FACULDADE DE TECNOLOGIA DE MAUÁ
DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

Relatório Técnico

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): Luana Lourenco

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:

_____

Luana Lourenco data:

_____

Andreza Maria de Souza Rocha data:

_____

Suely dos Santos Souza data:

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 Application Programming Interfaces 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; caronas; economia circular.

Abstract

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

1. 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 superior a 12 milhões de habitantes, enquanto sua região metropolitana abriga cerca de 21 milhões de pessoas. Esta vasta concentração urbana enfrenta desafios complexos relacionados à mobilidade, como o trânsito caótico e a superlotação dos meios de transporte público, problemas 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, fatores que tornam os níveis de poluição da cidade consistentemente superiores aos limites recomendados pela Organização Mundial da Saúde (IBGE, 2024; CIDADE DE SÃO PAULO, 2022).

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.

Este projeto possui uma metodologia que se divide em quatro partes na seção de Metodologia. Sendo no item 5.1 aborda as arquiteturas e metodologias de desenvolvimento utilizadas na plataforma, detalhando a arquitetura do Frontend e do Backend, bem como as tecnologias escolhidas para o desenvolvimento. No item 5.2,

são explorados os mecanismos de manutenção do sistema, explicando como cada um opera e a razão de sua escolha. No item 5.3 apresenta a modelagem e diagramação do sistema, descrevendo a estrutura do banco de dados, o funcionamento do sistema, os métodos necessários para sua implementação, os parâmetros a serem configurados e como as funcionalidades se comunicam, além de incluir diagramas para orientar o desenvolvimento. Finalmente, no item 5.4 é dedicado ao cronograma do projeto, oferecendo uma visão detalhada das tarefas realizadas durante o desenvolvimento.

2. Problematização

São Paulo enfrenta significativos desafios em sua infraestrutura de mobilidade urbana. O trânsito intenso e a superlotação dos meios de transporte público permanecem problemas recorrentes que impactam diretamente a qualidade de vida dos cidadãos. Esses problemas são agravados pelo crescimento populacional e pela expansão urbana desordenada, os quais intensificam a necessidade de soluções de transporte eficientes e sustentáveis (IBGE, 2024; CIDADE DE SÃO PAULO, 2022).

A mobilidade urbana inadequada gera diversos impactos negativos para a população. O tempo excessivo gasto em deslocamentos diários reduz a produtividade e a qualidade de vida dos indivíduos, ocasionando estresse e comprometendo a saúde física e mental. Além disso, o congestionamento constante nas vias urbanas contribui para o aumento da poluição do ar, o que torna São Paulo uma das cidades mais poluídas do Brasil. Este cenário agrava problemas respiratórios e cardiovasculares na população, além de intensificar as mudanças climáticas globais (MOBILIDADE URBANA, 2023; POLUIÇÃO DO AR, 2016).

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 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 promover as caronas compartilhadas 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.

3. 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.

- 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. De acordo com o "Estudo Mobilize 2022", São Paulo está entre as cidades com maior tempo de deslocamento diário no trânsito, evidenciando que em média, uma pessoa passa cerca de duas horas por dia em trânsito (MOBILIZE, 2022). A adoção de aplicativos que otimizam o uso de veículos particulares através de caronas compartilhadas é uma estratégia emergente para mitigar esses desafios, prometendo reduzir tanto o congestionamento quanto o tempo de deslocamento, proporcionando uma alternativa eficiente e conveniente para os usuários.

- 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(CETESB, 2022).

- 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 (WORLD BANK, 2022).

- 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

.

- Apoio às Políticas Pública

Governos locais e municipais têm buscado ativamente soluções para 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, auxiliando na formulação de políticas mais eficazes e no planejamento urbano para aprimorar a infraestrutura e os serviços de transporte na cidade (WORLD BANK, 2022).

- 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(McKinsey, 2023).

4. Objetivos

- Objetivo Geral

Desenvolver um aplicativo de mobilidade urbana inteligente que permita promover caronas compartilhadas de forma eficiente e sustentável na cidade de São Paulo.

- 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.

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.

5. Metodologia do projeto

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

5.1. Arquiteturas e metodologias de desenvolvimento

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.

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

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.

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 backend, escolhido pela sua eficiência e facilidade notáveis em servidores.

- 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 a sua facilidade e simplicidade e aderencia ao mercado, um framework que facilita o desenvolvimento eficiente em JS.

- 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.

- 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.

5.2. mecanismos de manutenção do sistema

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.

- Testes de caixa preta

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.

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.

- Testes de integração

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.

- Testes de caixa branca

Apesar de os testes de caixa branca serem geralmente considerados fundamentais em qualquer projeto de desenvolvimento, optou-se por não implementá-los neste caso. Essa decisão está relacionada à natureza individual do desenvolvimento, que poderia tornar a implementação desses testes especialmente

desafiadora e demorada. Dado o contexto específico deste projeto, concluiu-se que os recursos e o tempo disponíveis seriam mais eficazmente aplicados em outras áreas do desenvolvimento, de modo a garantir a conclusão eficiente e dentro dos prazos estabelecidos.

- Testes de regressão

Optou-se por não incluir testes de regressão neste projeto, embora sejam amplamente recomendados, especialmente em projetos de médio e grande porte. Esses testes desempenham um papel crucial na garantia de que novas alterações no código não comprometam funcionalidades já existentes. Contudo, a implementação de testes de regressão eficazes exigiria um investimento significativo de tempo e recursos, o que não se alinha com as restrições impostas a este projeto. Após uma análise cuidadosa, concluiu-se que, dadas as limitações atuais, o foco deve ser direcionado para os aspectos mais críticos e viáveis do desenvolvimento.

- Testes de aceitação

É importante destacar que o projeto não contará com testes de aceitação pois 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 mesmo em pequena escala.

- Testes de sistema

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.

- Ferramentas utilizadas para a criação dos testes

Para a realização dos testes unitários, uma abordagem essencial nos testes de integração, escolhemos as bibliotecas Jest, Supertest, e mock no Backend. Essas ferramentas foram selecionadas pela sua eficiência em conjunto com o Nest.js, que é o framework utilizado no projeto. O Jest foi escolhido por ser uma das bibliotecas mais populares para testes em JavaScript, oferecendo suporte robusto para testes unitários e de integração. Supertest foi selecionado para testar requisições HTTP, permitindo verificar se os endpoints funcionam corretamente. Já o mock foi integrado para simular dependências e isolar os testes, assegurando que possamos testar partes individuais do código sem influências externas, alguns dos motivos que nos levaram a usar essas ferramentas são:

Jest: A escolha do Jest foi motivada pela sua flexibilidade e ampla aceitação na comunidade de desenvolvimento JavaScript. Ele se integra perfeitamente com o Nest.js, além de oferecer uma sintaxe simples para criar testes e relatórios abrangentes, o que facilita a identificação de erros.

Supertest: A decisão de usar o Supertest está diretamente relacionada à sua capacidade de simular e testar endpoints de forma eficiente. Ao utilizá-lo em conjunto com o Jest, conseguimos realizar testes de integração robustos, garantindo que as APIs do Backend funcionem conforme esperado.

mock: Esta biblioteca foi selecionada para criar cenários de testes onde dependências externas (como chamadas a APIs de terceiros) não afetem os resultados, garantindo a precisão dos testes unitários.

Para os testes de API, optamos pelo uso do Postman, uma ferramenta amplamente reconhecida pela sua facilidade em simular requisições HTTP. A escolha do Postman foi baseada em sua capacidade de testar rapidamente os endpoints do Backend, simulando as interações que o Frontend terá com o Backend. Com ele, pudemos verificar se todas as rotas e respostas estavam corretas, facilitando a integração entre as duas camadas e identificando rapidamente possíveis falhas.

No Frontend, que é focado em um aplicativo mobile desenvolvido majoritariamente em Flutter, decidimos utilizar o Appium para os testes. O Appium foi escolhido por sua compatibilidade com aplicações móveis híbridas e nativas, sendo uma ferramenta ideal para automatizar testes em múltiplas plataformas. Como nosso projeto se baseia fortemente em dispositivos móveis, o Appium se mostrou

fundamental para testar a interface do usuário e as funcionalidades em diferentes ambientes, garantindo a qualidade e usabilidade do aplicativo.

Além disso, decidimos utilizar a biblioteca Color Contrast Checker para melhorar a acessibilidade da plataforma. Essa biblioteca permite verificar o contraste de cores dentro do aplicativo, garantindo que as combinações de cores sejam acessíveis para todos os usuários, especialmente aqueles com deficiência visual. Com isso, buscamos tornar a experiência mais inclusiva e conforme os padrões de acessibilidade.

- Avaliação das ferramentas

Jest e Supertest supriram as nossas necessidades, oferecendo uma maneira prática e eficiente de escrever e manter testes de Backend. A integração fluida entre estas bibliotecas e o Nest.js ajudou a aumentar a confiabilidade e cobertura dos testes, permitindo detectar problemas de maneira rápida e precisa.

Postman também atendeu nossas expectativas ao facilitar a simulação de requisições API. Sua interface gráfica permitiu uma configuração rápida de cenários de testes e ajudou a verificar se os endpoints criados funcionavam corretamente, acelerando a fase de integração com o Frontend.

Appium, por sua vez, se provou eficiente na automação de testes para dispositivos móveis, fornecendo uma plataforma robusta para garantir que o aplicativo desenvolvido em Flutter funcionasse adequadamente em diferentes dispositivos.

A biblioteca Color Contrast Checker supriu nossas necessidades, oferecendo uma solução eficaz para garantir a acessibilidade visual dentro do aplicativo. Sua integração foi simples e rápida, permitindo uma verificação eficiente do contraste de cores entre texto e fundo. A ferramenta nos ajudou a garantir que as combinações de cores atendam aos padrões exigidos pelas diretrizes de acessibilidade WCAG, facilitando a adequação aos níveis de conformidade AA e AAA.

Além disso, a análise automática proporcionada pela biblioteca foi crucial para identificar possíveis problemas de contraste, permitindo ajustes rápidos na interface do usuário. Isso acelerou o processo de desenvolvimento e garantiu que o aplicativo fosse mais inclusivo para usuários com deficiências visuais.

- Pontos de melhoria e observações após o uso

Jest: Durante o uso, observamos que o Jest oferece uma excelente documentação e ferramentas para testes de componentes, porém, em casos de testes mais complexos envolvendo muitos mocks, houve uma pequena curva de aprendizado.

Supertest: Suas funcionalidades se mostraram extremamente úteis para testar APIs, mas em alguns casos mais avançados, a criação de testes de integração pode se tornar um pouco trabalhosa.

Postman: Embora o Postman seja bastante intuitivo, ele exige a criação manual de algumas requisições, o que pode tornar o processo de teste repetitivo para grandes volumes de endpoints. Uma solução poderia ser integrar o Postman com um sistema de automação de testes.

Appium: Funcionou bem para nossos testes móveis, mas notamos que configurar corretamente os ambientes para testes em múltiplos dispositivos pode ser demorado. Isso pode ser um ponto de atenção em projetos futuros, onde a variedade de dispositivos pode crescer.

Color Contrast Checker: Durante o uso, a biblioteca atendeu bem às nossas expectativas, permitindo a verificação rápida e precisa do contraste de cores. No entanto, um ponto de melhoria seria a inclusão de uma funcionalidade que oferecesse sugestões automáticas de ajuste de cores quando as combinações não atendem aos critérios de acessibilidade. Atualmente, a ferramenta apenas indica se o contraste está inadequado, mas cabe ao desenvolvedor buscar manualmente as alternativas. Isso poderia acelerar o processo de ajuste em projetos com muitas combinações de cores.

Em resumo, as ferramentas escolhidas foram fundamentais para garantir a qualidade e eficiência dos testes tanto no Backend quanto no Frontend, atendendo bem às necessidades do projeto até o momento. Embora algumas pequenas limitações tenham sido identificadas, todas as ferramentas proporcionaram ganhos significativos em termos de produtividade e confiança na entrega final do projeto.

5.3. modelagem e diagramação do sistema

Este capítulo abordará aspectos cruciais para o desenvolvimento do sistema, incluindo a modelagem de dados, a modelagem do sistema e a criação de diagramas que orientam o fluxo de desenvolvimento. A modelagem de dados é essencial para definir como as informações serão armazenadas e organizadas no banco de dados, garantindo integridade e eficiência nas operações. Já a modelagem do sistema proporciona uma visão abrangente das interações entre os diversos componentes, permitindo que o desenvolvimento siga uma estrutura lógica e coerente.

Além disso, serão apresentados os diagramas desenvolvidos para representar visualmente as relações entre as entidades do sistema, o fluxo de dados e a arquitetura geral. Esses diagramas servem como guias fundamentais para o desenvolvimento, facilitando a comunicação entre os membros da equipe e assegurando que todos os elementos do sistema estejam bem integrados e funcionando em harmonia.

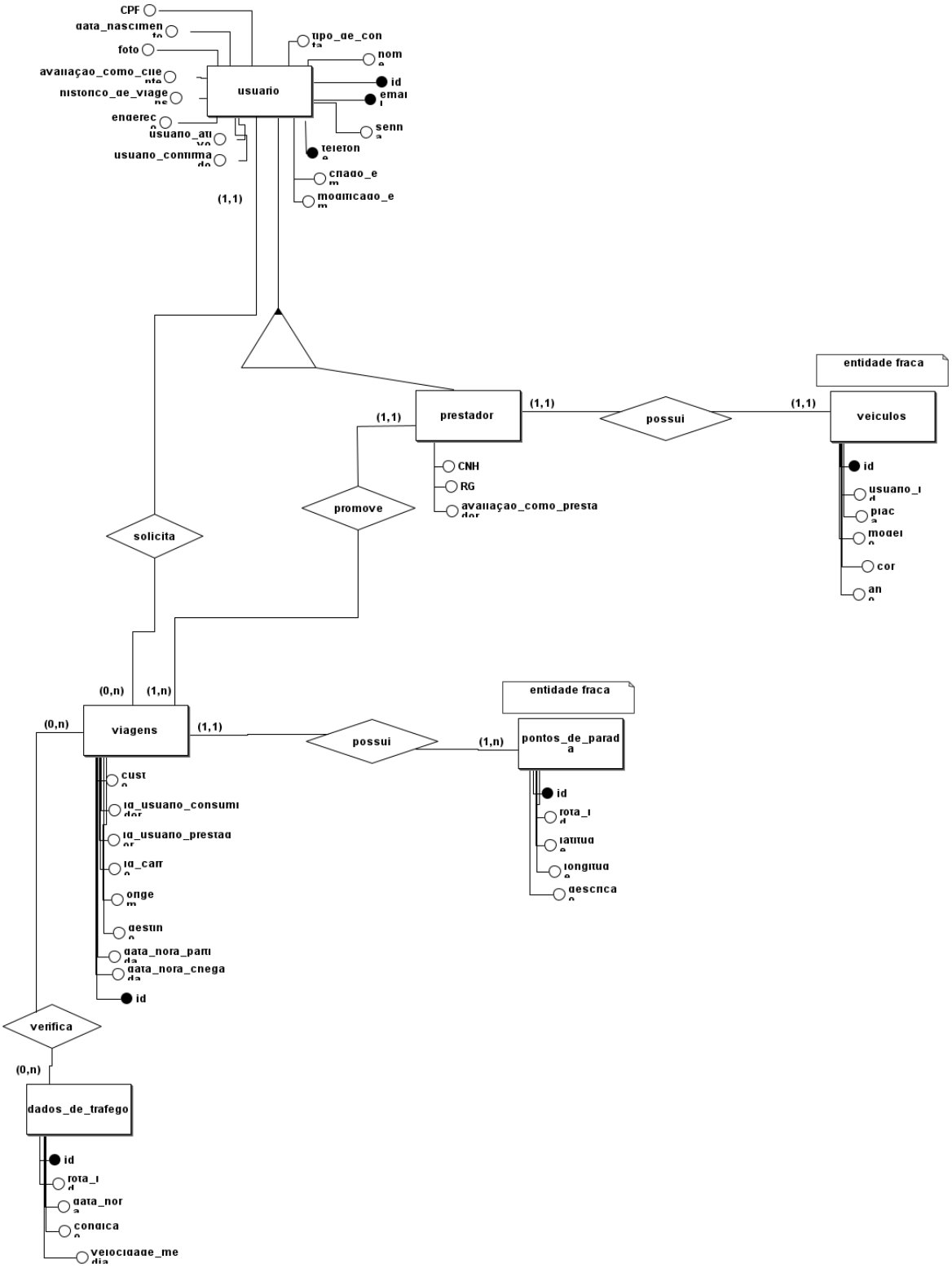
- Modelo conceitual

O diagrama de modelo conceitual apresentado a seguir é uma representação abstrata das principais entidades e suas relações dentro do sistema. Este modelo serve como um esboço inicial que define as bases para o desenvolvimento subsequente da estrutura de dados e da lógica de negócios.

O objetivo do modelo conceitual é fornecer uma visão clara e simplificada dos elementos centrais que compõem o sistema, sem se preocupar ainda com os detalhes de implementação. Nele, as entidades são identificadas e suas interações são delineadas, facilitando a compreensão das necessidades do sistema e garantindo que todas as partes interessadas compartilhem uma visão comum.

Esse diagrama desempenha um papel crucial no planejamento do banco de dados, pois ajuda a identificar as relações entre as entidades, os atributos essenciais e as possíveis interdependências. Com essa visão conceitual em mãos, a equipe de desenvolvimento pode prosseguir com maior segurança para as etapas de modelagem lógica e física, assegurando que o sistema atenda aos requisitos funcionais e não funcionais definidos.

Esquema 1 - Modelo Conceitual



Fonte:autor

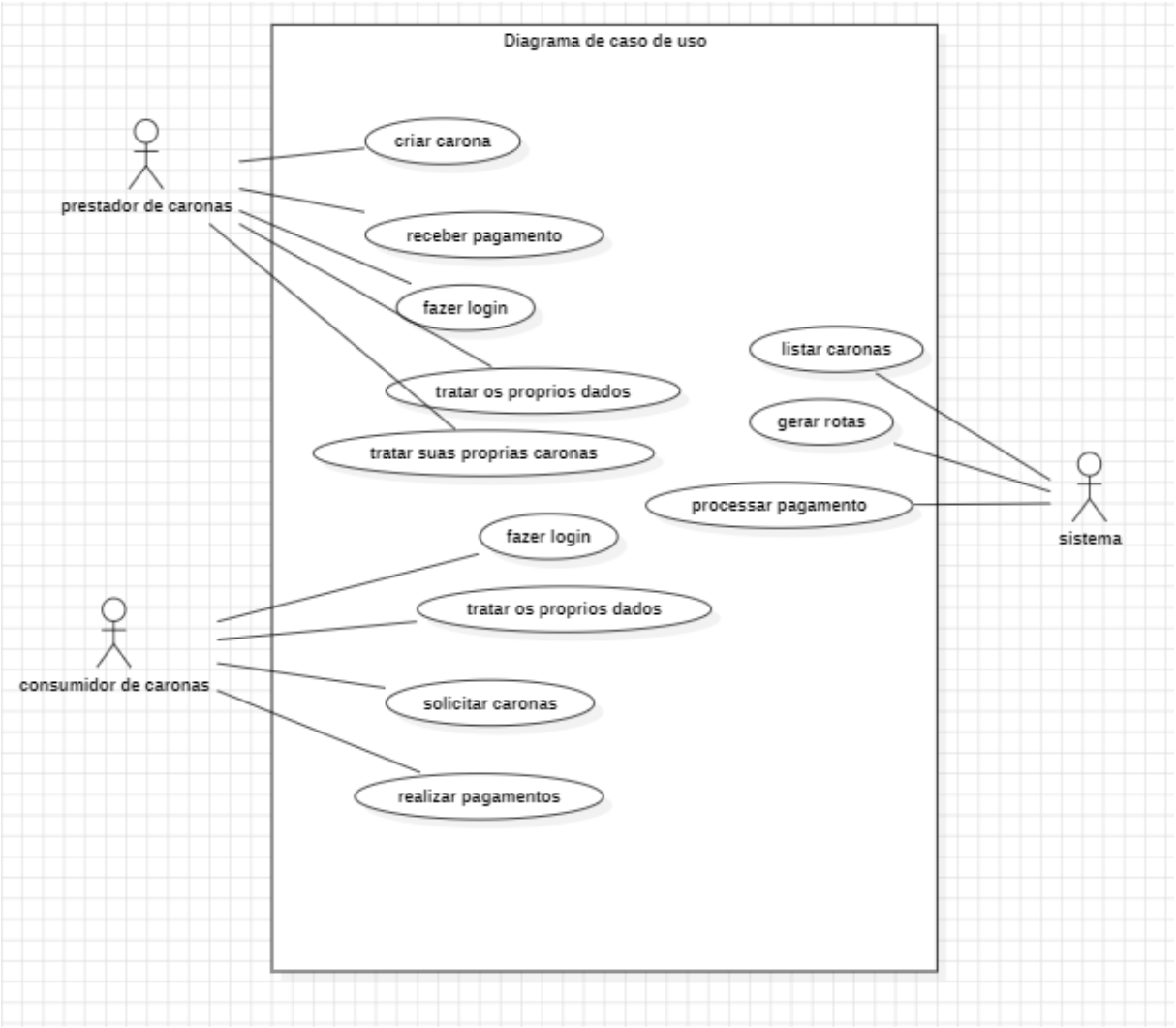
- Diagrama de caso de uso

O diagrama de caso de uso apresentado a seguir é uma ferramenta fundamental para ilustrar as interações entre os usuários (ou atores) e o sistema. Ele representa de forma clara e objetiva os diferentes cenários em que o sistema será utilizado, identificando as funcionalidades principais que devem ser atendidas.

Cada caso de uso descreve uma funcionalidade específica do sistema, mostrando como os atores externos, como usuários ou outros sistemas, interagem com o sistema para alcançar um objetivo particular. Esse diagrama não só ajuda a entender as necessidades do usuário, como também orienta o processo de desenvolvimento ao assegurar que todas as funcionalidades essenciais sejam contempladas.

O diagrama de caso de uso é especialmente valioso nas fases iniciais do projeto, pois facilita a comunicação entre as partes interessadas, permitindo que todos compreendam como o sistema deve se comportar em diferentes situações. A partir desse diagrama, pode-se derivar requisitos detalhados e iniciar o desenvolvimento com uma visão clara dos principais objetivos a serem alcançados pelo sistema.

Esquema 2 - Diagrama de Caso De Uso



Fonte:autor

- Diagrama de classe

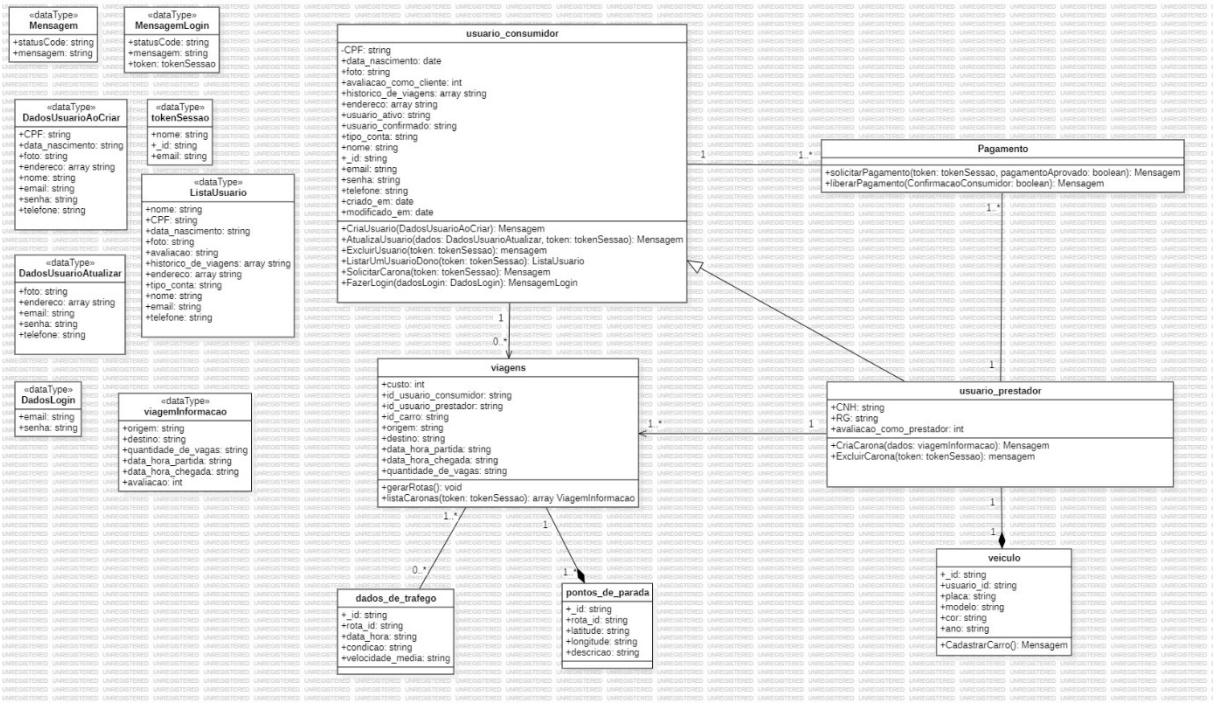
O diagrama de classe que segue é uma representação fundamental da estrutura estática do sistema, mostrando as classes que compõem o sistema, seus atributos, métodos e as relações entre elas. Este diagrama serve como uma blueprint essencial para o desenvolvimento, oferecendo uma visão detalhada da arquitetura interna do sistema.

Cada classe no diagrama representa um objeto ou conceito chave dentro do sistema, encapsulando tanto os dados (atributos) quanto as operações (métodos) que podem ser realizadas sobre esses dados. As relações entre as classes, como

associações, heranças e dependências, são também claramente delineadas, permitindo uma compreensão completa de como as diferentes partes do sistema interagem entre si.

O diagrama de classe é especialmente importante nas fases de design e implementação, pois fornece a base para a codificação, ajudando a garantir que o sistema seja bem estruturado e modular. Ele também facilita a manutenção e a extensão do sistema no futuro, ao proporcionar uma visão clara das responsabilidades de cada classe e das interações entre elas.

Esquema 3 – Diagrama de Classe



Fonte:autor

- Diagrame de sequência

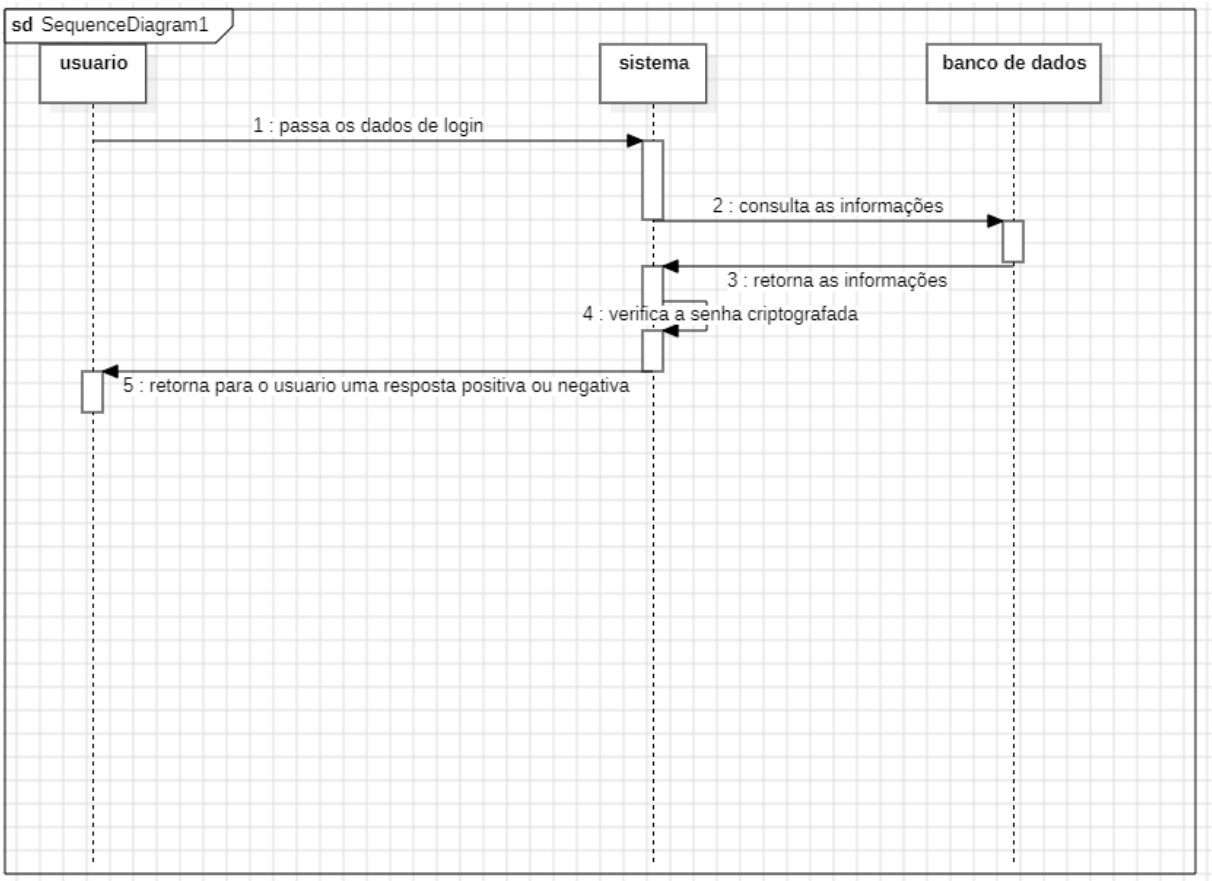
O diagrama de sequência apresentado a seguir ilustra a dinâmica das interações entre os diferentes componentes do sistema ao longo do tempo. Ele descreve, de forma detalhada, a ordem em que as mensagens são trocadas entre os objetos ou componentes do sistema para realizar uma funcionalidade específica.

Cada diagrama de sequência fornece uma visão clara de como os processos se desenrolam em tempo real, mostrando a sequência de eventos necessários para

executar um caso de uso ou uma operação específica. As mensagens trocadas, representadas por setas, indicam a direção e a ordem das comunicações, enquanto as linhas de vida dos objetos mostram o período durante o qual os objetos estão ativos e interagem.

Esse diagrama é crucial para o entendimento das interações temporais dentro do sistema, permitindo que a equipe de desenvolvimento visualize o fluxo de controle e identifique potenciais problemas, como interações complexas ou dependências excessivas. Ele serve também como um guia detalhado para a implementação das funcionalidades, assegurando que o comportamento do sistema esteja em conformidade com os requisitos definidos.

Esquema 4 - Diagrama de Sequência



Fonte:autor

- Diagrama de atividade

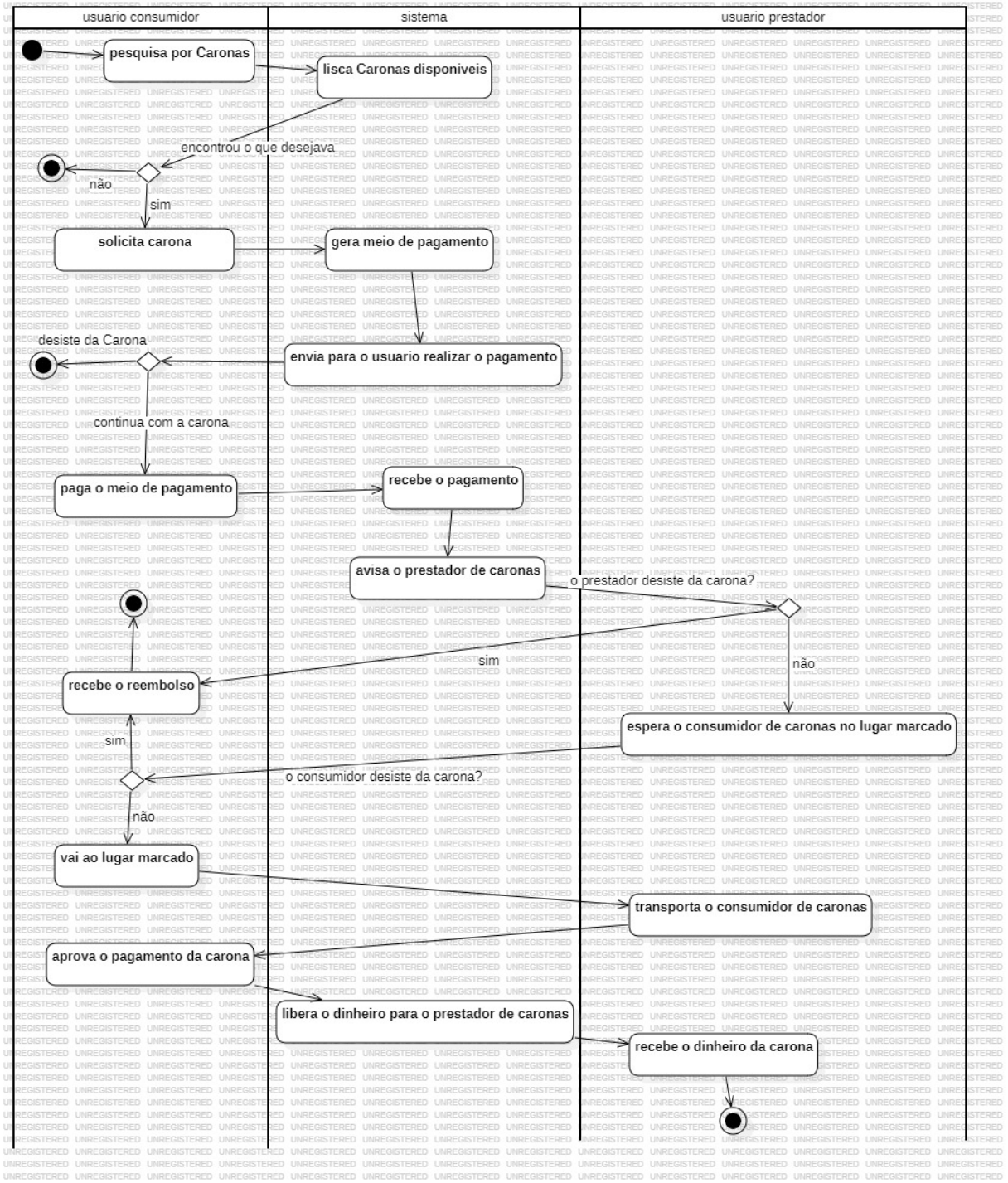
O diagrama de atividade que segue proporciona uma visão detalhada do fluxo de trabalho ou dos processos dentro do sistema, descrevendo como as atividades se sucedem e se interrelacionam para alcançar um determinado objetivo. Este diagrama é uma ferramenta essencial para modelar o comportamento dinâmico do sistema, especialmente em termos de fluxos de controle e paralelismo.

Cada atividade representada no diagrama corresponde a uma etapa específica dentro de um processo maior, e as transições entre essas atividades indicam a progressão lógica de uma tarefa ou operação. O diagrama também pode incluir decisões, paralelismos e sincronizações, permitindo uma compreensão mais rica e completa dos processos que o sistema deve executar.

O diagrama de atividade é particularmente útil para mapear processos complexos que envolvem múltiplos estados ou fluxos de trabalho paralelos. Ele auxilia

a equipe de desenvolvimento a identificar gargalos, redundâncias ou dependências que possam impactar o desempenho ou a eficiência do sistema. Além disso, serve como uma referência valiosa durante a implementação, garantindo que o fluxo de atividades no sistema esteja alinhado com os requisitos funcionais estabelecidos.

Esquema 5 – Diagrama de Atividade

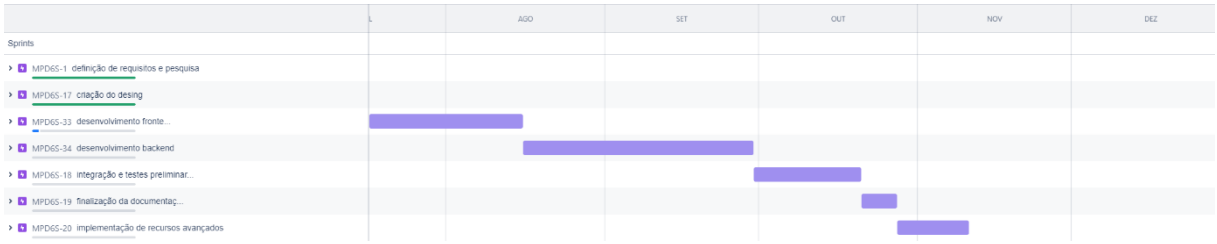


Fonte:autor

5.4. Cronograma do projeto

Para ilustrar as etapas do projeto, foram capturadas duas imagens da plataforma Jira, que é utilizada para o gerenciamento do projeto. A primeira imagem apresenta uma visão geral concisa do planejamento ao longo do tempo, destacando os marcos e cronogramas principais. A segunda imagem oferece uma visão mais detalhada e abrangente, mostrando todas as etapas e tarefas específicas do projeto com maior precisão. Essas imagens visam fornecer uma compreensão clara tanto do progresso geral quanto dos detalhes operacionais do gerenciamento do projeto.

Esquema 6 – cronograma visão condensada



Fonte:autor

Esquema 7 – cronograma visão extendida

Fonte: autor

6. 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.

Referências

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

CETESB. *Qualidade do ar no Estado de São Paulo 2022*. São Paulo: CETESB, 2022. Disponível em: https://cetesb.sp.gov.br/ar/. Acesso em: 11 out. 2024.

Dapper, Spohr,Zanini. Poluição do ar como fator de risco para a saúde: uma revisão sistemática no estado de São Paulo.Disponível em: https://www.scielo.br/j/ea/a/3bgQL4DTXtpQFnr7nYRQMJz/. Acesso em: 11 out 2024.

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

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

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

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

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

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

IBGE. São Paulo: Panorama. *Instituto Brasileiro de Geografia e Estatística*.

Disponível em: https://www.ibge.gov.br/cidades-e-estados/sp/sao-paulo.html. Acesso em: 11 out. 2024.

IEMA. CIDADE de São Paulo tem poluição do ar acima do recomendado pela OMS nos últimos 22 anos. 26 maio 2022. Disponível em: https://energiaeambiente.org.br/cidade-de-sao-paulo-tem-poluicao-do-ar-acima-do-recomendado-pela-oms-nos-ultimos-22-anos-20220526. Acesso em: 11 out. 2024.

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

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

McKINSEY & COMPANY. *Solutions for smart mobility in urban areas*. Disponível em: https://www.mckinsey.com. Acesso em: 11 out. 2024.

MOBILIZE. *Estudo Mobilize 2022 - Mobilidade Urbana em Dados e nas Ruas*. São Paulo, 2022. Disponível em: https://www.mobilize.org.br/estudos/mobilize-2022. Acesso em: 11 out. 2024.

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

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

PUB DEV. *Flutter and Dart packages.* Disponível em: https://pub.dev. Acesso em: 11 out. 2024.

Silva, Dall'Alba, Delduque. Mobilidade urbana e determinação social da

saúde: uma reflexão. Disponível em: https://www.scielo.br/j/sausoc/a/QkdqBnJ5dwfKY3BhwKmhw4b/. Acesso em: 11 out. 2024.

VASCONCELLOS, Eduardo A. A crise da mobilidade urbana em São Paulo. Disponível em: https://www.scielo.br/j/spp/a/Ld57ZY865v3jsmXDTPd3BVG/. Acesso em: 11 out. 2024.

WORLD BANK. *Smart Mobility Program For São Paulo*. Disponível em: https://www.worldbank.org/en/news/feature/2022/05/01/changing-commuters-choices-helps-sao-paulo-reduce-traffic-congestion. Acesso em: 11 out. 2024.