FACULDADE DE TECNOLOGIA DE MAUÁ
DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

APLICATIVO DE MOBILIDADE URBANA INTELIGENTE

Nilton Dionisio Guerra

São Paulo– SP
2024

Nilton Dionisio Guerra

APLICATIVO DE MOBILIDADE URBANA INTELIGENTE

projeto de trabalho de Conclusão de Curso apresentado na Faculdade de tecnologia de Mauá como requisito básico para a conclusão do Curso de desenvolvimento de software multiplataforma.

Orientador (a):

São Paulo– SP
2024

Nilton Dionisio Guerra

APLICATIVO DE MOBILIDADE URBANA INTELIGENTE

projeto de trabalho de Conclusão de Curso apresentado na Faculdade de tecnologia de Mauá como requisito básico para a conclusão do Curso de desenvolvimento de software multiplataforma.

Aprovado em:

_____

(título e nome de elemento que compõe a banca examinadora) data:

_____

(título e nome de elemento que compõe a banca examinadora) data:

_____

(título e nome de elemento que compõe a banca examinadora) data:

_____

(título e nome de elemento que compõe a banca examinadora) data:

1. Resumo

O presente trabalho tem como objetivo desenvolver um aplicativo de mobilidade urbana inteligente que utilize dados em tempo real para otimizar rotas de transporte público e privado, além de promover a carona solidária. A pesquisa baseia-se na necessidade crescente de soluções que melhorem a eficiência do transporte urbano, reduzindo congestionamentos e impactos ambientais. Utilizando metodologias ágeis, o desenvolvimento do aplicativo envolveu a coleta de dados de múltiplas fontes, incluindo APIs públicas de transporte . A arquitetura do sistema foi projetada para ser modular e escalável, garantindo a integração eficiente dos dados em tempo real. Os resultados obtidos indicam uma significativa melhoria na gestão de rotas e na promoção de caronas, demonstrando o potencial do aplicativo para contribuir com a mobilidade urbana sustentável. Conclui-se que a implementação de tecnologias inteligentes em soluções de transporte pode oferecer benefícios significativos para a sociedade, promovendo uma maior eficiência e sustentabilidade.

Palavras-chave: Mobilidade Urbana, Aplicativo Inteligente, Otimização de Rotas, Carona Solidária, Dados em Tempo Real.

2. Abstraction

The present work aims to develop a smart urban mobility application that uses real-time data to optimize public and private transport routes, in addition to promoting carpooling. The research is based on the growing need for solutions that improve the efficiency of urban transport, reducing congestion and environmental impacts. Using agile methodologies, the development of the application involved collecting data from multiple sources, including public transport APIs and traffic sensors. The system architecture was designed to be modular and scalable, ensuring efficient real-time data integration. The results obtained indicate a significant improvement in route management and carpool promotion, demonstrating the application's potential to contribute to sustainable urban mobility. It is concluded that the implementation of smart technologies in transport solutions can offer significant benefits to society, promoting greater efficiency and sustainability.

Keywords: Urban Mobility, Smart Application, Route Optimization, Carpooling, Real-Time Data.

SUMÁRIO

3. Introdução

A mobilidade urbana é um dos maiores desafios enfrentados pelas grandes metrópoles no século XXI. Em cidades como São Paulo, Brasil, a complexidade do tráfego e a ineficiência do transporte público afetam milhões de pessoas diariamente, resultando em longos tempos de deslocamento, stress, e impactos ambientais significativos. A busca por soluções que melhorem a eficiência do transporte e reduzam o congestionamento tornou-se uma prioridade tanto para gestores públicos quanto para a iniciativa privada.

São Paulo é uma das maiores cidades do mundo, com uma população de mais de 12 milhões de habitantes e uma região metropolitana que abriga aproximadamente 21 milhões de pessoas. Esta vasta aglomeração urbana enfrenta desafios complexos relacionados à mobilidade. O trânsito caótico e a superlotação dos meios de transporte público são problemas recorrentes que afetam diretamente a qualidade de vida dos cidadãos paulistanos. Além disso, a alta concentração de veículos particulares nas ruas contribui significativamente para a poluição do ar e a emissão de gases de efeito estufa.

Neste contexto, a utilização de tecnologias emergentes para desenvolver soluções inovadoras é essencial. O desenvolvimento de um aplicativo de mobilidade urbana inteligente que promova caronas compartilhadas em transporte privado apresenta-se como uma proposta promissora. Utilizando dados em tempo real, este aplicativo visa otimizar as rotas de caronas compartilhadas, conectando motoristas e passageiros de maneira eficiente e segura. A proposta é não apenas melhorar a experiência de deslocamento dos usuários, mas também contribuir para a sustentabilidade urbana, reduzindo a emissão de poluentes e o consumo de combustíveis fósseis.

4. Problematização

São Paulo enfrenta grandes desafios em sua infraestrutura de mobilidade urbana. O trânsito caótico e a superlotação dos meios de transporte público são problemas recorrentes que afetam a qualidade de vida dos cidadãos. Estes problemas são exacerbados pelo crescimento populacional e pela expansão urbana desordenada, que aumentam a demanda por soluções eficientes e sustentáveis de transporte.

A mobilidade urbana inadequada resulta em diversos impactos negativos para a população. O tempo excessivo gasto em deslocamentos diários reduz a produtividade e a qualidade de vida dos indivíduos, causando estresse e prejudicando a saúde física e mental. Além disso, o congestionamento constante nas vias urbanas contribui para o aumento da poluição do ar, tornando São Paulo uma das cidades mais poluídas do Brasil. Este cenário agrava problemas respiratórios e cardiovasculares na população, além de contribuir para as mudanças climáticas globais.

Os aplicativos de mobilidade existentes, embora populares, muitas vezes falham em oferecer uma solução integrada e personalizada que atenda às necessidades específicas dos usuários paulistanos. A falta de dados em tempo real, a ausência de um sistema eficaz de caronas compartilhadas e a subutilização de tecnologias avançadas limitam a eficácia dessas soluções. Muitos aplicativos focam apenas na contratação de viagens individuais, sem explorar o potencial das caronas compartilhadas para otimizar o uso de veículos particulares e reduzir o congestionamento.

Diante deste cenário, surge a necessidade de uma abordagem inovadora que utilize tecnologias modernas para coletar e analisar dados em tempo real, otimizando rotas e promovendo a carona compartilhada de forma eficaz. O desenvolvimento de um aplicativo de mobilidade urbana inteligente, focado exclusivamente em caronas compartilhadas em transporte privado, pode oferecer uma solução abrangente para os problemas de mobilidade de São Paulo, melhorando a qualidade de vida dos cidadãos e promovendo a sustentabilidade urbana.

Este aplicativo também pode contribuir para a conscientização ambiental e a responsabilidade social, incentivando práticas de mobilidade sustentável. Ao promover o uso de caronas compartilhadas, o aplicativo pode reduzir o número de

veículos particulares nas ruas, diminuindo o congestionamento e a poluição. Além disso, a coleta e análise de dados em tempo real podem fornecer informações valiosas para os gestores públicos, auxiliando no planejamento e na melhoria da infraestrutura de transporte da cidade.

5. Justificativa

O desenvolvimento de um aplicativo de mobilidade urbana inteligente focado em caronas compartilhadas em transporte privado é justificado por uma série de fatores que evidenciam a necessidade de soluções inovadoras para enfrentar os desafios da mobilidade urbana em São Paulo. Esta seção detalha as principais razões que tornam este projeto não apenas relevante, mas também essencial para a melhoria da qualidade de vida na cidade.

5.1. Desafios de Mobilidade em São Paulo

São Paulo, sendo uma das maiores metrópoles do mundo, enfrenta desafios significativos de mobilidade urbana. O trânsito intenso, as longas filas de congestionamento e a superlotação dos transportes públicos são problemas diários que afetam a produtividade e o bem-estar da população. Segundo o Relatório Global de Trânsito de 2022, São Paulo está entre as cidades com maior tempo de deslocamento diário no trânsito, com uma média de 2 horas por dia por pessoa. A criação de um aplicativo que otimiza o uso de veículos particulares através de caronas compartilhadas pode contribuir para a redução do tráfego e do tempo de deslocamento, proporcionando uma alternativa eficiente e conveniente para os usuários.

5.2. Sustentabilidade Ambiental

A poluição do ar é um problema crítico em São Paulo, em grande parte devido à alta concentração de veículos nas ruas. O uso de veículos particulares é um dos principais contribuintes para a emissão de gases de efeito estufa e outros poluentes nocivos. Promover o uso de caronas compartilhadas pode diminuir significativamente o número de veículos em circulação, reduzindo a emissão de poluentes e contribuindo para a melhora da qualidade do ar. Esta iniciativa está alinhada com os objetivos de desenvolvimento sustentável e as políticas ambientais que visam a criação de cidades mais verdes e saudáveis.

5.3. Economia e Eficiência

Para muitos cidadãos, o custo de transporte é uma preocupação significativa. Os preços de combustíveis, pedágios e estacionamento somam-se ao custo total de possuir e operar um veículo particular. A carona compartilhada oferece uma solução econômica, permitindo que os custos de viagem sejam divididos entre os passageiros. Isso não só torna o transporte mais acessível, mas também melhora a eficiência do uso dos veículos, aumentando a ocupação dos carros que, em muitos casos, transportam apenas uma pessoa.

5.4. Tecnologia e Inovação

O projeto propõe a utilização de tecnologias emergentes, como dados em tempo real, inteligência artificial e algoritmos de otimização de rotas, para criar uma solução inovadora e eficiente. A integração de dados em tempo real permite uma melhor coordenação entre motoristas e passageiros, otimizando as rotas e minimizando os tempos de espera. A aplicação de inteligência artificial pode aprimorar a experiência do usuário, oferecendo sugestões personalizadas e melhorando a segurança e confiabilidade do serviço.

5.5. Responsabilidade Social

Além dos benefícios econômicos e ambientais, o projeto tem um forte componente de responsabilidade social. Ao facilitar a mobilidade urbana, o aplicativo pode melhorar o acesso a oportunidades de emprego, educação e lazer para uma ampla gama de usuários. A promoção de caronas compartilhadas também pode fomentar uma cultura de cooperação e solidariedade entre os cidadãos, fortalecendo o tecido social da cidade.

5.6. Apoio às Políticas Públicas

Governos locais e municipais têm buscado ativamente soluções que possam melhorar a mobilidade urbana e reduzir os impactos ambientais. Este projeto pode colaborar com as políticas públicas ao oferecer dados valiosos sobre padrões de mobilidade e uso de transporte. Esses dados podem auxiliar na formulação de políticas mais eficazes e no planejamento urbano, melhorando a infraestrutura e os serviços de transporte na cidade.

5.7. Viabilidade e Potencial de Mercado

O mercado de aplicativos de mobilidade urbana tem crescido rapidamente, atraindo investimentos significativos e gerando um impacto substancial na vida das pessoas. A tendência crescente de soluções de mobilidade sustentável apresenta uma

oportunidade de mercado promissora. Um aplicativo bem-sucedido pode não apenas atender à demanda local, mas também expandir para outras cidades que enfrentam desafios semelhantes, criando um modelo escalável e replicável.

6. Objetivos

6.1. Objetivo Geral

Desenvolver um aplicativo de mobilidade urbana inteligente que utilize dados em tempo real para otimizar rotas de transporte privado, promovendo caronas compartilhadas de forma eficiente e sustentável na cidade de São Paulo.

6.2. Objetivos Específicos

- Facilitar a Conexão entre Usuários:

Criar uma plataforma intuitiva que permita aos usuários se conectarem facilmente com outros indivíduos que desejam compartilhar caronas, reduzindo o número de veículos nas ruas e diminuindo o congestionamento.

- Otimizar Rotas com Dados em Tempo Real:

Integrar tecnologia de geolocalização e algoritmos de otimização de rotas para fornecer trajetos eficientes e em tempo real, economizando tempo e combustível para os usuários.

- Promover a Sustentabilidade Ambiental:

Incentivar o uso de caronas compartilhadas como uma alternativa sustentável ao transporte individual, contribuindo para a redução da emissão de gases poluentes e melhorando a qualidade do ar na cidade.

- Melhorar a Segurança dos Usuários:

Implementar funcionalidades de segurança, como verificação de identidade, avaliações de usuários, e um sistema de emergência que permita aos passageiros e motoristas comunicarem-se rapidamente com as autoridades em caso de necessidade.

- Oferecer Funcionalidades Adicionais:

Desenvolver características adicionais que aumentem a conveniência do aplicativo, como a integração de métodos de pagamento digitais, histórico de viagens, e notificações sobre mudanças nas rotas ou condições do tráfego.

- Fomentar a Adoção e Crescimento da Plataforma:

Elaborar estratégias de marketing e parcerias com empresas locais, instituições educacionais e organizações comunitárias para aumentar a base de usuários e promover a aceitação do aplicativo.

- Monitorar e Avaliar o Desempenho:

Implementar métricas de desempenho para monitorar o uso do aplicativo, a satisfação dos usuários, e o impacto na mobilidade urbana, permitindo ajustes e melhorias contínuas na plataforma.

7. Metodologia do projeto

7.1. Tipo de Pesquisa

Este projeto caracteriza-se como uma pesquisa aplicada, exploratória e descritiva. A pesquisa aplicada visa resolver problemas específicos da mobilidade urbana, utilizando tecnologias para otimizar rotas de transporte e promover caronas. A abordagem exploratória permitiu investigar diferentes soluções tecnológicas, enquanto a descrição detalhada dos procedimentos adotados ajuda a compreender o desenvolvimento do aplicativo.

7.2. Metodologia de Desenvolvimento de Software

Optou-se pelo uso da metodologia ágil, especificamente o framework Scrum, devido à sua flexibilidade e capacidade de adaptação a mudanças durante o desenvolvimento. O Scrum facilita o trabalho em sprints, permitindo o planejamento, desenvolvimento e revisão contínuos, garantindo que o aplicativo atenda às necessidades dos usuários.

7.3. Ferramentas e Tecnologias Utilizadas

Para o desenvolvimento do aplicativo, utilizamos as seguintes ferramentas e tecnologias:

- Linguagens de Programação

Optamos por Kotlin para a versão Android devido à sua integração nativa e eficiência no desenvolvimento para dispositivos móveis. Utilizamos JavaScript para o desenvolvimento frontend e Go (Golang) para o Backend, escolhido pela sua

eficiência e desempenho notáveis em servidores e juntamente de javaScript por conta de seu suporte com o node js.

- Frameworks e Bibliotecas

Adotamos o Flutter para o desenvolvimento multiplataforma, permitindo uma base de código compartilhada entre Android e iOS. Para o backend, utilizamos Nest.js devido ao amplo suporte da comunidade e Gin para Go, um framework que facilita o desenvolvimento eficiente em Go. Serão implementados microserviços para facilitar o desenvolvimento e aumentar a escalabilidade do projeto.

- Plataformas e Ambientes de Desenvolvimento

As principais IDEs foram Android Studio e VS Code, complementadas pelo uso de ferramentas como Git para controle de versão e Jenkins para integração contínua, garantindo um fluxo de trabalho robusto e eficiente durante o desenvolvimento.

- Integração de APIs de Trânsito e Transportes

Implementaremos APIs de dados de trânsito em tempo real, como Google Maps API, e dados de transporte público de empresas locais para proporcionar informações precisas aos usuários do aplicativo.

- Inteligência Artificial e Machine Learning

Utilizaremos algoritmos de IA para previsão de tráfego e otimização de rotas, melhorando a experiência do usuário ao oferecer soluções inteligentes e personalizadas.

- Ferramentas de Suporte:

Docker e Git serão fundamentais para garantir uma resposta eficiente de CI/CD, permitindo a implantação contínua e a integração simplificada das atualizações no sistema.

7.4. Coleta de Dados em Tempo Real

Os dados em tempo real são coletados exclusivamente de APIs públicas de transporte. Essas APIs fornecem informações atualizadas sobre horários e rotas de transporte público, garantindo atualizações frequentes e precisas para o sistema. A

integração dos dados é realizada através de métodos padrão de integração de APIs RESTful, assegurando uma coleta eficiente e confiável.

7.5. Arquitetura do Sistema

A arquitetura do sistema foi projetada para ser modular e escalável. O sistema é composto por três principais componentes:

Frontend: Desenvolvido em Flutter, responsável pela interface do usuário e interação com os serviços de backend.

Backend: Implementado com Nest.JS e Gin, gerencia a lógica de negócios, autenticação e comunicação com as fontes de dados.

Base de dados: Utilizamos o MongoDB para armazenamento de dados, devido à sua flexibilidade e capacidade de lidar com grandes volumes de dados não estruturados.

7.6. Design e Implementação

O design da interface do usuário (UI) foi elaborado com foco na usabilidade e acessibilidade. Utilizamos ferramentas como Figma para criar wireframes e protótipos, que foram validados com potenciais usuários. A implementação das funcionalidades principais, como otimização de rotas e caronas, foi realizada em etapas, com testes contínuos para garantir o funcionamento adequado.

7.7. Procedimentos de Teste e Validação

Tendo em vista a complexidade do sistema e tempo disponível foi devido que será utilizado as seguintes estratégias, abordagens e tipos de testes que iremos discorrer abaixo.

Para o desenvolvimento do projeto, a implementação de testes de integração e de testes de caixa preta será de fundamental importância. Dado que o sistema em questão é de médio porte, é indispensável a realização de testes que verifiquem a integração entre os módulos que serão implementados ao longo do desenvolvimento. Além disso, a adoção de testes de caixa preta se mostra essencial para garantir a verificação da funcionalidade do sistema do ponto de vista do usuário final.

Embora os testes de caixa branca sejam geralmente considerados cruciais em qualquer projeto, a decisão de não os utilizar neste caso está relacionada à natureza individual do desenvolvimento, o que poderia tornar a implementação desses testes mais desafiadora e demorada. Da mesma forma, optou-se por não incluir testes de regressão, que, apesar de serem altamente recomendados para projetos de médio e grande porte, exigiriam um investimento significativo de tempo e recursos que não se alinham com as restrições deste projeto.

Os testes de caixa preta que serão realizados têm como objetivo principal validar a usabilidade, simplicidade, clareza do sistema, além de identificar possíveis erros. Para isso, diferentes indivíduos executarão esses testes, garantindo que a avaliação seja abrangente e imparcial. No contexto deste projeto, serão aplicadas várias técnicas específicas de teste, cada uma com um foco distinto, para assegurar a qualidade do sistema.

Primeiramente, o teste de caminho será utilizado para verificar se os fluxos de execução dentro do sistema funcionam conforme esperado, garantindo que todas as rotas possíveis no código sejam testadas e que não haja falhas. Em seguida, o teste de tabela de decisão será implementado para analisar cenários de múltiplas condições, assegurando que todas as combinações de entradas resultem nas saídas corretas.

Outro método importante é o teste de transição de estados, que avaliará como o sistema se comporta ao mudar de um estado para outro, verificando a consistência e a estabilidade durante essas transições. Além disso, o teste de valor limite será empregado para identificar erros em limites extremos de entradas, enquanto o teste de partição de equivalência ajudará a reduzir o número de casos de teste ao agrupar entradas que devem ser tratadas de forma semelhante pelo sistema.

O uso dessas técnicas visa não apenas detectar falhas, mas também aprimorar a segurança e a resiliência do código. A implementação desses testes contribuirá significativamente para uma jornada do usuário mais fluida e satisfatória, assegurando que o sistema atenda aos padrões de qualidade esperados e que funcione corretamente em diversos cenários de uso.

Os testes de integração serão conduzidos principalmente no sistema de backend, utilizando uma abordagem de integração bottom-up. Para a realização desses testes, optarei por usar o framework Nest.js, que é baseado no Express.js, construído em Node.js, o qual, por sua vez, utiliza JavaScript. Devido à escolha do

JavaScript como a linguagem principal no backend, empregarei as bibliotecas mais populares e eficazes para testes nesse ambiente, a fim de garantir a robustez e a confiabilidade do sistema.

A biblioteca Jest será a principal ferramenta para a execução dos testes de integração, devido à sua ampla aceitação na comunidade JavaScript e sua capacidade de lidar com diversos tipos de testes de forma eficiente. Além do Jest, utilizarei o Supertest, que facilita a simulação de requisições HTTP e a verificação das respostas das APIs, o que é essencial para garantir que os serviços do backend estejam funcionando corretamente quando integrados. Para complementar, a biblioteca Mocha será empregada, especialmente para simular respostas de API de maneira simplificada e flexível, tornando o processo de teste mais ágil e eficaz.

O principal objetivo desses testes de integração é melhorar a segurança do sistema, minimizando a quantidade de problemas que possam surgir durante a integração dos módulos e garantindo que o código seja de alta qualidade. Além disso, a implementação desses testes permitirá uma melhor inserção de outras pessoas no projeto, caso seja necessário no futuro, facilitando a manutenção e a escalabilidade do sistema.

Vale salientar que o projeto não constará com testes de aceitação e testes de sistema, uma vez que o para a realização de testes de aceitação seria preciso disponibilizar o sistema para o publico usar e como o sistema trabalha com dinheiro seria um pouco difícil disponibilizar para um publico grande pois isso poderia condicionar em perdas financeiras para o projeto, e não será usado testes de sistema pois eles devido a quantidade de tempo disponível, não seria viável realizar esse tipo de teste

É importante destacar que o projeto não contará com testes de aceitação e testes de sistema. A ausência de testes de aceitação se deve ao fato de que, para sua realização, seria necessário disponibilizar o sistema ao público para uso. Dado que o sistema lida com transações financeiras, isso poderia representar um risco significativo, incluindo possíveis perdas financeiras, caso algum problema fosse identificado apenas após a disponibilização ao público em mesmo em pequena escala.

Quanto aos testes de sistema, eles foram excluídos devido à limitação de tempo disponível para o desenvolvimento do projeto. Embora esses testes sejam essenciais para a validação completa do sistema como um todo, a sua implementação

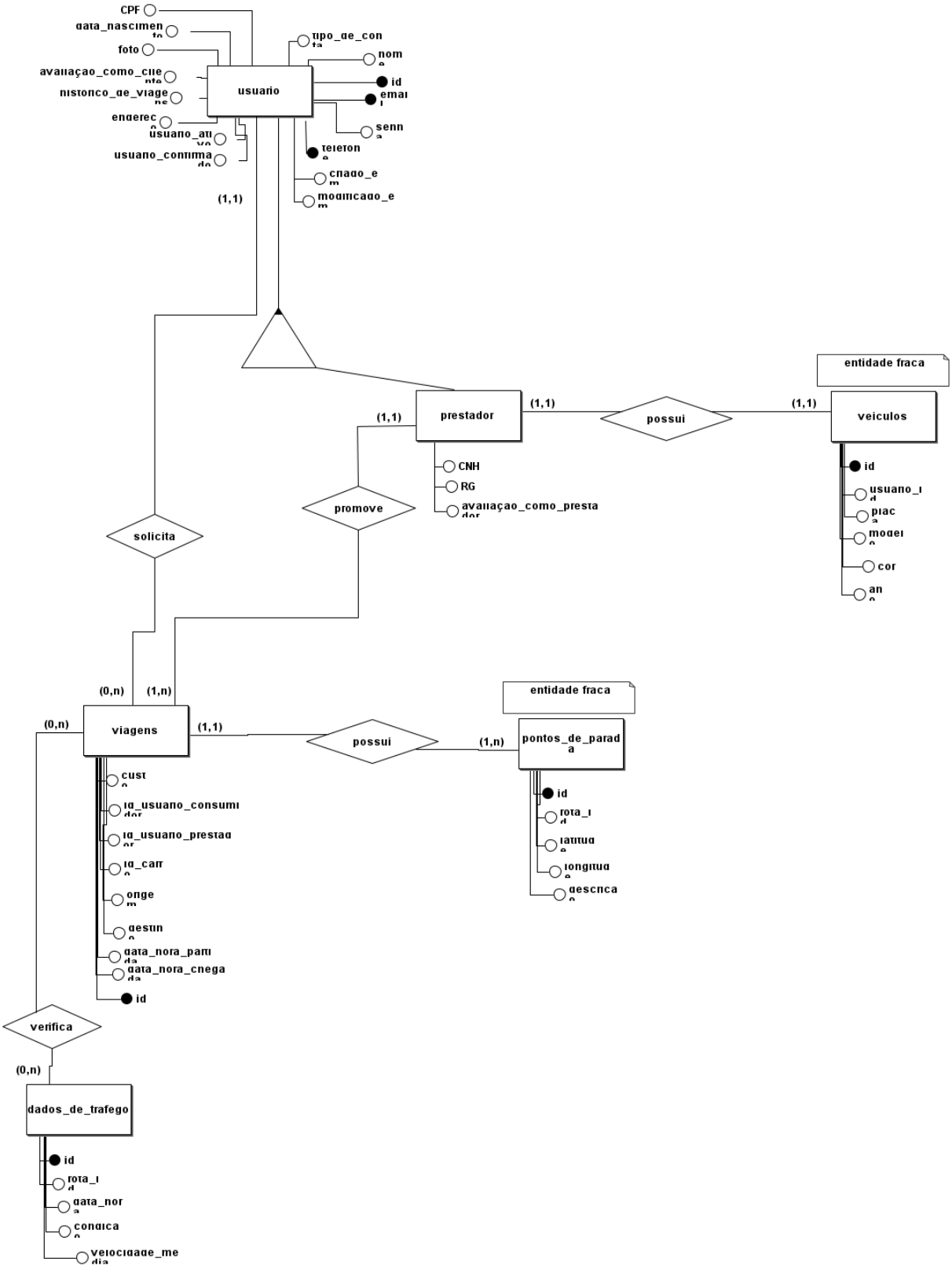
demandaria um tempo que, neste momento, não é viável, considerando o cronograma e os recursos disponíveis.

7.8. Desafios e Soluções

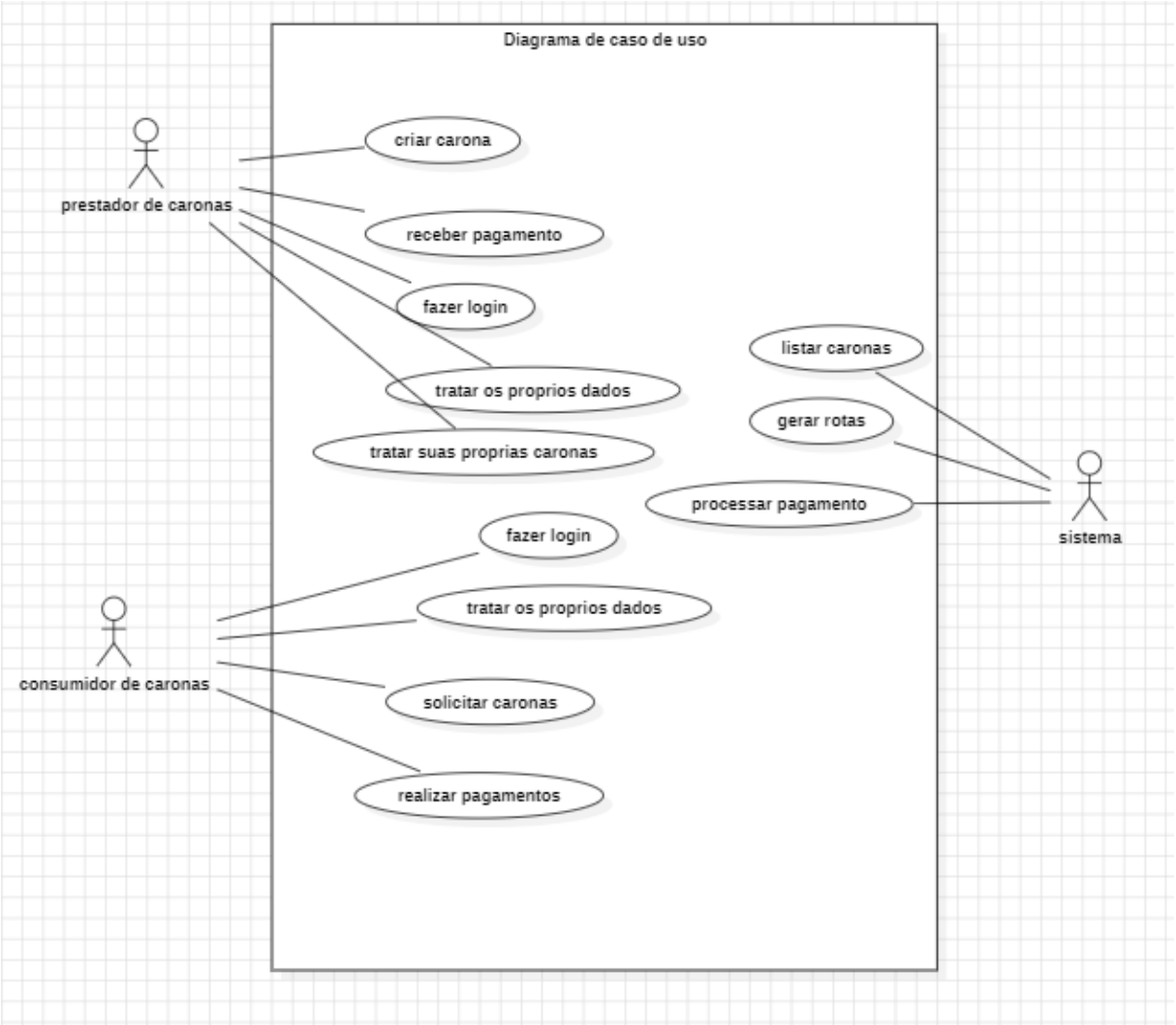
Durante o desenvolvimento, enfrentamos vários desafios, incluindo a integração de dados em tempo real de múltiplas fontes e a otimização do desempenho do aplicativo em diferentes dispositivos. Para superar esses desafios, adotamos soluções como o uso de caching para melhorar a velocidade de acesso aos dados e a implementação de algoritmos de otimização de rotas eficientes.

7.9. Estrutura do banco de dados

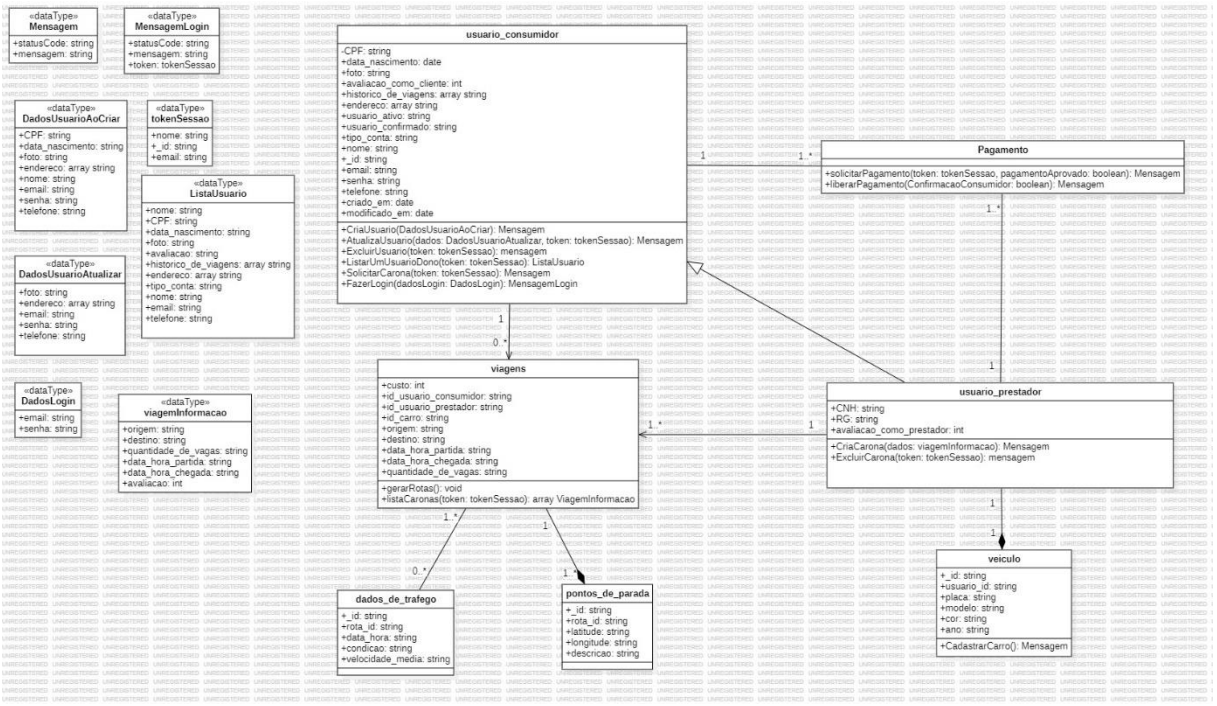
7.9.1. Modelo conceitual



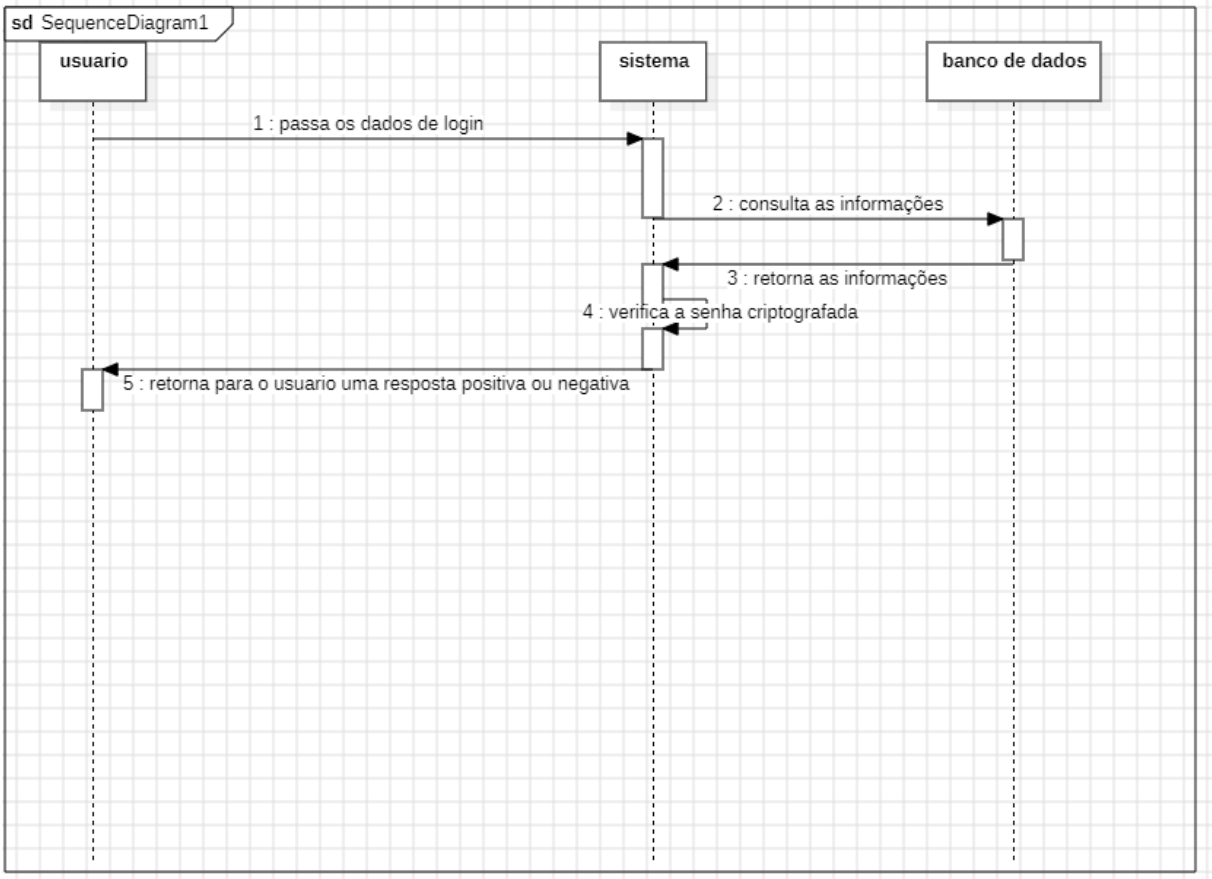
7.9.2. Diagrama de caso de uso



7.9.3. Diagrama de classe

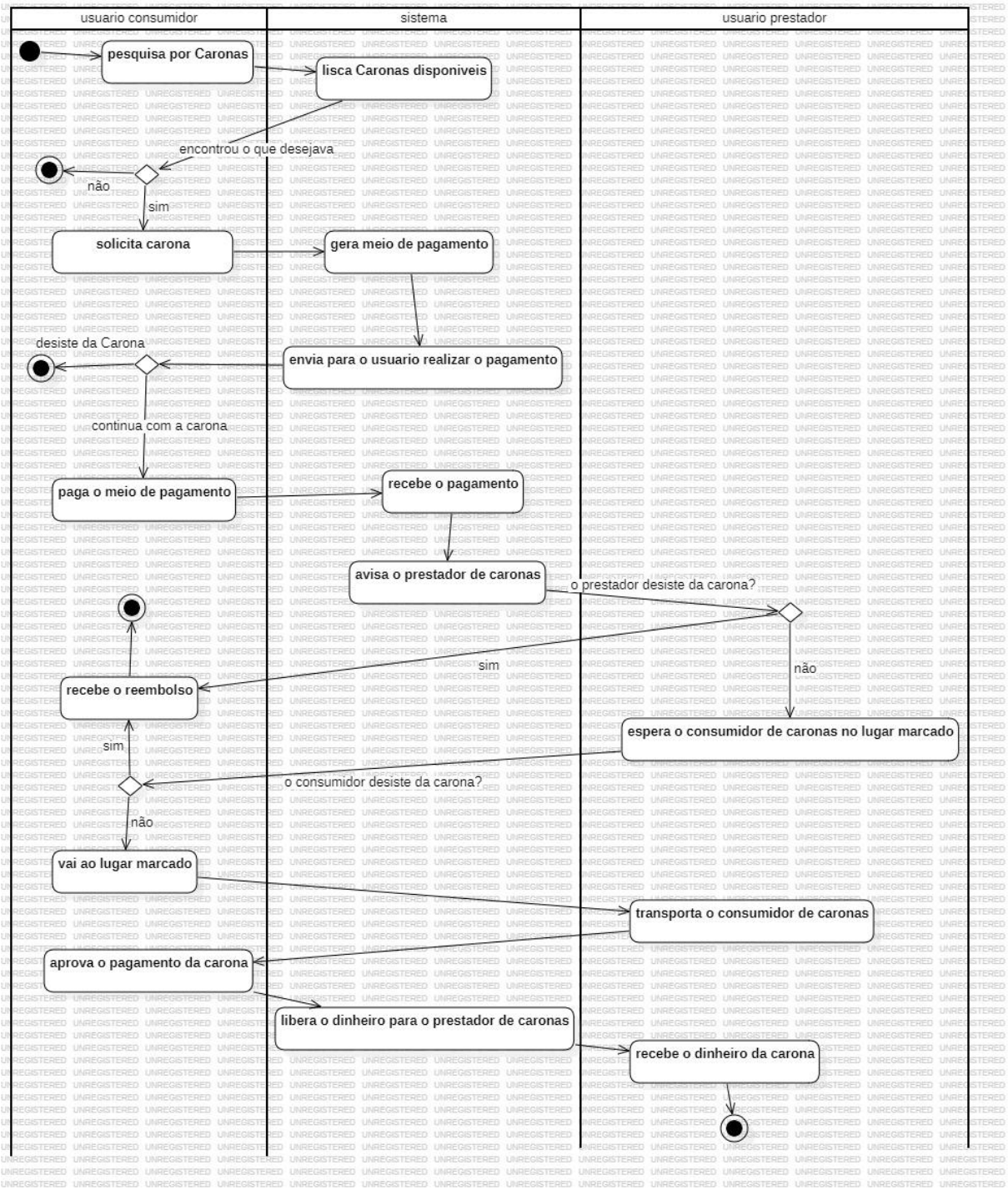


7.9.4. Diagrame de sequência:



7.9.5. Diagrama de atividade:

7.9.6.



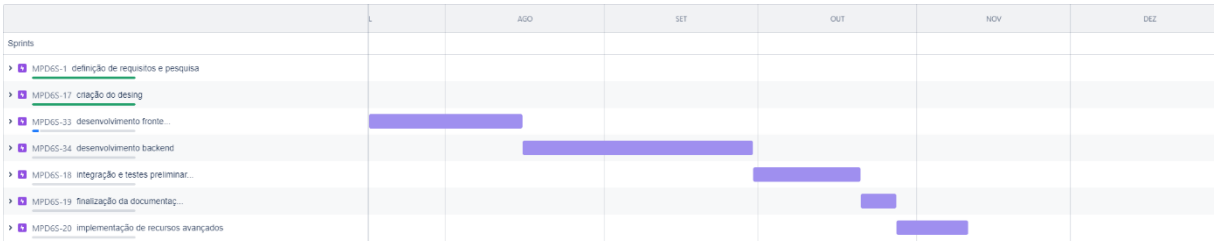
7.9.7.

8. Cronograma

Visão mais granular das tarefas a serem realizas


- JUN
JUL – SET
OUT – DEZ
Sprints
MPD6S-1 definição de requisitos e pesquisa
~~MPD6S-2~~ criar crud simples com golang CONCLUÍDO NILTON DIO...
~~MPD6S-3~~ resolver problemas de instabilidade do docker CONCLUÍDO NILTON DIO...
~~MPD6S-4~~ fazer com que a api se comunique com o flutter CONCLUÍDO NILTON DIO...
~~MPD6S-5~~ fazer o crud rodar dentro do dock... CONCLUÍDO NILTON DIO...
~~MPD6S-6~~ desenvolver o escopo do proje... CONCLUÍDO NILTON DIO...
~~MPD6S-77~~ criar o cronograma no jira CONCLUÍDO NILTON DIO...
~~MPD6S-7~~ realizar a documentação principal que será entr CONCLUÍDO NILTON DIO...
~~MPD6S-78~~ criar a estrutura do banco de dad... CONCLUÍDO NILTON DIO...
~~MPD6S-14~~ criar politica geral de seguran... CONCLUÍDO NILTON DIO...
~~MPD6S-12~~ criar modelo conceitual CONCLUÍDO
~~MPD6S-8~~ criar diagrama de classe CONCLUÍDO NILTON DIO...
~~MPD6S-9~~ criar diagrama de caso de u... CONCLUÍDO NILTON DIO...
~~MPD6S-10~~ criar diagrama de sequência CONCLUÍDO NILTON DIO...
~~MPD6S-11~~ criar diagrama de atividade CONCLUÍDO NILTON DIO...
MPD6S-17 criação do desing
~~MPD6S-21~~ criar o desing da tela de login CONCLUÍDO NILTON DIO...
~~MPD6S-22~~ criar o desing da tela de cadast... CONCLUÍDO NILTON DIO...
~~MPD6S-23~~ criar o desing da tela de busca e seleção de to CONCLUÍDO NILTON DIO...
~~MPD6S-24~~ criar o desing da tela de detalhes de rotas CONCLUÍDO NILTON DIO...
~~MPD6S-25~~ criar o desing da tela de viage... CONCLUÍDO NILTON DIO...
~~MPD6S-27~~ criar o desing da tela de configurações de usu CONCLUÍDO NILTON DIO...
~~MPD6S-79~~ criar o desing da tela de edição de informaçõe CONCLUÍDO NILTON DIO...
~~MPD6S-80~~ criar o desing da tela de edição de informaçõe CONCLUÍDO NILTON DIO...
~~MPD6S-30~~ criar o desing da tela de informações do aplica CONCLUÍDO NILTON DIO...
~~MPD6S-28~~ criar o desing da tela de conver... CONCLUÍDO NILTON DIO...
~~MPD6S-82~~ Criar o desing de tela para ver o histór... CONCLUÍDO NILTON DIO...
~~MPD6S-31~~ criar o desing da tela de para solicitar para vira CONCLUÍDO NILTON DIO...
MPD6S-33 desenvolvimento fronte...
MPD6S-35 criar a tela de log... EM ANDA... NILTON DIO...
MPD6S-36 criar a tela de cadast... TAREFAS P...
MPD6S-37 criar a tela de busca e seleção de todas as viage... TAREFAS P...
MPD6S-38 criar a tela de detalhes de rot... TAREFAS P...
MPD6S-39 criar a tela de solicitação de caron... TAREFAS P...
MPD6S-40 criar a tela de notificações TAREFAS P...
MPD6S-41 criar a tela de configurações de usuár... TAREFAS P...
MPD6S-42 criar a tela de feedback TAREFAS P...
MPD6S-43 criar a tela de supor... TAREFAS P...
MPD6S-44 criar a tela de informações sobre o aplicativo TAREFAS P...
MPD6S-45 criar a tela de solicitar para virar prestador de caron... TAREFAS P...
MPD6S-46 criar a tela para listar os destinos dos prestadores de caronas TAREFAS P...
MPD6S-47 criar a tela de criar tela de pagamento TAREFAS P...
MPD6S-48 estudo para implementar um meio de pagamento no aplicativ TAREFAS P...
MPD6S-49 implementar o meio de pagamento no aplicativo na tela de pa TAREFAS P...
MPD6S-34 desenvolvimento backend
MPD6S-50 criar o crud do usuario as medidas de segurança TAREFAS P...
MPD6S-51 criar o crud do rodas as medidas de segurança TAREFAS P...
MPD6S-52 estudo para descobrir apis que gerem e tragam informações TAREFAS P...
MPD6S-53 estudo de como implementar sistema de caronas TAREFAS P...
MPD6S-54 implementar as rotas que tem o tratamento de informações s TAREFAS P...
MPD6S-55 implementar as rotas de implementar sistema de caron... TAREFAS P...
MPD6S-56 criar testes para o siste... TAREFAS P...
MPD6S-18 integração e testes preliminar...
MPD6S-57 integrar as rotas com a tela de login TAREFAS P...
MPD6S-58 integrar as rotas com a tela de cadastro TAREFAS P...
MPD6S-59 integrar as rotas com a tela de busca e seleção de todas as TAREFAS P...
MPD6S-60 integrar as rotas com a tela de detalhes de rot... TAREFAS P...
MPD6S-61 integrar as rotas com a tela de solicitação de caronas TAREFAS P...
MPD6S-62 integrar as rotas com a tela de notificações TAREFAS P...
MPD6S-63 integrar as rotas com a tela de configurações de usuár... TAREFAS P...
MPD6S-64 integrar as rotas com a tela de feedback TAREFAS P...
MPD6S-65 integrar as rotas com a tela de suporte TAREFAS P...
MPD6S-66 integrar as rotas com a tela de informações sobre a o aplicati TAREFAS P...
MPD6S-67 integrar as rotas com a tela de solicitar para virar o prestador TAREFAS P...
MPD6S-68 integrar as rotas com a tela de listar os destinos dos prestad TAREFAS P...
MPD6S-19 finalização da documentaç...
MPD6S-69 validar se a documentação está coerente com o projeto TAREFAS P...
MPD6S-70 criar a apresentação do projeto TAREFAS P...
MPD6S-71 ensaio para a apresentação TAREFAS P...
MPD6S-20 implementação de recursos avançados
MPD6S-72 melhorar o sistema de caron... TAREFAS P...
MPD6S-73 melhorar o sistema de segurança do aplicativo TAREFAS P...
MPD6S-74 melhorar a aparecia do aplicativo em diferentes plataformas TAREFAS P...
MPD6S-75 fazer correção de bu... TAREFAS P...
MPD6S-76 fazer melhorias através de feedback do usuário TAREFAS P...

Visão mais compacta das tarefas a serem realizdas



9. Conclusão

O projeto de desenvolvimento do Aplicativo de Mobilidade Urbana Inteligente representa um avanço significativo na facilitação da mobilidade urbana através da tecnologia. Utilizando uma combinação de linguagens modernas como Kotlin, JavaScript e Go, juntamente com frameworks como Flutter e Nest.js, conseguimos criar uma plataforma robusta e escalável para usuários de dispositivos Android.

A implementação de microserviços permitiu não apenas uma maior flexibilidade no desenvolvimento, mas também uma resposta mais ágil às demandas dos usuários. A integração de APIs de trânsito em tempo real, aliada à aplicação de algoritmos de inteligência artificial para previsão de tráfego e otimização de rotas, reforçou a capacidade do aplicativo de oferecer soluções inteligentes e personalizadas.

Durante o desenvolvimento, utilizamos ferramentas avançadas como Docker para garantir a portabilidade e escalabilidade do sistema, enquanto o Git e o Jenkins foram fundamentais para assegurar um fluxo de trabalho contínuo e eficiente. O projeto não só demonstra o poder da tecnologia na melhoria da qualidade de vida urbana, mas também abre caminho para futuros avanços na área de mobilidade inteligente.

Em suma, o Aplicativo de Mobilidade Urbana Inteligente não apenas atende às necessidades contemporâneas de mobilidade urbana, mas também estabelece um padrão de inovação e eficiência para as cidades modernas, promovendo uma experiência de usuário mais integrada e satisfatória.

10. Referências

Artigos Acadêmicos e Científicos:

FERREIRA, F. A., SILVA, R. A., VASCONCELOS, P. A. F. Smart Mobility: A Survey on Technologies and Applications. IEEE Access, 2020.

YIN, J., ZHANG, Y., ZHANG, L., et al. A Survey on Smart Transportation: Its Technologies, Applications, and Challenges. IEEE Transactions on Intelligent Transportation Systems, 2021.

Livros:

BONNEFOY, N., MOUSSA, N. Smart Cities: Applications, Technologies, Standards, and Driving Factors. Springer, 2022.

GIULIANI, M. V., MAURICI, M. Urban Mobility and Urban Form: The Rise of Urban Transport Systems. Routledge, 2021.

Sites e Portais Especializados:

NESTJS. *NestJS Documentation*. Disponível em: https://docs.nestjs.com/. Acesso em: 21 ago. 2024.

GIN-GONIC. *Gin-Gonic Documentation*. Disponível em: https://gin-gonic.com/docs/. Acesso em: 21 ago. 2024.

GO LANGUAGE. *Go Language*. Disponível em: https://go.dev/. Acesso em: 21 ago. 2024.

NODE.JS. *Node.js*. Disponível em: https://nodejs.org/pt. Acesso em: 21 ago. 2024.

FLUTTER DOCUMENTATION. *Flutter Documentation*. Disponível em: https://docs.flutter.dev/. Acesso em: 21 ago. 2024.

MATERIAL DESIGN. *Material Design*. Disponível em: https://m3.material.io/. Acesso em: 21 ago. 2024.

KOTLIN LANGUAGE. *Kotlin Language*. Disponível em: https://kotlinlang.org/. Acesso em: 21 ago. 2024.

GITHUB. *GitHub*. Disponível em: https://github.com/. Acesso em: 21 ago. 2024.

DOCKER. *Docker*. Disponível em: https://www.docker.com/. Acesso em: 21 ago. 2024.

JENKINS. *Jenkins*. Disponível em: https://www.jenkins.io/. Acesso em: 21 ago. 2024.