia de sistemas de informação geográfica. 1996. Acesso em: 12 de nov. 2025. Citado na página 31.

CARTER, M.; GROVER, V. Me, my self, and i (t). *MIS quarterly*, JSTOR, v. 39, n. 4, p. 931–958, 2015. Acesso em: 25 set. 2025. Citado na página 29.

CARVALHO, C. H. R. d. et al. A mobilidade urbana no brasil. Instituto de Pesquisa Econômica Aplicada (Ipea), 2011. Acesso em: 22 ago. 2025. Citado 2 vezes nas páginas 16 e 20.

CASTRO, M. F.; TEDESCO, P. Aplicaçao de conceitos de wayfinding em interfaces mobile de recomendaçao de rota. In: SBC. *Simpósio Brasileiro de Sistemas de Informação (SBSI)*. [S.l.], 2014. p. 470–481. Acesso em: 25 set. 2025. Citado na página 30.

CEBDS. *Clima, Enegia e Finanças Sustentáveis. O Desefio da Mobilidade Sustentável. Disponível no*. <https://cebds.org/noticia/o-desafio-da-mobilidade-sustentavel/?gad_source=1&gad_campaignid=1717588074&gbraid=0AAAADiVo8BtigG3X8GQKqGHDBmhIE5Vb&gclid=Cj0KCQjwqqDFBhDhARIsAIHTlkusuSQ8xFVRZHM4qoeKpZ1-wQQ47BOm-F2qRASRdRf6Gskl2bkv4NgaAjkeEALw_wcB>, 2016. Acesso em: 25 set. 2025. Disponível em: <https://cebds.org/>. Citado na página 25.

CHAVES, A. P.; STEINMACHER, I.; VIEIRA, V. Social networks and collective intelligence applied to public transportation systems: A survey. *Viii Sbsc*, p. 16–23, 2011. Acesso em: 25 set. 2025. Citado na página 29.

CHEN, C. et al. Tripplanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 16, n. 3, p. 1259–1273, 2014. Acesso em: 25 set. 2025. Citado na página 29.

CHILORA, J. M. Proposta de um sistema inteligente de controle de tráfego urbano na cidade do huambo. Caala Instituto Superior Politecnico, 2024. Acesso em: 12 de nov. 2025. Citado na página 33.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Acesso em: 25 set. 2025. Citado 3 vezes nas páginas 41, 42 e 64.

DARGAY, J.; GATELY, D.; SOMMER, M. Vehicle ownership and income growth, worldwide: 1960-2030. *The energy journal*, SAGE Publications Sage CA: Los Angeles, CA, v. 28, n. 4, p. 143–170, 2007. Acesso em: 22 ago. 2025. Citado na página 18.

DOOLAN, R.; MUNTEAN, G.-M. Ecotrec—a novel vanet-based approach to reducing vehicle emissions. *IEEE transactions on intelligent transportation systems*, IEEE, v. 18, n. 3, p. 608–620, 2016. Acesso em: 25 set. 2025. Citado na página 45.

DOURADO, D. A. F.; CAMPOS, V. B. G. Sistemas de informação em tempo real no gerenciamento da demanda de tráfego urbano. *Rio de Janeiro*, 2007. Acesso em: 12 de nov. 2025. Citado na página 46.

EFAGUNDES, E. M. *OMS: Poluição do ar mata cerca de 7 milhões de pessoas por ano. 2014. Disponível no*. <https://news.un.org/pt/story/2014/03/1469391>, 2014. Acesso em: 22 ago. 2025. Disponível em: <https://news.un.org/pt/>. Citado na página 18.

EFAGUNDES, E. M. *Como a Inteligência Artificial pode ajudar na mobilidade humana. Efagundes. 2019. Disponível no*. <https://efagundes.com/blog/como-a-inteligencia-artificial-pode-ajudar-na-mobilidade-humana/>, 2019. Acesso em: 22 ago. 2025. Disponível em: <https://efagundes.com/>. Citado na página 34.

EFAGUNDES, E. M. *O que é inteligência artificial? Como funciona, exemplos e aplicações. TOTVS. 2019. Disponível no*. <https://www.totvs.com/blog/inovacoes/o-que-e-inteligencia-artificial/>, 2019. Acesso em: 22 ago. 2025. Disponível em: <https://www.totvs.com/blog/>. Citado na página 33.

EFAGUNDES, E. M. *As Causas e Soluções para os Engarrafamentos no Brasil. 2019. Disponível no*. <https://mobilidade.estadao.com.br/mobilidade-para-que/as-causas-e-solucoes-para-os-engarrafamentos-no-brasil/>, 2020. Acesso em: 22 ago. 2025. Disponível em: <https://mobilidade.estadao.com.br/>. Citado na página 34.

ESEME, S. *GraphQL vs REST: Tudo o Que você Precisa Saber*. <https://kinsta.com/pt/blog/graphql-vs-rest/>, 2025. Acesso em: 13 de nov. 2025. Disponível em: <https://kinsta.com/pt/blog/author/solomoneseme/>. Citado na página 41.

FANG, K. Smart mobility": Is it the time to re-think urban mobility. *The World Bank Group. Available at: blogs. worldbank. org/transport*, 2015. Acesso em: 22 ago. 2025. Citado na página 29.

FANINI, V. Mobilidade urbana. série de cadernos técnicos. *Publicações temáticas da Agenda Parlamentar do Conselho Regional de Engenharia, Arquitetura e Agronomia do Paraná–CREA-PR*, 2011. Acesso em: 22 ago. 2025. Citado na página 24.

FERREIRA, M. N. F. et al. Ensinando design de interface de usuário de aplicativos móveis no ensino fundamental. *Revista Brasileira de Informática na Educação*, v. 28, p. 48–72, 2020. Acesso em: 17 de out. 2025. Citado na página 22.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine, 2000. Acesso em: 13 de nov. 2025. Citado na página 39.

FIGUEIREDO, C. M.; NAKAMURA, E. Computação móvel: Novas oportunidades e novos desafios. *T&C Amazônia*, ano, v. 1, n. 2, p. 21, 2003. Acesso em: 22 ago. 2025. Citado na página 17.

FLANAGAN, D.; MATILAINEN, P. *JavaScript*. [S.l.]: Anaya Multimedia, 2007. Acesso em: 14 de nov. 2025. Citado na página 54.

FLORIDI, L. *Information: A very short introduction*. [S.l.]: Oxford University Press, 2010. v. 225. Acesso em: 22 ago. 2025. Citado na página 30.

FRANÇOZO, M. T.; MELLO, N. C. D. Influência dos aplicativos de smartphones para transporte urbano no transito. In: *7th Luso-Brazilian Congress for Urban, Regional, Integrated and Sustainable Planning*. [S.l.: s.n.], 2016. Acesso em: 22 ago. 2025. Citado na página 29.

FRIEDMAN, J. The elements of statistical learning: Data mining, inference, and prediction. *(No Title)*, 2009. Acesso em: 22 ago. 2025. Citado na página 42.

GIL, A. C. *Métodos e técnicas de pesquisa social*. [S.l.]: 6. ed. Ediitora Atlas SA, 2008. Acesso em: 22 ago. 2025. Citado na página 64.

GOMES, D. A.; JR, P. J. Web services soap e rest. *Revista Intellectus*, v. 6, n. 1, p. 88–114, 2009. Acesso em: 13 de nov. 2025. Citado na página 39.

GOOGLEINC. *What is android*. <http://developer.android.com/guide/basics/what-isandroid.html>, 2011. Acesso em: 12 nov. 2025. Disponível em: <https://developer.android.com/>. Citado na página 35.

GOWRISHANKAR, S.; STERN, E.; WORK, D. B. Including the social component in smart transportation systems. In: *National Workshop on Transportation Cyber-Physical Systems*. [S.l.: s.n.], 2014. Acesso em: 22 ago. 2025. Citado na página 29.

GROVER, S.; PEA, R. Computational thinking in k–12: A review of the state of the field. *Educational researcher*, Sage Publications Sage CA: Los Angeles, CA, v. 42, n. 1, p. 38–43, 2013. Acesso em: 22 ago. 2025. Citado na página 21.

GUIMARÃES, J. P. C. Sigtran: um sistema integrado de gestão de transporte acadêmico para monitoramento em tempo real. 2025. Acesso em: 12 de nov. 2025. Citado na página 33.

HANSON, S. Gender and mobility: new approaches for informing sustainability. *Gender, Place & Culture*, Taylor & Francis, v. 17, n. 1, p. 5–23, 2010. Acesso em: 22 ago. 2025. Citado na página 24.

JENKINS, H. Cultura da convergência: a colisão entre os velhos e novos meios de comunicação; trad. *Susana Alexandria, 2a ed. São Paulo: Aleph*, 2009. Acesso em: 22 ago. 2025. Citado na página 17.

JR, G. D. Simulação de controle adaptativo de tráfego urbano através de sistema multiagentes e com base em dados reais. 2015. Acesso em: 25 set. 2025. Citado na página 34.

JUNIOR, E. F. G. Application programming interface. 2024. Acesso em: 13 de nov. 2025. Citado na página 38.

JUNIOR, J. B. B. O aplicativo kahoot na educação: verificando os conhecimentos dos alunos em tempo real. In: *Livro de atas X Conferência Internacional de TIC na Educação–Clallenges*. [S.l.: s.n.], 2017. p. 1587–1602. Acesso em: 17 de out. 2025. Citado na página 21.

KOTSIANTIS, S. B. et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, Amsterdam, v. 160, n. 1, p. 3–24, 2007. Acesso em: 22 ago. 2025. Citado na página 42.

KUNIEDA, M.; GAUTHIER, A. *Gender and urban transport: smart and affordable: module 7a*. [S.l.]: Deutsche Gesellschaft für Technische Zusammenarbeit (GTZ), 2007. Citado na página 29.

LEADMINE. *Application Programming Interface (API). Disponível no*. <https://www.leadmine.net/glossary/application-programming-interface/>, 2021. Acesso em: 13 de nov. 2025. Disponível em: <https://www.leadmine.net/glossary/>. Citado na página 38.

LEFUNDES, C. *Entenda Sobre API de Uma Vez Por todas. Disponível no*. <https://medium.com/@crystian.lf/entenda-sobre-api-de-uma-vez-por-todas-fb5475df8db0>, 2024. Acesso em: 13 de nov. 2025. Disponível em: <https://medium.com/@crystian.lf>. Citado na página 39.

LONDON, S. J.; ROMIEU, I. Health costs due to outdoor air pollution by traffic. *The Lancet*, Elsevier, v. 356, n. 9232, p. 782–783, 2000. Acesso em: 22 ago. 2025. Citado na página 21.

LOUREIRO, C.; GUEDES, E. B.; FIGUEIREDO, C. M. Monitoramento inteligente de tempo real para tráfego urbano em cidades brasileiras. In: SBC. *Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*. [S.l.], 2025. p. 41–50. Acesso em: 12 de nov. 2025. Citado na página 46.

LUCCHESI, C. *Domine HTML 5 e CSS 3: Do Básico ao Avançado*. [S.l.]: Claudio Lucchesi, 2024. Acesso em: 13 de nov. 2025. Citado na página 49.

LUTZ, M. *Programming python*. [S.l.]: "O'Reilly Media, Inc.", 2001. Acesso em: 14 de nov. 2025. Citado 2 vezes nas páginas 52 e 53.

MACHADO, B. C.; SILVA, S. da. Influências da inteligência artificial (ia) na otimização do tráfego urbano: um referencial teórico. In: EDITORA CIENTÍFICA DIGITAL. *TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO: PESQUISAS EM INOVAÇÕES TECNOLÓGICAS-VOLUME 2*. [S.l.], 2022. v. 2, p. 161–172. Acesso em: 25 set. 2025. Citado na página 33.

MACHADO, L.; PICCININI, L. S. Os desafios para a efetividade da implementação dos planos de mobilidade urbana: uma revisão sistemática. *Urbe. Revista Brasileira de Gestão Urbana*, SciELO Brasil, v. 10, n. 1, p. 72–94, 2018. Acesso em: 25 set. 2025. Citado na página 28.

MENESES, H. B.; LEANDRO, C. H. P.; LOUREIRO, C. F. G. Indicadores de desempenho para sistemas centralizados de controle do tráfego urbano em tempo real. In: *Anais do XVII Congresso de Pesquisa e Ensino em Transportes*. [S.l.: s.n.], 2003. Acesso em: 12 de nov. 2025. Citado na página 45.

OLIVEIRA, J. C. de; NETO, W. P. de S.; SANTOS, A. d. P. dos. Aplicando api do google maps para criar mapa interativo. estudo de caso: Campus-viçosa. In: *XIV Simposio Internacional SELPER*. [S.l.: s.n.], 2010. Acesso em: 12 de nov. 2025. Citado na página 31.

OLIVEIRA, M. *Blog Ponto de vista: O Guia Definitivo para Frameworks de Desenvolvimento de API. Disponível no*. <https://apidog.com/pt/blog/api-development-frameworks-pt/>, 2025. Acesso em: 13 de nov. 2025. Disponível em: <https://apidog.com/>. Citado 3 vezes nas páginas 57, 58 e 59.

PAN, J. et al. Proactive vehicle re-routing strategies for congestion avoidance. In: IEEE. *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*. [S.l.], 2012. p. 265–272. Acesso em: 22 ago. 2025. Citado na página 47.

PAULINO, L. S. Nodejs. *REFAQI-REVISTA DE GESTÃO EDUCAÇÃO EE TECNOLOGIA*, v. 4, n. 2, p. 3–3, 2018. Acesso em: 14 de nov. 2025. Citado na página 55.

RAJABOV, S. B. The role of backend and frontend information systems infrastructure. *Science and Education*, «Open science», v. 4, n. 3, p. 212–216, 2023. Acesso em: 13 de nov. 2025. Citado na página 37.

RIBEIRO, L. Graphql: uma alternativa para webservices. Universidade Estadual de Goiás, 2019. Acesso em: 13 de nov. 2025. Citado na página 40.

RODRIGUES, J. M. Mobilidade urbana no brasil: crise e desafios para as políticas públicas. *Revista do Tribunal de Contas do Estado de Minas Gerais*, v. 34, n. 3, p. 80–93, 2016. Acesso em: 12 de nov. 2025. Citado na página 25.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: A modern approach;[the intelligent agent book]*. [S.l.]: Prentice hall, 1995. Acesso em: 22 ago. 2025. Citado na página 64.

SCHROEDER, R. Html5, um novo desenvolvimento para a web. *Revista on-line de divulgação científica da UNIDAVI*, p. 25, 2012. Acesso em: 13 de nov. 2025. Citado 2 vezes nas páginas 51 e 56.

SILVA, E. M. da. Sistemas de controle adaptativo de semáforos com coleta de dados em tempo real do tráfego nas vias urbanas. *Revista Brasileira em Tecnologia da Informação*, v. 7, n. 1, p. 3–13, 2025. Acesso em: 12 de nov. 2025. Citado na página 46.

SILVA, M. M. da; SANTOS, M. T. P. Os paradigmas de desenvolvimento de aplicativos para aparelhos celulares. *Revista TIS*, v. 3, n. 2, 2014. Acesso em: 12 de nov. 2025. Citado 2 vezes nas páginas 30 e 35.

SILVA, M. S. *Criando sites com HTML: sites de alta qualidade com HTML e CSS*. [S.l.]: Novatec Editora, 2008. Acesso em: 13 de nov. 2025. Citado na página 50.

STETS, J. E.; BIGA, C. F. Bringing identity theory into environmental sociology. *Sociological theory*, Wiley Online Library, v. 21, n. 4, p. 398–423, 2003. Acesso em: 22 ago. 2025. Citado na página 30.

SVENNERBERG, G. *Beginning google maps API 3*. [S.l.]: Apress, 2010. Acesso em: 12 de nov. 2025. Citado na página 32.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. [S.l.]: Pearson Education India, 2016. Acesso em: 22 ago. 2025. Citado na página 21.

TANENBAUM, A. S. *Sistemas operacionais modernos* . [S.l.]: Pearson Educación, 2009. Acesso em: 12 de nov. 2025. Citado na página 35.

TECNOLOGIA, U. *Front-end: o que é, quais as diferenças, linguagens e frameworks. Disponível no*. <https://uds.com.br/blog/front-end-o-que-e-linguagens-frameworks>, 2024. Acesso em: 13 de nov. 2025. Disponível em: <https://uds.com.br/blog/>. Citado na página 37.

TEREKHOV, A. A.; VERHOEF, C. The realities of language conversions. *IEEE Software*, IEEE, v. 17, n. 6, p. 111–124, 2002. Acesso em: 13 de nov. 2025. Citado na página 37.

THOMÉ, M. et al. Um arcabouço para detecçao e alerta de anomalias de mobilidade urbana em tempo real. In: SBC. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. [S.l.], 2020. p. 784–797. Acesso em: 12 de nov. 2025. Citado na página 47.

ULIAN, F. A cidade e o sistema de circulação sob os enfoques sociológico e político do transporte urbano, por eduardo vasconcellos. *Mobilidades desiguais*, p. 103, 2022. Acesso em: 22 ago. 2025. Citado na página 64.

VASCONCELLOS, E. A. de; CARVALHO, C. H. R. de; PEREIRA, R. H. M. *Transporte e mobilidade urbana*. [S.l.], 2011. Acesso em: 22 ago. 2025. Citado na página 16.

WACHS, M. Women's travel issues: creating knowledge, improving policy, and making change. In: *Transportation Research Board Conference Proceedings*. [S.l.: s.n.], 2010. Acesso em: 22 ago. 2025. Citado na página 24.

WARANASHIWAR, J.; UKEY, M. Ionic framework with angular for hybrid app development. *International Journal of New Technology and Research*, Nextgen Research Publication, v. 4, n. 5, p. 263068, 2018. Citado na página 61.

WARD, P. M.; SMITH, C. Housing rehab for consolidated informal settlements: A new policy agenda for 2016 un-habitat iii. *Habitat International*, Pergamon, v. 50, n. 2015, p. 373–384, 2015. Acesso em: 17 de out. 2025. Citado na página 17.

WEISS, M. C. Cidades inteligentes como nova prática para o gerenciamento dos serviços e infraestruturas urbanas: Estudo de caso da cidade de porto alegre. Dissertação de mestrado em Administração de Empresas. Centro Universitário . . . , 2013. Acesso em: 12 de nov. 2025. Citado na página 31.

WILHEIM, J. Mobilidade urbana: um desafio paulistano. *Estudos avançados*, SciELO Brasil, v. 27, p. 7–26, 2013. Acesso em: 12 de nov. 2025. Citado na página 29.

WING, J. M. Computational thinking. *Communications of the ACM*, ACM New York, NY, USA, v. 49, n. 3, p. 33–35, 2006. Acesso em: 22 ago. 2025. Citado na página 22.

WITTEN, I. H. et al. Practical machine learning tools and techniques. In: ELSEVIER PUBLISHERS AMSTERDAM, THE NETHERLANDS. *Data mining*. [S.l.], 2017. v. 2, n. 4, p. 403–413. Acesso em: 22 ago. 2025. Citado na página 42.

Apêndices

.1 Pseudocódigo da Função extract_features_from_station

```
função extrair_features(station, include_traffic):

    distância = station.distancia_m ou 0
    duração = station.route.duration se rota existir
              senão station.duration_s ou 0

    features = { distance_m, duration_s }

    se include_traffic for verdadeiro:
        baseline = distância
        se duração > baseline:
            traffic = (duração / baseline - 1) / 2
            limitar traffic entre 0 e 1
        senão:
            traffic = 0

        adicionar traffic_factor ao conjunto features

    retornar features
```

.2 Pseudocódigo da Função compute_knn_score

```
função calcular_score_knn(stations, k, include_traffic, max_distance):

    se stations estiver vazia:
        retornar lista vazia

    filtrar apenas estações com distance_m <= max_distance

    se nenhuma estação restante:
        retornar vazia

    PARA cada estação filtrada:
        extrair suas features
        adicionar à lista de features
```

```
    converter lista de features para matriz X

    normalizar cada coluna de X no intervalo [0, 1]

    se incluir tráfego:
        pesos = [0.5, 0.4, 0.1]
    senão:
        pesos = [0.5, 0.5]

    calcular score = X_normalizada · pesos

    ordenar estações por menor score

    PARA cada estação ordenada:
        criar registro com:
            osm_id, nome, coordenadas,
            distance_m, duration_s, traffic_factor,
            score, rank, tags

    retornar lista ordenada
```

.3 Pseudocódigo da Classe PredictionView

```
ao receber POST:

    validar dados
    se inválido → retornar erro

    obter lista de stations, k e include_traffic

    se lista de stations vazia → erro

    ranked = calcular_score_knn(stations, k, include_traffic)

    se ranked for vazio → erro

    best = primeira estação da lista
    alternatives = próximas k estações
```

```
    retornar:
        best_station, alternatives, model_features
```

.4 Pseudocódigo da Classe PredictFromLocationView

```
ao receber POST:

    validar dados
    obter lat, lon, radius, k, include_traffic

    ---- Etapa 1: Buscar postos ----
    tentar query_overpass(lat, lon, radius)
    se falhar → erro
    se retornar vazio → erro 404

    ---- Etapa 2: Normalizar ----
    PARA cada posto:
        normalizar usando normalize_station

    ---- Etapa 3: Obter rotas ----
    PARA cada estação normalizada:
        tentar obter rota OSRM (origem → destino)
        se falhar → route = null

    ---- Etapa 4: Calcular ranking ----
    ranked = calcular_score_knn(normalized, k, include_traffic)

    best = primeira estação
    alternatives = próximas k estações

    retornar:
        best_station, alternatives,
        model_features, total de estações encontradas
```

Anexos

Código do Aplicativo em Python

```
def compute_knn_score(stations, k=3, include_traffic=False,
   max_distance_m=15000):
    """
    Use KNN to rank gas stations by:
    - distance_m: straight-line distance
    - duration_s: driving time
    - traffic_factor: estimated traffic (optional)

    Args:
        stations: list of station dicts
        k: number of top results to return
        include_traffic: whether to include traffic factor in scoring
        max_distance_m: maximum distance in meters (default 15 km =
           15000 m)

    Returns ranked list of stations with scores (filtered by
       max_distance_m).
    """
    if not stations:
        return []

    # Filter stations by max distance
    filtered_stations = [s for s in stations if s.get("distance_m",
       0) <= max_distance_m]

    if not filtered_stations:
        return []

    # Extract features
    feature_list = []
    for station in filtered_stations:
        features = extract_features_from_station(station,
           include_traffic=include_traffic)
        feature_list.append(features)

    # Normalize features for KNN
    feature_names = list(feature_list[0].keys())
    X = np.array([[f[name] for name in feature_names] for f in
       feature_list])

```

```
    # Normalize to [0, 1] per column
    X_min = X.min(axis=0)
    X_max = X.max(axis=0)
    X_norm = np.divide(X - X_min, X_max - X_min + 1e-9, where=(X_max - X_min) > 0)

    # Simple scoring: weighted sum of normalized features
    # weight: distance=0.5, duration=0.4, traffic=0.1
    if include_traffic:
        weights = np.array([0.5, 0.4, 0.1])
    else:
        weights = np.array([0.5, 0.5])

    scores = np.dot(X_norm, weights[:len(feature_names)])

    # Rank by score (lower is better)
    ranked_indices = np.argsort(scores)

    results = []
    for rank, idx in enumerate(ranked_indices, start=1):
        station = filtered_stations[idx]
        features = extract_features_from_station(station, include_traffic=include_traffic)
        results.append({
            "osm_id": station.get("osm_id"),
            "name": station.get("name", ""),
            "lat": station.get("lat"),
            "lon": station.get("lon"),
            "distance_m": features["distance_m"],
            "duration_s": features["duration_s"],
            "traffic_factor": features.get("traffic_factor"),
            "knn_score": float(scores[idx]),
            "rank": rank,
            "tags": station.get("tags", {}),  # Include full OSM tags
        })

    return results
```