

**FACULDADE DE TECNOLOGIA DO IPIRANGA**

**CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

Felipe Ribeiro de Souza

João Lucas Miclos de Negreiros

Lettícia de Sousa Tonon

Lucas Kaue Guimarães Mota

**Nossa Via:**

**Aplicativo Mobile para reclamação sobre problemas nas vias públicas.**

SÃO PAULO

2024

  
**FACULDADE DE TECNOLOGIA DO IPIRANGA**

**CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

Felipe Ribeiro de Souza

João Lucas Miclos de Negreiros

Lettícia de Sousa Tonon

Lucas Kaue Guimarães Mota

**Nossa Via:**

**Aplicativo Mobile para reclamação sobre problemas nas vias públicas.**

Trabalho de Conclusão de Curso apresentado à FATEC Ipiranga, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadora: Prof. (a) Dr. (a) Ana Paula Gonçalves Serra

SÃO PAULO

2024

**RESUMO**

Parte obrigatória do Trabalho de Conclusão de Curso, o resumo deve ser do tipo informativo ou analítico. O Resumo deve ser redigido na terceira pessoa do singular com verbo na voz ativa, em parágrafo único, e conter no mínimo 150 e no máximo 500 palavras. Não devem aparecer abreviações, nem símbolos, nem citações. Em seguida, colocam-se as “palavras-chave”, descritivas do tema da pesquisa. As palavras-chave devem aparecer em ordem alfabética, iniciadas com letra maiúsculas e separadas por ponto entre si. Devem ter no mínimo três e no máximo seis palavras-chave. O resumo sempre deve ser redigido após o término do relatório. No trabalho de graduação da Faculdade de Tecnologia do Ipiranga deve-se colocar também um resumo em língua estrangeira, mais comumente em inglês o “Abstract”, seguido das “*Keywords*”. **No Resumo devem ser apresentados a contextualização do tema, o objetivo, metodologia e resultados.** Sugere-se que o resumo seja escrito depois do término do trabalho.

Palavras-chave: FATEC. Modelo. Normas. Resumo. Trabalho de Graduação.

**ABSTRACT**

Tradução do resumo.

Keywords:

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Exemplo de conversão Python para SQL 14](#_Toc182517782)

[Figura 2 - Exemplo de token JWT 14](#_Toc182517783)

[Figura 3 - Diagrama de caso de uso 14](#_Toc182517784)

[Figura 4 - Diagrama de classes 14](#_Toc182517785)

[Figura 5 - Diagrama de implantação 14](#_Toc182517786)

[Figura 6 - Diagrama de classes do caso de uso criar reclamação 14](#_Toc182517787)

[Figura 7 - Diagrama de sequência criar reclamação. – Fluxo Principal CSU001 14](#_Toc182517788)

[Figura 8 - Diagrama de sequência criar reclamação – Fluxo Alternativo 1 CSU001 14](#_Toc182517789)

[Figura 9 - Diagrama de sequência criar reclamação – Fluxo Alternativo 2 CSU001 Fonte: autores (2024). 14](#_Toc182517790)

[Figura 10 - Diagrama de classes do caso de uso comentar reclamação 15](#_Toc182517791)

[Figura 11 - Diagrama de sequência comentar reclamação – Fluxo Principal CSU011 Fonte: autores (2024). 16](#_Toc182517792)

[Figura 12 - Diagrama de classes do caso de uso curtir reclamação 17](#_Toc182517793)

[Figura 13 - Diagrama de sequência curtir reclamação – Fluxo Principal CSU010 Fonte: autores (2024). 18](#_Toc182517794)

[Figura 14 - Diagrama de classes do caso de uso alterar status 19](#_Toc182517795)

[Figura 15 - Diagrama de sequência alterar status reclamação – Fluxo Principal CSU006 Fonte: autores (2024). 20](#_Toc182517796)

[Figura 16 - Diagrama de classes do caso de uso cadastrar usuário 21](#_Toc182517797)

[Figura 17 – Diagrama de sequência cadastrar usuário – Fluxo Principal CSU002 22](#_Toc182517798)

[Figura 18 - Diagrama de atividades 23](#_Toc182517799)

[Figura 19 - Diagrama de estados da classe reclamação 24](#_Toc182517800)

**LISTA E ABREVIAURAS E SIGLAS**

*API - Application Programming Interface*

APK - *Android Application Package*

Blob - *Binary Large Object*

CEP - Código de Endereçamento Postal

EAS - *Expo Application Service*

HTTP - *Hypertext Transfer Protocol*

IaaS – *Infrastructure as a Service*

IPA - iOS Package App Store

JWT - *JSON Web Token*

MVC - *Model-View-Controller*

DAO - *Data Access Object*

MVCC - Controle de Concorrência Multiversão

*ORM - Object Relational Mapping*

PaaS – *Platform as a Service*

REST - *Representational State Transfer*

SGBDRO - Sistema de Gerenciamento de Banco de Dados Objeto-Relacional

SQL - *Structured Query Language*

UML - *Unified Modeling Language*

**Sumário**

[1 INTRODUÇÃO 12](#_Toc182518523)

[1.1 Objetivos 13](#_Toc182518524)

[1.1.1 Objetivos Específicos 13](#_Toc182518525)

[1.2 Justificativa 14](#_Toc182518526)

[1.3 Procedimentos metodológicos e tecnologias 14](#_Toc182518527)

[2 TECNOLOGIAS, FERRAMENTAS E PADRÕES EMPREGADOS NO DESENVOLVIMENTO DE SOFTWARE 14](#_Toc182518528)

[2.1 Framework Scrum 14](#_Toc182518529)

[2.1.1 Manifesto Ágil 14](#_Toc182518530)

[2.1.2 Pilares do Scrum 14](#_Toc182518531)

[2.1.3 Componentes do time Scrum 14](#_Toc182518532)

[2.1.4 Sprint 14](#_Toc182518533)

[2.1.4.1 Product Backlog 14](#_Toc182518534)

[2.2 Diagramação e Prototipação 14](#_Toc182518535)

[2.2.1 Draw.io 14](#_Toc182518536)

[2.2.2 Figma 14](#_Toc182518537)

[2.3 Banco de dados 14](#_Toc182518538)

[2.3.1 Banco de dados relacional 14](#_Toc182518539)

[2.3.2 PostgreSQL 14](#_Toc182518540)

[2.3.3 Aiven 14](#_Toc182518541)

[2.4 Tecnologias e padrões utilizados no desenvolvimento da API 14](#_Toc182518542)

[2.4.1 Padrão REST 14](#_Toc182518543)

[2.4.2 MVC e DAO 14](#_Toc182518544)

[2.4.3 Python 14](#_Toc182518545)

[2.4.3.1 Micro-Framework Flask 14](#_Toc182518546)

[2.4.3.2 SQLAlchemy 14](#_Toc182518547)

[2.4.4 APIs Externas 14](#_Toc182518548)

[2.4.4.1 ViaCEP 14](#_Toc182518549)

[2.4.4.2 Google Maps e API Geocoding 14](#_Toc182518550)

[2.4.5 JWT 14](#_Toc182518551)

[2.4.6 Docker 14](#_Toc182518552)

[2.5 Testes de software 14](#_Toc182518553)

[2.6 Tecnologias utilizadas no desenvolvimento do sistema mobile 14](#_Toc182518554)

[2.6.1 Android 14](#_Toc182518555)

[2.6.2 JavaScript 14](#_Toc182518556)

[2.6.3 React Native 14](#_Toc182518557)

[2.6.4 Expo 14](#_Toc182518558)

[2.7 Arquitetura cliente-servidor 14](#_Toc182518559)

[2.7.1 Nuvem 14](#_Toc182518560)

[2.7.2 Azure 14](#_Toc182518561)

[2.7.2.1 Container Apps 14](#_Toc182518562)

[2.7.2.2 Blob Storage 14](#_Toc182518563)

[2.8 Integração e Implantação 14](#_Toc182518564)

[2.8.1 Controle de Versionamento 14](#_Toc182518565)

[2.8.2 GitHub 14](#_Toc182518566)

[2.8.3 GitHub Actions 14](#_Toc182518567)

[3 REQUISITOS DO SISTEMA DE SOFTWARE 14](#_Toc182518568)

[3.1 Requisitos Funcionais 14](#_Toc182518569)

[3.2 Regras de Negócio 14](#_Toc182518570)

[3.3 Modelagem Funcional 14](#_Toc182518571)

[3.4 Requisitos Não-Funcionais 14](#_Toc182518572)

[4 ANÁLISE 14](#_Toc182518573)

[4.1 Diagrama de Classes de Análise (Visão de Negócio) 14](#_Toc182518574)

[5 PROJETO 14](#_Toc182518575)

[5.1 Arquitetura do Sistema 14](#_Toc182518576)

[5.2 Diagrama de Classes de Projeto por Caso de Uso 14](#_Toc182518577)

[5.2.1 Diagrama de classes do caso de uso criar reclamação 14](#_Toc182518578)

[5.2.2 Diagrama de classes do caso de uso comentar reclamação 14](#_Toc182518579)

[5.2.3 Diagrama de classes do caso de uso curtir reclamação 16](#_Toc182518580)

[5.2.4 Diagrama de classes do caso de uso alterar status 18](#_Toc182518581)

[5.2.5 Diagrama de classes do caso de uso cadastrar usuário 20](#_Toc182518582)

[5.3 Diagrama de atividades 22](#_Toc182518583)

[5.4 Diagrama de estados 24](#_Toc182518584)

[6 CONSIDERAÇÕES FINAIS 25](#_Toc182518585)

[7 REFERÊNCIAS 25](#_Toc182518586)

[8 APÊNDICE A – INSTRUMENTO DE PESQUISA 26](#_Toc182518587)

[1. Levantamento dos Requisitos do Sistema de Software 26](#_Toc182518588)

[8.1 1.1. Extração de Requisitos 26](#_Toc182518589)

# INTRODUÇÃO

As ruas e estradas se tornaram uma parte fundamental da nossa sociedade, elas são a principal forma de locomoção nas grandes cidades, utilizadas todos os dias por inúmeros brasileiros, seja através de carros, ônibus, motos ou até mesmo a pé, elas conectam toda a infraestrutura urbana, ajudando no transporte de pessoas e bens por todo o país. Embora tragam diversos benefícios, elas não estão isentas de defeitos, o despejo de lixo indevido, alagamentos, falta de manutenção e quedas de arvores são apenas algumas dificuldades que o povo enfrenta durante seu dia a dia. Algumas irregularidades, como falta de sinalização, buracos e desníveis no asfalto, podem contribuir para causa acidentes se não passarem pela devida manutenção “o número de pontos críticos identificados ao longo dos 108.863 quilômetros pesquisados aumentou 75,6%. Passou de 454 em 2018 para 797 em 2019” (CNT, 2019).

O desgaste tem um grande papel nesses problemas, como pode ser visto nos danos constantes em ruas e avenidas. Além da degradação, existem diversos outros fatores que dificultam o uso das vias públicas, muitas vezes a má gestão torna o estado das vias ainda mais precário, principalmente nas regiões mais pobres.

Os buracos no asfalto são o problema mais lembrado por moradores da cidade de São Paulo, segundo pesquisa Datafolha divulgada neste sábado (16). O transtorno é o campeão de reclamações em todas as regiões da cidade, com exceção do Centro, onde a presença de usuários de droga e de moradores de rua foi mais citada (G1, 2024).

Ainda que esses problemas se espalhem através do “boca-a-boca”, dificilmente ganham espaço nos canais de comunicação adequados, o que atrapalha a identificação e solução deles.

Tendo em vista a crescente presença de tecnologias digitais na nossa sociedade, bem como a falta de visibilidade dos problemas em vias públicas, foi proposto a criação de um aplicativo que auxilie na divulgação dessas irregularidades, tendo como principal função a criação de reclamações para relatar esses problemas e auxiliar no aumento de visibilidade dos mesmos.

## Objetivos

No contexto das cidades modernas, a manutenção e o monitoramento das vias públicas são desafios constantes que impactam diretamente a qualidade de vida dos cidadãos. A percepção da população sobre a eficácia da execução dessas tarefas é fundamental para entender suas falhas e oportunidades de melhoria.

Outra questão importante para o projeto é a relação entre como a visibilidade dos problemas pode contribuir para melhorias nessas infraestruturas. Sendo esse o princípio utilizado para melhorar a qualidade das vias públicas.

Para a resolução dos problemas apresentados, o objetivo é aumentar a visibilidade por meio de um aplicativo *mobile*, criando um canal de comunicação que permita a publicação e divulgação de reclamações sobre problemas encontrados nas vias públicas, como buracos, falta de sinalização, lixo e falta de iluminação.

### Objetivos Específicos

Um aplicativo *mobile* desenvolvido utilizando as linguagens de programação Python e JavaScript, o softwae deverá ter uma interface simples e de fácil utilização, assim, os usuários poderão realizar a publicação de reclamações sobre quaisquer problemas que encontrarem em vias públicas, de forma transparente, rápida e direta.

Com o intuito de desenvolver uma comunidade ativa e engajada, o aplicativo deverá permitir que os usuários comentem e curtam as publicações, além de possuir um sistema de pontuação onde os usuários mais ativos terão seu perfil destacado em relação aos demais.

O aplicativo possui um *feed* exibindo as reclamações mais próximas, relevantes (maior número de curtidas) e recentes, bem como seu status de resolução. Com essas informações facilmente disponíveis, a população e as autoridades responsáveis têm mais facilidade em identificar as irregularidades e realizar a manutenção adequada.

## Justificativa

Nosso país enfrenta constantes problemas em suas vias de mobilidade, sejam esses, buracos, falta de sinalização e iluminação, descarte indevido de lixo e entulho, alagamentos, e muitos outros. Diversas dessas irregularidades diminuem a qualidade de vida das pessoas, principalmente em comunidades e regiões de baixa renda (Garcia, 2023).

Também deve-se levar em conta a necessidade de divulgar essas adversidades, por meio da sociedade, a fim de chamar a atenção das autoridades competentes, para que possam corrigir esses problemas.

Considerando essas questões, nota-se a necessidade de uma plataforma centralizada que viabilize a publicação e visualização de irregularidades. Dessa forma, foi idealizado um aplicativo que permite que os cidadãos criem e acompanhem as reclamações de problemas em vias públicas, além de incentivar a interação entre usuários, criando assim um ambiente dinâmico e interativo.

## Procedimentos metodológicos e tecnologias

A arquitetura do projeto seguirá o modelo MVC (*Model-View-Controller*) junto ao padrão DAO (*Data Access Object*). O banco de dados escolhido foi PostgreSQL, já as linguagens de programação utilizadas foram Python para o *back-end* e JavaScriptcomReact Nativepara o *front-end*.

Para metodologia de desenvolvimento foi adotado o *framework* Scrum baseado no método ágil, organizando as entregas do projeto em *sprints*, já para o versionamento foi utilizado a ferramenta GIT em conjunto a plataforma GitHub, enquanto no processo de integração e *deploy*, foi utilizado do expo e GitHub *Actions* para o *front-end* e *back-end* respectivamente.

# TECNOLOGIAS, FERRAMENTAS E PADRÕES EMPREGADOS NO DESENVOLVIMENTO DE SOFTWARE

## Framework Scrum

A metodologia adotada para o desenvolvimento do aplicativo foi o *framework* Scrum, que é uma estrutura que segue os princípios do Manifesto Ágil e é amplamente utilizada para gerenciar projetos de maneira eficiente.

No Guia do Scrum, é definido que:

Scrum emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco. Scrum envolve grupos de pessoas que, coletivamente, possuem todas as habilidades e conhecimentos necessários para fazer o trabalho e compartilhar ou adquirir essas habilidades conforme necessário (Schwaber; Sutherland, 2020, p. 4).

Em outras palavras, o Scrum facilita a entrega contínua e incremental de produtos, permitindo que as equipes se adaptem rapidamente às mudanças e aprimorem seus processos continuamente.

### Manifesto Ágil

O Manifesto para Desenvolvimento Ágil de Software, foi criado em 2001 por um grupo de desenvolvedores de software e estabelece os valores e princípios fundamentais que norteiam as metodologias ágeis, incluindo o Scrum. O manifesto ágil valoriza:

Indivíduos e interações mais que processos e ferramentas. Software em funcionamento mais que documentação abrangente. Colaboração com o cliente mais que negociação de contratos. Responder a mudanças mais que seguir um plano (Manifesto Ágil, 2001).

Esses valores são complementados pelos seguintes princípios definidos pelo Manifesto Ágil (2001):

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

### Pilares do Scrum

O Scrum é sustentado por três pilares, que são essenciais para o funcionamento eficaz do *framework*, esses pilares são citados por Schwaber e Sutherland (2020, p. 4), “os pilares empíricos do Scrum: transparência, inspeção e adaptação”.

* **Transparência:** Todos os aspectos do processo devem estar visíveis para os envolvidos, facilitando a compreensão e o acompanhamento dos progressos. Processos com baixa transparência podem levar a decisões que diminuem o valor e aumentam o risco. O uso de artefatos como o backlog e reuniões diárias garante a visibilidade do trabalho.
* **Inspeção:** O progresso em direção aos objetivos é inspecionado regularmente. Eventos como as revisões de sprint permitem que a equipe e o cliente avaliem os entregáveis e identifiquem possíveis problemas.
* **Adaptação:** Baseado nas inspeções, ajustes são feitos no processo ou no produto. Isso permite que o time se adapte rapidamente às mudanças nas necessidades do projeto.

Esses pilares trabalham juntos, a transparência permite a inspeção, a inspeção habilita a adaptação, e a adaptação garante que o projeto permaneça eficiente e flexível diante de mudanças.

### Componentes do time Scrum

O time Scrum é composto por três papéis principais:

* **Scrum Master:** Atua como facilitador, removendo obstáculos e garantindo que o time siga os princípios e as práticas do Scrum. Ele também assegura que todos entendam o processo e a sua importância para o sucesso do projeto.
* **Time de Desenvolvimento:** Consiste em profissionais que possuem as habilidades necessárias para entregar incrementos de produto em cada sprint. O time é autogerenciado e multidisciplinar, garantindo que todas as competências necessárias estejam presentes
* **Product Owner:** Representa os interesses do cliente e é responsável por maximizar o valor do produto desenvolvido. Ele também é responsável por gerenciar o Product Backlog, garantindo que o time tenha uma visão clara do que deve ser feito.

### Sprint

O Sprint é o coração do Scrum. Trata-se de um ciclo de trabalho com duração fixa, segundo Pereira, Torreão e Marçal (2007, p. 4): “A recomendação é de que a duração da Sprint fique entre 2 a 4 semanas.”, nesse período, o time busca desenvolver incrementos de produto prontos para serem entregues. Cada Sprint começa com uma Sprint Planning, onde o time seleciona os itens do Product Backlog que serão trabalhados durante aquele ciclo.

No final da Sprint, uma revisão é realizada para inspecionar o trabalho concluído e se necessário ajustar as prioridades para as Sprints futuras. As Sprints criam uma cadência previsível, oferecendo ao cliente e ao time a possibilidade de avaliar continuamente o progresso (Schwaber; Sutherland, 2020, p. 11).

Para o desenvolvimento deste trabalho, foi definido 5 Sprints com duração de 21 dias cada.

Tabela 1 – Cronograma Entrega Sprints

|  |  |
| --- | --- |
| Data de Entrega | Sprint |
| 28/08/2024 | Sprint 1 |
| 18/09/2024 | Sprint 2 |
| 09/10/2024 | Sprint 3 |
| 30/10/2024 | Sprint 4 |
| 20/11/2024 | Sprint 5 |

Fonte: Autores (2024).

#### Product Backlog

O Product Backlog é essencial para o Scrum, pois permite que a equipe de desenvolvimento saiba exatamente o que precisa ser feito e em qual ordem. De acordo com Sabbagh (2013, p. 111), “O Product Backlog é uma lista de tudo o que se acredita que será desenvolvido pelo Time de Desenvolvimento no decorrer do projeto”.

Segundo Sabbagh (2013, p. 112), “O Scrum não prescreve nenhum formato ou padrão para o Product Backlog. O importante é que ele tenha o formato de itens em uma sequência”. Cada item no Product Backlog deve estar claro e detalhado, de maneira que facilite o entendimento e a execução das tarefas, além disso o Product Backlog é projetado para ser dinâmico e flexível, ele pode e deve ser alterado conforme novas informações surgem e as prioridades mudam.

As figuras a seguir irão mostrar o Product Backlog desenvolvido para o software Nossa Via

Figuraaaaaaaaaaas

## Diagramação e Prototipação

Durante o processo de desenvolvimento de software, muitas vezes uma das maiores dificuldades dos desenvolvedores não é a programação ou implantação do projeto em si, mas sim compreender as características e escopo.

Segundo Sommerville (2012), projetos que não são bem documentados, ou que não têm seus requisitos adequadamente definidos, enfrentam grandes problemas com incertezas ou mudanças constantes.

Devido a esses problemas, muitas metodologias foram criadas a fim de ajudar a representar graficamente um software, tanto para os desenvolvedores, quanto para os clientes. Exemplos disso podem ser vistos nas técnicas de diagramação e prototipação.

A diagramação é uma forma de documentação que envolve especificar as características de um sistema de forma gráfica, ela ajuda a representar a arquitetura, componentes e relações do sistema, servindo como meio para comunicar ideias complexas para os membros da equipe e stakeholders.

Segundo Lima (2011, p. 27):

[...] na Tecnologia da Informação, a construção de modelos exige uma linguagem de modelagem que inclua elementos visuais para expressar conceitos e uma notação simples, mas poderosa para esses elementos.

Por isso, escolhemos utilizar a UML (*Unified Modeling Language*) para o desenvolvimento do projeto Nossa Via. Além de ter uma ótima clareza visual e versatilidade, seu padrão é reconhecido globalmente e tem suporte em diversas ferramentas. Utilizamos a ferramenta draw.io para a criação dos seguintes diagramas da UML: caso de uso, classe, sequência, atividade, implantação e estado.

Em relação a prototipação, Engholm (2010) destaca que, ao criar aplicações, a implementação de protótipos de interfaces é crucial para garantir bons resultados na usabilidade. Esses protótipos permitem uma visualização imediata do sistema pelos participantes do projeto, facilitando a validação e o refinamento do produto em desenvolvimento. Eles também garantem que tanto os desenvolvedores, quanto os clientes possam visualizar como será o produto antes que esteja completamente desenvolvido, auxiliando na identificação de erros de usabilidade e design logo no início do desenvolvimento, assim, mudanças e correções podem ser feitas previamente sem um grande custo.

Após uma análise detalhada dos diferentes tipos de protótipos existentes, optamos por utilizar protótipos de média fidelidade no desenvolvimento do projeto Nossa Via. A escolha se deve à necessidade de garantir uma representação precisa e detalhada da interface do usuário, proporcionando uma visualização clara do produto final. Para a criação desses protótipos, utilizamos a ferramenta Figma.

### Draw.io

Para o desenvolvimento dos diagramas do projeto Nossa Via, utilizamos o Draw.io, uma ferramenta web de diagramação que permite criar uma grande variedade de diagramas, destacando-se tanto por sua facilidade de usabilidade quanto pela gratuidade de utilização.

A ferramenta também inclui opções de integração com diversas plataformas de armazenamento (Draw.io, ano), uma função que utilizamos para guardar os diagramas no repositório do projeto no GitHub.

### Figma

Figma é uma ferramenta web especializada na criação, compartilhamento e teste de designs de software, destacando-se pelo seu alto grau de detalhamento e personalização, isso fez com que se tornasse amplamente utilizado na criação de protótipos para telas dos mais variados softwares.

Possui vantagens como visualização e criação com múltiplos usuários em tempo real, além da função de feedback integrado, que permite adicionar avaliações diretamente no design (Figma, ano).

## Banco de dados

Em um projeto de software, a escolha do banco de dados é uma decisão crítica, pois afeta diretamente a estrutura e a eficiência do sistema. Segundo Date (2004, p. 3), o banco de dados é um repositório central onde uma coleção de dados computadorizados podem ser armazenados e organizados, esses dados podem ser acessados pelos usuários por meio de solicitações na aplicação.

No desenvolvimento deste projeto, foi escolhido o modelo de banco de dados relacional que pudesse garantir a integridade dos dados e a eficiência nas consultas.

### Banco de dados relacional

Um banco de dados relacional funciona armazenando dados em tabelas, onde cada tabela representa uma entidade ou objeto específico. As colunas de uma tabela representam atributos ou características, como “Nome”, “Data”, “Quantidade”, etc. Cada coluna possui um tipo de dado específico, como “INTEGER” para valores numéricos inteiros ou “CHAR” para valores de texto. As linhas de uma tabela representam os registros únicos de uma entidade dentro desta tabela.

A principal qualidade que diferencia os bancos de dados relacionais dos demais são os “relacionamentos” entre tabelas. Esses relacionamentos permitem que diferentes tabelas “conversem” entre si. Por exemplo, em um banco de dados de uma loja virtual, existem as tabelas “Cliente”, que armazena os dados pessoais dos clientes, e “Pedido”, que armazena os dados dos pedidos específicos. Essas tabelas têm um relacionamento que permite identificar qual cliente realizou quais pedidos.

Para criar relacionamentos, toda tabela precisa de uma “chave primária”, que é uma coluna preenchida com um dado único, servindo como identidade para cada registro. Existem também as “chaves estrangeiras”, que são colunas que referenciam a chave primária de outra tabela, estabelecendo a conexão entre as duas tabelas.

Existem três tipos de relacionamentos:

* **Um-para-um (1:1):** Cada registro em uma tabela está associado a um único registro em outra. Útil para separar dados raramente usados ou confidenciais, como dados pessoais de um funcionário.
* **Um-para-muitos (1:N):** Um registro em uma tabela está relacionado a vários registros em outra. Exemplo: um cliente pode fazer vários pedidos, mas cada pedido pertence a um único cliente.
* **Muitos-para-muitos (N:N):** Vários registros de uma tabela se relacionam com vários registros de outra. Exige uma tabela intermediária. Exemplo: alunos podem estar em várias disciplinas e cada disciplina pode ter vários alunos.

Esses conceitos podem ser exemplificados na seguinte imagem:

Figura banco de dados própria

### PostgreSQL

PostgreSQL é um Sistema de Gerenciamento de Banco de Dados Objeto-Relacional (SGBDRO) de código aberto com mais de 35 anos de desenvolvimento ativo, o que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho (POSTGRESQL, 2024).

Dentre suas diversas vantagens, o PostgreSQL conta com um sistema de Controle de Concorrência Multiversão (MVCC), em que cada requisição de leitura ou escrita acessa uma versão dos dados existente no momento em que foi iniciada, isso isola as alterações feitas por outras requisições, permitindo que múltiplas operações de leitura e escrita ocorram simultaneamente, sem interferência mútua.

Essas características tornam o PostgreSQL altamente adequado para o desenvolvimento do projeto Nossa Via. Os principais fatores que determinaram sua escolha foram a gratuidade do SGBDRO e a extensão PostGIS, um módulo extremamente útil para manipulação de dados geoespaciais. O PostGIS possibilita operações avançadas em dados de localização, como cálculos de distâncias, áreas e interseções, simplificando processos fundamentais do projeto.

### Aiven

Aiven é uma plataforma que tem como objetivo facilitar a configuração, monitoramento e gerenciamento de infraestrutura de serviços de banco de dados em diversas nuvens, como a Microsoft Azure, AWS e Google Cloud.

Considerando que os serviços da Aiven são gratuitos até um certo limite de capacidade e tornam o gerenciamento de infraestrutura mais simples, essa solução se mostrou ideal para ser aplicada durante o desenvolvimento do projeto Nossa Via.

## Tecnologias e padrões utilizados no desenvolvimento da API

Para o desenvolvimento do aplicativo, foram utilizadas APIs (*Application Programming Interface*), que funcionam como uma interface para a comunicação entre aplicações. Conforme definido pela Amazon (2023), "APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos."

Essa tecnologia foi fundamental para o desenvolvimento da solução de software, atuando no o *back-end* da aplicação. O *back-end* é responsável pela interação com o banco de dados, servidores e pela execução de toda a lógica que está por trás da interface visível ao usuário final.

### Padrão REST

O modelo de API adotado para a aplicação foi o padrão REST, amplamente utilizado devido à sua flexibilidade e popularidade. De acordo com a Amazon (2023), "Essas são as APIs mais populares e flexíveis encontradas na Web atualmente. O cliente envia solicitações ao servidor na forma de dados, o servidor processa essas solicitações e retorna os dados de saída ao cliente."

O termo REST vem do inglês *Representational State Transfer* e pode ser traduzido como "Transferência de Estado Representacional", e uma de suas principais características é a ausência de estado (*stateless*). Isso significa que o servidor não armazena dados de sessão do cliente entre requisições. A comunicação entre cliente e servidor segue o padrão HTTP (*Hypertext Transfer Protocol*), o que simplifica a troca de dados e permite uma integração ágil e eficiente com outras plataformas.

### MVC e DAO

Para a arquitetura de desenvolvimento do sistema, optou-se pelo modelo MVC (*Model View Controller*) em conjunto com o padrão DAO (*Data Access Object*). Essa escolha visa garantir a segurança e a manutenibilidade do sistema. A utilização do padrão MVC proporciona um aumento significativo na reutilização e manutenção do projeto, uma vez que separa as camadas de apresentação, lógica de negócio e gerenciamento de fluxo, conforme discutido por Luciano e Alves (2011).

Nesse modelo, temos a camada de apresentação, a camada de negócios e a camada de dados. Cada uma dessas camadas possui suas responsabilidades específicas, o que ajuda a manter o sistema modular e facilita a manutenção e evolução da aplicação (AWARI, 2023).

Complementarmente, o padrão DAO permite o isolamento dos dados, possibilitando alterações na estrutura do projeto de forma mais ágil e eficiente.

Assim, o padrão de projeto DAO pode ser aplicado nas aplicações que façam acesso a qualquer tipo de banco de dados (relacional, orientado à objetos, arquivos de texto pleno, arquivos XML, servidores remotos, etc.), e que possa ter que acessar outros tipos quaisquer durante ou após o desenvolvimento da aplicação (Trindade; Fischer, 2015).

Essa flexibilidade é essencial, pois permite a adaptação a diferentes tipos de dados, garantindo que o sistema possa evoluir conforme as necessidades.

A combinação dessas arquiteturas resulta em um software que apresenta flexibilidade tanto nas perspectivas de negócios quanto no sistema em si. Isso significa que alterações podem ser implementadas de maneira individualizada, sem impactar outras partes do sistema. Essa característica é especialmente importante quando se considera o ciclo de vida do *software*.

### Python

Para o desenvolvimento do *back-end* do aplicativo, foi utilizada a linguagem Python para a criação das APIs , “Embora simples, é também uma linguagem poderosa, podendo ser usada para administrar sistemas e desenvolver grandes projetos. É uma linguagem clara e objetiva, pois vai direto ao ponto, sem rodeios (MENEZES, 2011)”. O Python é uma linguagem de programação orientada a objetos, funcional, de tipagem dinâmica e forte, sendo uma linguagem de alto nível com uma sintaxe básica semelhante ao inglês.

Ao utilizar do Python como linguagem de programação se obtém uma série de vantagens como uso de bibliotecas de programação que podem ser nativas, ou desenvolvidas por terceiros, ao utilizar bibliotecas já estabelecidas, o desenvolvimento se torna mais rápido e prático ao reutilizar códigos ao invés de escrevê-los do zero.   
 Outra vantagem trazida pelo Python são seus *frameworks*

“Um framework Python é uma coleção de pacotes e módulos. Um módulo é um conjunto de códigos relacionados, e um pacote é um conjunto de módulos. Os desenvolvedores podem usar frameworks Python para criar aplicações Python mais rapidamente, porque não precisam se preocupar com detalhes mínimos, a forma como as comunicações acontecem na aplicação Web nem como o Python tornará o programa mais rápido (AWS, 2023)”.

Esse recurso permite utilizar de estruturas preestabelecidas, como no caso do F*ramework Flask* utilizado na aplicação.

#### Micro-Framework Flask

O Flask é um *Micro-Framework*, que possui funcionalidades mínimas já estabelecidas para a criação de uma aplicação web Python , além de vantagens como o suporte da comunidade, documentação acessível, suporte para ferramentas de banco de dados e testes unitários. Essas características permitem um desenvolvimento mais rápido e prático ao utilizar desses padrões.

#### SQLAlchemy

Uma das bibliotecas de maior importância para o desenvolvimento da aplicação é o SQLAlchemy, funcionando como uma interface para a comunicação com banco de dados relacionais, essa biblioteca permite a conversão entre Python e SQL (*Structured Query Language*) através de *Object Relational Mapping* (ORM) para fazer consultas e modificações no banco de dados. Essa biblioteca também possui um controle de seção e transações, sendo assim as alterações feitas durante uma consulta só serão persistidas no banco de dados se não houver nenhum erro, além de possibilitando a reversão da transação.  
 Em conjunto ao SQLAlchemy o Alembic é uma ferramenta de migração de banco de dados, que permite sempre que necessário fazer uma alteração no modelo de dados, como a adição e remoção de tabelas e colunas. Assim quando necessário um novo arquivo de migração pode ser criado automaticamente, permitindo um aumento na liberdade de alterações na estrutura do banco de dados durante o desenvolvimento do projeto.

Figura 1 - Exemplo de conversão Python para SQL

A diagram of a diagram

Description automatically generated

Fonte: Sosache

### APIs Externas

Além das APIs internas desenvolvidas para o *back-end* da aplicação, foram utilizadas algumas APIs externas criadas previamente por terceiros. O uso dessas APIs já estabelecidas para executar funções recorrentes contribuiu para reduzir o tempo de desenvolvimento do projeto, além de possibilitar o acesso a bancos de dados robustos que aumentam a qualidade das informações obtidas pelo aplicativo.

Durante o desenvolvimento, foram integradas duas APIs externas com o objetivo de promover a usabilidade para cidadãos preocupados com as condições das vias públicas: a ViaCEP e a API do Google Maps.

#### ViaCEP

De acordo com os desenvolvedores do ViaCEP, a ferramenta é um “Webservice gratuito de alto desempenho para consulta de Código de Endereçamento Postal (CEP) do Brasil (ViaCEP, 2023).” A API do ViaCEP utiliza uma base de dados robusta, integrada com informações do IBGE e outras plataformas públicas. Quando recebe uma requisição contendo um CEP válido, a API retorna um JSON com informações detalhadas, como 'cep', 'logradouro', 'complemento', 'unidade', 'bairro', 'localidade', 'uf', 'estado', 'região', 'ibge', 'gia', 'ddd' e 'siafi'. Ao integrar essa API ao aplicativo, reduzimos o esforço do usuário para preencher dados de endereço, permitindo que os campos sejam automaticamente completados com as informações fornecidas pelo webservice. Assim, o usuário precisa apenas confirmar as informações obtidas.

#### Google Maps e API Geocoding

Ao utilizar da API fornecida pela Google em integração com o sistema Google Maps, podemos utilizar da interface gráfica já consolidada para a visualização de localização das reclamações, além de permitir que o usuário selecione no mapa a localização da reclamação.   
 Outra função disponibilizada pela Google é a API Geocoding, essa aplicação permite a conversão entre coordenadas geográficas e endereços. A API pode ser usada para converter os dados de endereço fornecidos pelo usuário para as coordenadas de exibição do mapa, assim facilitando a localização das reclamações.

### JWT

O padrão JWT (*JSON Web Token*) é um sistema de verificação normalmente utilizado para autentificação de usuários “Quando o usuário está conectado, cada solicitação subsequente inclui o JWT, que permite que o usuário acesse serviços que são permitidos por esse token (IBM ,2024)”. Utilizando desse sistema, o usuário ao fazer login envia suas informações para validação e ao confirmar seu registro recebe do servidor a autenticação através do JWT que é armazenado localmente em seu dispositivo.  
 Nesse padrão cada requisição feita pelo usuário após o *login* leva consigo o *token* de identificação, permitindo a persistência do *login*, esse JWT possui 3 partes sendo elas o cabeçalho que contém a o algoritmo utilizado, a carga útil onde se encontram as solicitações como a prazo de expiração e por último temos a assinatura do *token* que garante que a mensagem não foi alterada e quem é o emissor do JWT.  
https://www.ibm.com/docs/pt-br/cics-ts/6.x?topic=cics-json-web-token-jwt

Figura 2 - Exemplo de token JWT

A screenshot of a computer

Description automatically generatedFonte: IBM

### Docker

O Docker é uma plataforma aberta que permite empacotar código e suas dependências em imagens, garantindo a portabilidade e a execução consistente da aplicação, independentemente do ambiente físico. Diferente das máquinas virtuais, que isolam todo o sistema operacional e têm seus próprios *kernels*, os contêineres compartilham o *kernel* do sistema operacional do *host*, reduzindo significativamente o uso de recursos de hardware e espaço de armazenamento. Ao utilizar contêineres, as aplicações podem ser executadas de maneira consistente em qualquer máquina física ou servidor, garantindo o mesmo comportamento e desempenho, sem a necessidade de replicar um sistema operacional completo em cada instância. Essa abordagem traz uma série de benefícios,

“Portabilidade, onde todas as dependências para uma aplicação conteinerizada são empacotadas dentro do contêiner, permitindo-a executar em qualquer *host* docker. Previsibilidade, onde o *host* não se importa com o que está sendo executado dentro do contêiner e nem mesmo qual máquina ele está executando, as interfaces são padronizadas e as interações são previsíveis. (JUNIOR; LIMA, 2020)”.

A plataforma Docker também possui o DockerHub um sistema de armazenamento de imagens disponibilizadas pela comunidade, que podem ser utilizadas como base para a criação de novas imagens com as configurações necessárias para o funcionamento da aplicação, assim facilitando a configuração do ambiente.

## Testes de software

No desenvolvimento de um software, é necessário garantir que o produto esteja atendendo todos os requisitos que foram especificados. Com a crescente complexidade dos sistemas, é inevitável que erros aconteçam, por isso é necessário validar todas as funcionalidades, infelizmente, nem sempre é fácil identificar o comportamento do sistema em todos os cenários possíveis, é comum que funções específicas sejam esquecidas, ou que o sistema reaja de maneira inesperada a diferentes situações. A fim de solucionar esse problema, foram criadas múltiplas metodologias para testar as funções e características de um software.

Testar um software traz muitos benefícios, desde melhorar a experiência do usuário, até economizar tempo e dinheiro durante o processo de desenvolvimento. Porém, muitas vezes esses benefícios são intangíveis, sendo difícil prever e comparar como teria sido o desenvolvimento de um projeto com ou sem o uso dos testes. Isso é ainda mais agravado pelo fato de que realizar testes é um processo longo, complexo e caro, como pode ser confirmado pela fala de Sommerville (2011), “aproximadamente 60% dos custos de software são de desenvolvimento, 40% são custos de testes. Para o software customizado, os custos de evolução frequentemente superam os custos de desenvolvimento”.

Em muitos casos empresas possuem prazos e custos apertados, por isso escolhem não testar seus produtos para agilizar a entrega. Mas negligenciar os testes pode facilmente levar a perdas muito maiores, como falhas em ambiente de produção, experiência do usuário insatisfatória, custos de manutenção, danos à reputação e muitos outros.

Isso mostra como é fundamental utilizar estratégias de testes no desenvolvimento de um sistema, no projeto Nossa Via em particular, utilizamos dois métodos para testar o software, o teste de aceite no *front-end*, e testes unitários no *back-end*.

Os testes de aceite são realizados simulando o comportamento do usuário final, nele verificamos se o software está executando corretamente as funções para quais foi construído. O roteiro com os testes realizados no projeto está localizado no apêndice X

Nos testes unitários, os componentes individuais do código, como funções, métodos, ou classes são avaliados individualmente, verificando se funcionam de forma isolada. No caso do projeto, são escritos pelos próprios desenvolvedores, e funcionam de maneira automatizada. O resultado dos testes está localizado no apêndice X

## Tecnologias utilizadas no desenvolvimento do sistema mobile

O desenvolvimento de sistemas mobile exige uma seleção cuidadosa das tecnologias para garantir que o aplicativo seja eficiente, responsivo e fácil de manter. As tecnologias escolhidas para o desenvolvimento deste software incluem o sistema operacional Android, a linguagem de programação JavaScript e o *framework* React Native. A seguir, detalharemos cada uma delas e explicaremos as razões pelas quais foram escolhidas.

### Android

O sistema operacional Android é atualmente uma das plataformas mais populares para dispositivos móveis, oferecendo uma base sólida para o desenvolvimento de aplicativos. Desenvolvido pelo Google, o Android é uma plataforma *open-source* que permite personalizações e modificações amplas, o que o torna ideal para projetos que requerem flexibilidade e controle sobre o hardware e software do dispositivo. Além disso, o Android possui uma vasta comunidade de desenvolvedores e suporte extensivo para uma ampla gama de dispositivos, o que facilita o desenvolvimento e o teste de aplicativos em diversas condições de uso (GOOGLE, 2024).

Outro fator importante na escolha do Android é sua ampla base de usuários, já que mais de 84% de brasileiros utilizam esse sistema operacional. Isso garante que o aplicativo desenvolvido tenha um alcance significativo e possa impactar um número maior de usuários (POSSA, 2023).

Além disso, a plataforma Android oferece também suporte nativo a várias linguagens de programação e *frameworks*, como Java, Kotlin e, mais recentemente, *frameworks* híbridos como o React Native, facilitando o desenvolvimento multiplataforma e o acesso a funcionalidades nativas do dispositivo.

### JavaScript

JavaScript é uma linguagem de programação amplamente usada para o desenvolvimento de aplicativos web e mobile. Conhecida por sua flexibilidade e simplicidade, o JavaScript se tornou uma escolha popular para o desenvolvimento de aplicativos multiplataforma graças a sua compatibilidade com *frameworks* como React Native, que permite a criação de aplicativos nativos usando uma base de código compartilhada. A escolha pela linguagem JavaScript para este projeto se justifica pela sua curva de aprendizado acessível, pela vasta comunidade de desenvolvedores, e pela sua ampla gama de bibliotecas e ferramentas que facilitam o desenvolvimento de funcionalidades complexas de forma eficiente (FLANAGAN, 2020).

### React Native

O React Native é um *framework* de desenvolvimento de aplicativos mobile, criado pela Meta e que utiliza JavaScript para criar aplicativos nativos para plataformas iOS e Android. O React Native possibilita o desenvolvimento de aplicações multiplataforma com uma única base de código, reduzindo significativamente o tempo e os custos de desenvolvimento. Além disso, ele permite que os aplicativos tenham desempenho semelhante ao dos aplicativos nativos, pois os componentes do React Native são compilados para código nativo, aproveitando ao máximo os recursos do dispositivo (META, 2024).

No desenvolvimento desse projeto, foi escolhido esse *framework* devido a sua eficiência e facilidade de integração com APIs e bibliotecas externas, além de oferecer suporte a funcionalidades nativas, como acesso à câmera, geolocalização e armazenamento local. O React Native também proporciona uma experiência de desenvolvimento moderna com funcionalidades como o *hot reload*, que permite que as mudanças no código sejam refletidas em tempo real no aplicativo, tornando o processo de desenvolvimento mais ágil e interativo.

### Expo

O Expo é uma plataforma e conjunto de ferramentas que facilitam o desenvolvimento, teste e implantação de aplicativos móveis utilizando React Native. Entre suas funcionalidades, estão bibliotecas de APIs prontas para uso, permitindo que os desenvolvedores acessem recursos como câmera, GPS e armazenamento local. Outra utilidade é o aplicativo Expo Go, que após instalado, permite que os desenvolvedores visualizem o projeto em dispositivos físicos durante o desenvolvimento, desde que estejam na mesma rede do host. A plataforma também oferece funções que facilitam a publicação na App Store e Google Play, além do "Expo Application Service" (EAS), que simplifica a geração de aplicativos binários (APK e IPA).

A utilização do Expo no aplicativo Nossa Via permitiu um desenvolvimento muito mais rápido e eficiente. O processo de configurar, implementar e testar o aplicativo foi extremamente simplificado. Também utilizamos suas APIs para obter imagens do armazenamento do dispositivo móvel e a localização do dispositivo.

## Arquitetura cliente-servidor

A arquitetura cliente-servidor é uma modelo de organização de rede, onde os computadores de um sistema são subdivididos em dois grupos, o cliente, que realiza requisições em busca de recursos ou serviços, e os servidores, que são responsáveis por receber e responder a essas requisições (CONTROLE NET, 2024).

### Nuvem

No contexto de tecnologia, a “Nuvem” é o termo utilizado para descrever a ampla gama de servidores remotos conectados por todo o mundo, diversas empresas especializadas vendem o acesso desses servidores, assim, ao invés de armazenar dados ou softwares em suas próprias máquinas físicas, seus clientes podem acessá-los online.

É graças à computação em nuvem, que os componentes de um software podem funcionar sem a necessidade de realizar a computação em suas próprias máquinas físicas. A arquitetura cliente-servidor em particular, permite que os usuários de um sistema possam realizar requisições a um servidor que está localizado até mesmo em outro lado do mundo.

Essa tecnologia oferece diversos benefícios, como a terceirização de recursos, flexibilidade na implementação e facilidade de compartilhamento de dados. Nos serviços em nuvem, a responsabilidade pelo porte, manutenção e gerenciamento de *hardware* e *software* fica a cargo do provedor, dessa forma, o cliente é isento de preocupações relacionadas a esses aspectos. Na maioria dos casos, a implantação desses serviços é rápida e simples, permitindo ao usuário escolher a quantidade, o tipo e até a localização das máquinas necessárias.

Essa personalização é um dos principais motivos que tornam os serviços em nuvem mais econômicos. Além disso, devemos considerar o papel fundamental da nuvem no compartilhamento de dados. Em redes empresariais, os arquivos podem ser armazenados de forma segura e acessível, já em serviços de nuvem pública, é possível compartilhar arquivos globalmente, com algumas empresas oferecendo planos gratuitos até um certo limite de capacidade (GOOGLE, 2024).

Segundo Bhardwaj *et al*. (2010), a Infraestrutura como Serviço (IaaS), é um serviço em nuvem que envolve a entrega de hardware e software como um serviço, permitindo que os usuários provisionem recursos sob demanda sem a necessidade de um compromisso a longo prazo. Diferente da Plataforma como Serviço (PaaS), o provedor de IaaS realiza pouca gestão além de manter o data center operacional, cabendo aos usuários a implantação e gestão dos serviços de software, similar ao que fariam em seus próprios data centers. Alguns exemplos de IaaS incluem Microsoft Azure, Amazon Web Services (AWS), Google Cloud etc. Entre esses, aquele escolhido para hospedar o *back-end* do aplicativo Nossa Via foi o Microsoft Azure.

### Azure

A Azure é uma plataforma de computação em nuvem da Microsoft que disponibiliza uma ampla gama de produtos e serviços. A Azure destaca-se por sua flexibilidade, permitindo a utilização de diferentes ferramentas e *frameworks*, além de oferecer soluções escaláveis e seguras para setores variados, como saúde, finanças, governo e varejo (MICROSOFT, 2024).

Dentre seus serviços, dois se mostraram adequados para serem integrados no aplicativo Nossa Via, os Containers Apps, que são responsáveis pela implantação e execução do *back-end*, e os Blob Storages que armazenam todas as imagens do aplicativo.

#### Container Apps

Segundo Susnjara e Smalley (2024), no desenvolvimento de software, containers são ambientes executáveis que incluem uma aplicação juntamente com todas as suas dependências, bibliotecas e arquivos de configuração essenciais para seu funcionamento.

Os Containers Apps da Azure são uma solução da Microsoft que permite a implantação de containers de maneira fácil e rápida, com isso, não há necessidade de configurar ou gerenciar servidores, além de ajudar a economizar custos durante a utilização dessa tecnologia (Microsoft, 2024).

#### Blob Storage

Binary Large Object, também conhecido como Blob, é uma grande massa de dados binários que pode ou não se enquadrar em um formato de arquivo.

O Blob Storage, é uma forma de armazenamento de objetos*,* onde os Blobs são guardados sem nenhuma hierarquia nos dados, dessa forma, podem ser escalados quase indefinidamente. Esse modelo é particularmente adequado para armazenar dados de mídias, como fotos e vídeos (Could Flare, 2024).

## Integração e Implantação

- introdução de ingraçã e implementação

### Controle de Versionamento

Atualmente, os softwares têm se tornado cada vez mais complexos, isso pode trazer certas dificuldades durante o processo de desenvolvimento como programar colaborativamente, perda de histórico do código, gerenciar versões de um sistema e complicações ao implementar ou revisar mudanças. Esses desafios geraram a inspiração para a criação dos sistemas de controle de versionamento.

Por esses motivos, foram criados os sistemas de controle de versionamento, ferramentas que viabilizam o acompanhamento e gerenciamento das alterações feitas em arquivos ao longo do tempo. Assim, após armazenar um arquivo ou conjunto de arquivos em um repositório como o GitHub, um desenvolvedor pode realizar alterações no código em sua própria máquina e, posteriormente, enviá-las para o repositório.

Criado por Linus Torvalds em 2005, o Git é a ferramenta de controle de versionamento mais utilizada no mercado. Ele possui diversas vantagens sobre seus concorrentes, sendo uma delas a maneira como armazena e organiza seus dados.

O Git não trata nem armazena seus dados desta forma. Em vez disso, o Git trata seus dados mais como um conjunto de imagens de um sistema de arquivos em miniatura. Toda vez que você fizer um commit, ou salvar o estado de seu projeto no Git, ele basicamente tira uma foto de todos os seus arquivos e armazena uma referência para esse conjunto de arquivos. Para ser eficiente, se os arquivos não foram alterados, o Git não armazena o arquivo novamente, apenas um link para o arquivo idêntico anterior já armazenado (Git, 2014).

O Git também possui um sistema de tags, que tem como proposito marcar *commits* específicos, sendo frequentemente utilizado para destacar pontos importantes na linha do tempo de um projeto, como lançamentos ou atualizações oficiais de um software.

Outra funcionalidade essencial que o Git oferece são as chamadas "branches". Elas funcionam como linhas independentes de desenvolvimento dentro de um repositório, o que permite que os desenvolvedores trabalhem em diferentes versões do projeto ao mesmo tempo, experimentem novas ideias e desenvolvam funcionalidades sem afetar a linha principal do código. Além disso, ao enviar atualizações para o repositório remoto, o Git permite escolher quais branches compartilhar, incentivando a experimentação sem exigir o compartilhamento imediato com outros colaboradores (Git).

O Git é utilizado no projeto Nossa Via em conjunto da plataforma GitHub, a fim de garantir um armazenamento seguro, acessível e eficiente.

### GitHub

O GitHub é uma plataforma baseada em nuvem que permite a hospedagem de código-fonte, oferecendo funcionalidades para armazenar, gerenciar e compartilhar sistemas de software utilizando o sistema de controle de versão Git. Além disso, integra-se a diversas ferramentas que auxiliam no desenvolvimento e na gestão de projetos, como Jira, Microsoft Teams e Microsoft Azure. A plataforma também inclui uma ferramenta nativa chamada GitHub Actions, que permite automatizar fluxos de trabalho, especialmente em processos de CI/CD, tornando o ciclo de desenvolvimento mais ágil e eficiente.

Sua popularidade tem crescido nos últimos anos, graças a vantagens como uma estrutura de repositórios que facilita a colaboração e o monitoramento de projetos. A interface amigável e intuitiva também simplifica o uso do Git, tornando-o mais acessível para desenvolvedores de todos os níveis.

No caso do projeto Nossa Via, o GitHub é utilizado para o armazenamento e desenvolvimento, sendo dividido em três repositórios: o Documentacao-TCC-NossaVia, que contém a monografia, diagramas, planos de testes e a documentação completa; o Back-end-TCC-NossaVia, que armazena o código do *back-end* e a *pipeline* de CI/CD do *back-end*; e o Front-end-TCC-NossaVia, com o código do *front-end.*

### GitHub Actions

O GitHub Actions é uma ferramenta integrada ao GitHub, cujo propósito é automatizar e otimizar fluxos de trabalho. Ela permite que os desenvolvedores configurem pipelines para tarefas repetitivas, como testes, builds e deploys, simplificando os processos de CI/CD.

Graças à integração com os repositórios do GitHub, a ferramenta pode, por exemplo, ser configurada para rodar testes automaticamente sempre que um commit é realizado e, em seguida, implantar a nova versão do software após a aprovação dos testes. Isso oferece vantagens como a redução de erros manuais e maior agilidade no desenvolvimento.

No desenvolvimento do aplicativo Nossa Via, utilizamos o GitHub Actions para automatizar os processos de CI/CD do *back-end*. Após efetuar um commit no repositório, é disparada uma nova execução da pipeline onde são executados os testes unitários, e em seguida, caso a versão especificada no arquivo image.json tenha sido alterada, o código é automaticamente integrado e implantado.

# REQUISITOS DO SISTEMA DE SOFTWARE

Este capítulo visa especificar os requisitos funcionais e não funcionais, juntamente com as regras de negócio, além de fornecer o protótipo das telas e o cronograma de atividades para o desenvolvimento de software.

## Requisitos Funcionais

Neste item serão descritos os requisitos funcionais que especificam ações que o sistema deve ser capaz de executar, ou seja, os objetivos do sistema, incluindo prioridade e regras de negócio.

**[RF001] – Cadastrar usuário**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário crie uma conta para a utilização do sistema.

**[RF002] – Realizar login**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário faça o login em sua conta.

**[RF003] – Criar uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário publique uma reclamação sobre as vias.

**[RF004] – Visualizar feed de reclamações**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário veja as principais informações das reclamações em um feed.

**[RF005] – Visualizar detalhes de uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário visualize todos os dados de uma reclamação.

**[RF006] – Curtir uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário adicione ou remova uma curtida em qualquer reclamação.

**[RF007] – Comentar uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário adicione ou remova um comentário em qualquer reclamação.

**[RF008] – Alterar status de uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que um usuário altere o status de uma reclamação.

**[RF009] – Alterar informações de uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário altere alguns dados de uma reclamação feita anteriormente.

**[RF010] – Deletar uma reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário delete uma reclamação feita anteriormente.

**[RF011] – Pontuação de usuário**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | ¨ | Essencial | ¨ | Importante | n | Desejável |

**Descrição**: Este requisito permite que o usuário tenha uma pontuação dentro do sistema.

**[RF012] – Visualizar feed pessoal de reclamação**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | ¨ | Essencial | n | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário visualize um feed contendo apenas suas próprias reclamações.

**[RF013] – Redefinir senha**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prioridade**: | n | Essencial | ¨ | Importante | ¨ | Desejável |

**Descrição**: Este requisito permite que o usuário redefina a sua senha de acesso.

## Regras de Negócio

**[RN001] – Ações do usuário**

**Descrição:** O usuário pode visualizar o feed e os detalhes de reclamações sem estar logado, mas para realizar os demais casos de uso, o usuário deve estar logado.

**[RN002] – Cadastro**

**Descrição:** Para que o usuário realize o cadastro ele deve ter preenchido todos os seguintes campos corretamente: Nome completo; CPF; Data de Nascimento; E-mail; Senha; Telefone; Sexo; CEP; Numero; Rua; Bairro; Cidade e Estado.

**[RN003] – Login**

**Descrição:** Para que o usuário realize o login ele deve estar cadastrado e preencher os campos de e-mail e senha corretamente.

**[RN004] – Criar reclamação**

**Descrição:** Para que o usuário crie uma reclamação ele deve preencher os seguintes campos: Fotos; Descrição; Categoria (podendo ser: “Irregularidades no asfalto”, “Irregularidades na calçada”, “Falta de sinalização”, “Lixo e/ou entulho na via”, “Falta de iluminação”, “Veículo abandonado” e “Outros”), CEP, Numero, Rua, Bairro, Cidade, Estado e coordenadas conforme dados obtidos pelo Google Maps dentro do próprio sistema. Após os campos serem preenchidos o sistema deve verificar se já existe uma reclamação com a mesma categoria na região (raio de 70 metros), caso positivo, será exibido ao usuário uma lista com as reclamações encontras dando a opção de curti-las caso seu problema esteja dentro da lista, ou seguir com a criação de uma nova reclamação.

**[RN005] – Deletar uma reclamação**

**Descrição:** O usuário pode deletar qualquer reclamação feita por ele mesmo.

**[RN006] – Modificar status de uma reclamação**

**Descrição:** O status da reclamação pode ser alterado apenas pelo usuário criador, podendo ser “Resolvida” ou “Não Resolvida”.

**[RN007] – Alterar reclamação**

**Descrição:** O usuário pode alterar as informações de qualquer reclamação feita por ele mesmo, com exceção das imagens da reclamação.

**[RN008] – Visualizar feed de reclamação**

**Descrição:** O feed deve ser organizado pelas reclamações mais próximas do usuário, mais curtidas e que são mais novas, nessa ordem, permitindo o filtro por categoria.

**[RN009] – Sistema de pontuação**

**Descrição:** O sistema dará uma pontuação ao usuário nos seguintes cenários: 10 pontos para cada reclamação criada, 2 pontos a cada comentário feito ou recebido e 1 ponto a cada curtida feita ou recebida.

**[RN010] – Visualizar reclamação**

**Descrição:** Esta regra permite que o usuário visualize todos os dados de uma reclamação, incluindo: Fotos; Descrição; Categoria; Localização; Curtidas; Comentários; Status da reclamação; além de Foto e nome do usuário criador.

**[RN011] – Alterar informações do usuário**

**Descrição:** O sistema deverá permitir que o usuário altere todas as informações da sua conta menos o seu e-mail e CPF. Também adicione uma foto de perfil através da mesma interface.

**[RN012] – Visualizar feed pessoal de reclamação**

**Descrição:** Esta regra permite que o usuário acesse uma área do sistema onde será apresentada uma lista contendo todas as reclamações feitas por ele mesmo, ordenadas por data de criação.

**[RN013] – Visibilidade de reclamações resolvidas**

**Descrição:** As reclamações que estão com status “Resolvida” serão exibidas no feed durante 7 dias, depois disso ficarão ocultadas permitindo consulta apenas na interface de reclamação próprias.

**[RN014] – Recuperar senha**

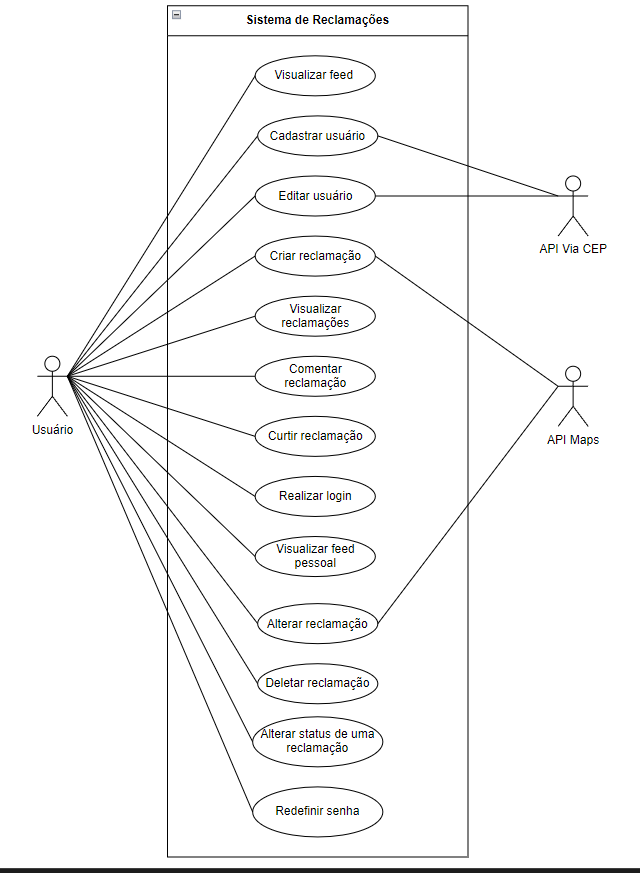
**Descrição:** Esta regra permite que o usuário redefina a sua senha de acesso a partir do seu e-mail cadastrado, a validação de usuário na interface de “Nova Senha” será feita por meio de um token com validade de 30 minutos enviado no e-mail informado.

## Modelagem Funcional

Neste item será descrito os requisitos a serem atendidos funcionalmente pelo sistema, contendo atores e o diagrama de caso de uso.

**Diagrama de caso de uso**

Api’s Maps, Denununcias  
API via CEP usuario

Figura 3 - Diagrama de caso de uso 

Fonte: autores (2024).

**Atores**

Detalhamento dos atores que aparecem no diagrama do caso de uso.

|  |  |
| --- | --- |
| **Nome** | **Descrição** |
| Usuário | Usuário do sistema responsável por criar, visualizar, modificar, alterar o status e deletar reclamações. Além de poder interagir com reclamações de outros usuários. |

**Especificação de casos de uso**

|  |  |
| --- | --- |
| **CSU001 – Criar reclamação** | |
| **Sumário:** | Este caso de uso tem como objetivo a criação de uma nova reclamação |
| **Pré-condição:**  O usuário deve estar logado. [RN001] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o ator usuário escolhe a opção de “Criar reclamação”.  1. O ator usuário adiciona fotos, descrição, coordenadas, CEP, estado, cidade, rua, número e categoria. [RN004]  2. Os dados da reclamação são enviados para o Sistema, que busca reclamações próximas com a mesma categoria. [FA001] e [FA002]  3. Caso não exista uma reclamação próxima na mesma categoria, o sistema permite que o usuário continue a criar uma reclamação. | |
| **Fluxos Alternativos**  **[FA001] Fluxo Alternativo 1: Existem reclamações próximas na mesma categoria e uma delas atende o mesmo problema que o usuário relata.**  Este fluxo alternativo ocorre quando o sistema encontra uma ou mais reclamações próximas e uma delas atende o mesmo problema que o usuário relata.  1. Será exibido ao usuário uma lista com as reclamações encontradas dando a opção de curti-las caso seu problema esteja dentro da lista.  O usuário curte uma reclamação que relata o mesmo problema que a sua.  **[FA002] Fluxo Alternativo 2: Existe reclamações próxima na mesma categoria e nenhuma atende o mesmo problema que o usuário relata.**  Este fluxo alternativo ocorre quando o sistema encontra uma ou mais reclamações próximas e nenhuma atende o mesmo problema que o usuário relata.  1. Será exibido ao usuário uma lista com as reclamações encontradas dando a opção de curti-las caso seu problema esteja dentro da lista.  2. O usuário não identifica uma reclamação que atende o mesmo problema que ele relata.  3. Retorna ao passo 3 do fluxo principal. | |
| Fluxos de Exceção  Não se aplica. | |
| Pós-condições:  O sistema atualiza a pontuação do usuário. [RN009] | |
| Regras de Negócio:  RN001, RN004 e RN009 | |

|  |  |
| --- | --- |
| **CSU002 – Cadastrar usuário** | |
| Sumário: | Este caso de uso tem como objetivo a adição de um novo perfil de usuário no sistema. |
| **Pré-condição:**  Não se aplica. | |
| **Fluxo Principal**  Este caso de uso se inicia quando o ator usuário deseja criar um perfil para si.   1. O ator usuário informa os dados necessários. [RN002] 2. O Sistema verifica se os dados estão de acordo. [FE001] 3. O sistema consolida a criação do perfil. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  **[FE001] Fluxo de Exceção 1: Dados incompletos**  Este fluxo exceção ocorre quando o usuário não informa todos os dados corretamente.   1. O usuário deve corrigir as informações. 2. Retornar ao passo 2 do fluxo principal. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:** RN002 | |

|  |  |
| --- | --- |
| **CSU003 – Realizar login** | |
| Sumário: | Este caso de uso tem como objetivo realizar login no sistema. |
| **Pré-condição:**  O usuário deve possuir cadastro. [RN003] | |
| **Fluxo Principal**  Este caso de uso se inicia quando um usuário quer realizar login.   1. O usuário informa o e-mail e senha da sua conta. [RN003] 2. O sistema valida os dados. [FE001] 3. O usuário entra no sistema. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  **[FE001] Fluxo de Exceção 1: Dados incorretos**  Este fluxo de exceção ocorre quando o usuário informa o e-mail ou senha incorretos.   1. Retornar ao passo 1 do fluxo principal. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:** RN003 | |

|  |  |
| --- | --- |
| **CSU004 – Deletar reclamação** | |
| Sumário: | Este caso de uso tem como objetivo a exclusão de uma reclamação existente |
| **Pré-condição:**  O usuário deve estar logado e dentro do feed pessoal.[RN001] [RN012] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário escolhe a opção de “Deletar reclamação”.   1. O usuário deve selecionar “excluir reclamação” na reclamação que deseja excluir. [RN005] 2. O sistema abre uma janela de alerta solicitando a confirmação da exclusão da reclamação. [FA001] 3. O usuário confirma a exclusão da reclamação. 4. O sistema exclui a reclamação de maneira lógica. | |
| **Fluxos Alternativos**  **[FA001] Fluxo Alternativo 1: Não confirmado**  Este fluxo alternativo ocorre quando o usuário não confirma que deseja deletar a reclamação.  O fluxo é interrompido. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:** RN001, RN005 e RN012 | |

|  |  |
| --- | --- |
| **CSU005 – Editar reclamação** | |
| Sumário: | Este caso de uso tem como objetivo a modificação das informações de uma reclamação. |
| **Pré-condição:**  O usuário deve estar logado e dentro do feed pessoal.[RN001] [RN012] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário escolhe a opção de “Alterar reclamação”.   1. O ator usuário seleciona “alterar reclamação” na reclamação que deseja alterar. 2. O usuário insere as informações que deseja alterar, exceto as imagens. [RN007] 3. O sistema atualiza as informações da reclamação com os dados informados. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:** RN001, RN007 e RN012 | |

|  |  |
| --- | --- |
| **CSU006 – Alterar status de uma reclamação** | |
| Sumário: | Este caso de uso tem como objetivo modificar o status de uma reclamação. |
| **Pré-condição:**  O usuário deve estar logado e dentro do feed pessoal.[RN001] [RN012] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário escolhe a opção de “Alterar status da reclamação”.   1. O usuário escolhe a opção de status que deseja atribuir a reclamação, podendo ser resolvida ou não-resolvida. [RN006] 2. O sistema atualiza o status da reclamação com a opção escolhida. | |
| **Fluxos Alternativo**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:** RN001, RN006 e RN012 | |

|  |  |
| --- | --- |
| **CSU007 – Feed de reclamação** | |
| Sumário: | Este caso de uso tem como objetivo permitir que qualquer interessado veja as reclamações feitas no aplicativo. |
| **Pré-condição:**  Não se aplica. | |
| **Fluxo Principal**  Este caso de uso se inicia quando qualquer usuário entra no APP.   1. O sistema irá organizar o feed. [RN008] | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:**  RN008 | |

|  |  |
| --- | --- |
| **CSU008 – Visualizar reclamação** | |
| Sumário: | Este caso de uso permite que o usuário visualize todos os dados de uma reclamação. |
| **Pré-condição:**  Não se aplica. | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário clica em uma reclamação   1. O sistema irá exibir os seguintes detalhes da reclamação clicada: Fotos; Descrição; Categoria; Localização; Curtidas; Comentários; Status da reclamação; Foto e nome do usuário criador. [RN010] | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:**  RN010 | |

|  |  |
| --- | --- |
| **CSU009 – Curtir reclamação** | |
| Sumário: | Este caso de uso permite que o usuário curta uma reclamação. |
| **Pré-condição:**  O usuário deve estar logado. [RN001] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário clica no botão “Curtir” de uma reclamação.   1. O sistema irá computar a curtida. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  O sistema atualiza a pontuação do usuário. [RN009] | |
| **Regras de Negócio:**  RN001 e RN009 | |

|  |  |
| --- | --- |
| **CSU010 – Comentar reclamação** | |
| Sumário: | Este caso de uso permite que o usuário comente uma reclamação. |
| **Pré-condição:**  O usuário deve estar logado. [RN001] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário clica no botão “Comentar” de uma reclamação.   1. O sistema irá registar o comentário | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  O sistema atualiza a pontuação do usuário. [RN009] | |
| **Regras de Negócio:**  RN001 e RN009 | |

|  |  |
| --- | --- |
| **CSU011 – Editar Usuário** | |
| Sumário: | Este caso de uso permite que o usuário altere as suas informações pessoais. |
| **Pré-condição:**  O usuário deve estar logado. [RN001] | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário escolhe a opção de “Visualizar perfil”.   1. O ator usuário seleciona “Visualizar perfil” no menu do aplicativo. 2. O usuário poderá alterar todas as informações da sua conta exceto o seu e-mail e CPF. [RN011] 3. O usuário confirma a edição dos dados. 4. O sistema atualiza as informações do usuário com os dados informados. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  Não se aplica. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:**  RN001 e RN011 | |

|  |  |
| --- | --- |
| **CSU012 – Recuperar senha** | |
| Sumário: | Este caso de uso permite que o usuário redefina a sua senha de acesso ao sistema. |
| **Pré-condição:**  Não se aplica. | |
| **Fluxo Principal**  Este caso de uso se inicia quando o usuário escolhe a opção de “Redefinir Senha”.   1. O ator usuário informa o seu e-mail cadastrado. [RN014] 2. O sistema valida se o e-mail está cadastrado. [FE001] 3. O sistema gera um token e envia para o e-mail cadastrado. 4. O usuário informa o token recebido e sua nova senha. | |
| **Fluxos Alternativos**  Não se aplica. | |
| **Fluxos de Exceção**  **[FE001] Fluxo de Exceção 1: E-mail não cadastrado no sistema**  Este fluxo de exceção ocorre quando o usuário informa um e-mail que não está cadastrado no sistema.   1. O fluxo é interrompido. | |
| **Pós-condições:**  Não se aplica. | |
| **Regras de Negócio:**  [RN014] | |

Caso de uso para Editar ?

## Requisitos Não-Funcionais

**[RNF001] - Segurança**

**Descrição:** O sistema deve estar protegido de acessos de usuários não permitidos, por meio de login, senha criptografada e controle de sessões. A senha deve conter no mínimo 8 dígitos dentre eles letras minúsculas, maiúsculas, números e caracteres especiais. Esses valores serão criptografados utilizando o método AES256 e o controle de sessão via tokens JWT para manter a conexão do usuário logado.

**[RNF002] - Usabilidade**

**Descrição:** A interface do sistema deve ser simples e amigável para o usuário, permitindo que qualquer ação dentro dele seja feita em uma média de 2 minutos.

**[RNF003] - Arquitetura**

**Descrição:** A arquitetura utilizada seguirá o modelo de projeto MVC junto com o padrão DAO.

**[RNF004] - Tecnologia**

**Descrição:** A tecnologia utilizada para o *back-end* será Python com o micro *framework* Flask; para o *front-end* será utilizado a biblioteca React Native; e para o banco de banco de dados, PostgreSQL.

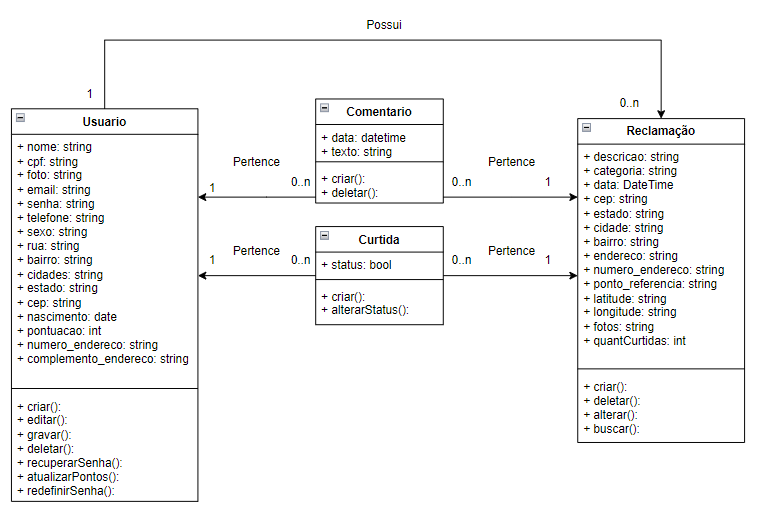
# ANÁLISE

Este capítulo tem como objetivo analisar, detalhar e propor uma solução geral do sistema, sob o ponto de vista de negócio, de acordo com os requisitos levantados e validados no capítulo 3.

## Diagrama de Classes de Análise (Visão de Negócio)

Neste item, o diagrama de classes é exibido, compilando todas as classes selecionadas para o sistema. Ele também detalha os atributos e métodos pertencentes a cada classe, bem como as conexões existentes entre elas.

Figura 4 - Diagrama de classes

**** Fonte: autores (2024).

# PROJETO

Nesse capítulo será apresentado a solução geral do sistema, contendo solução técnica com visão de projeto e implementação, arquitetura e tecnologias utilizadas.

## Arquitetura do Sistema

Neste item é apresentada a arquitetura do sistema por meio de um diagrama de implantação.

Figura 5 - Diagrama de implantação

Diagrama

Descrição gerada automaticamente

Fonte: autores (2024).

VOCÊS TEM UMA INFRA DE DEPLOY MAIS COMPLETA. CERTO? - REPRESENTAR.

## Diagrama de Classes de Projeto por Caso de Uso

### Diagrama de classes do caso de uso criar reclamação

Figura 6 - Diagrama de classes do caso de uso criar reclamação

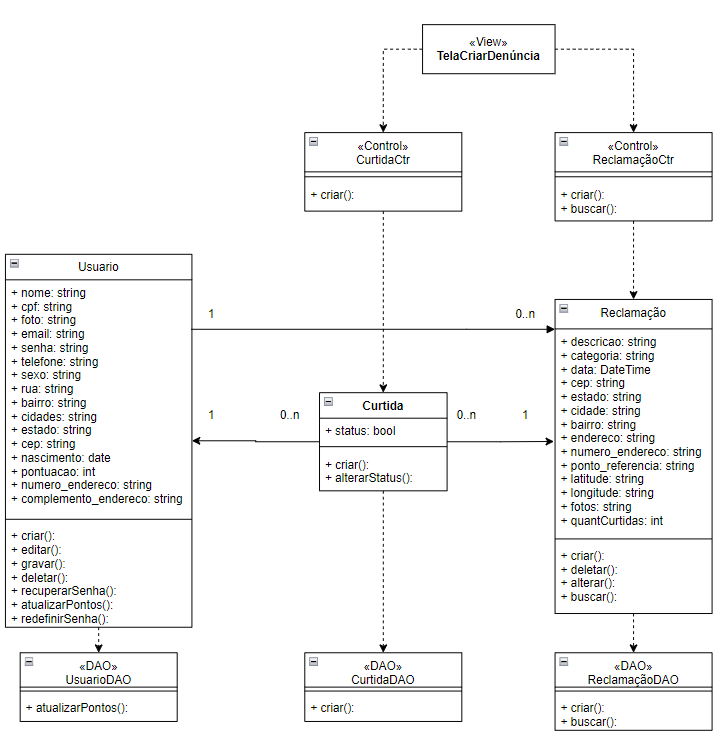
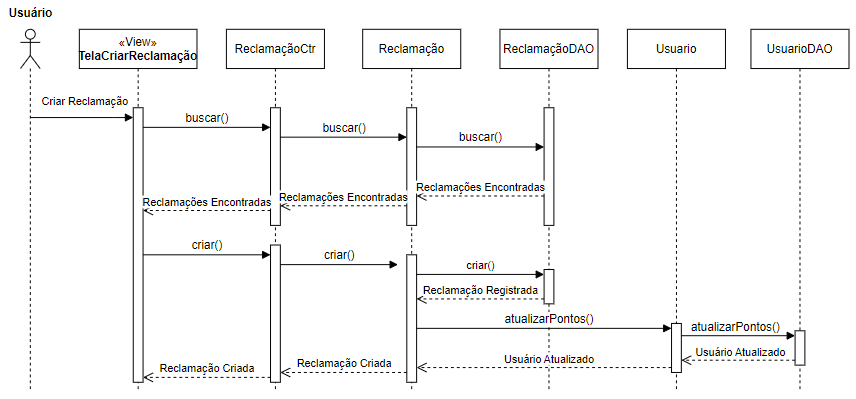
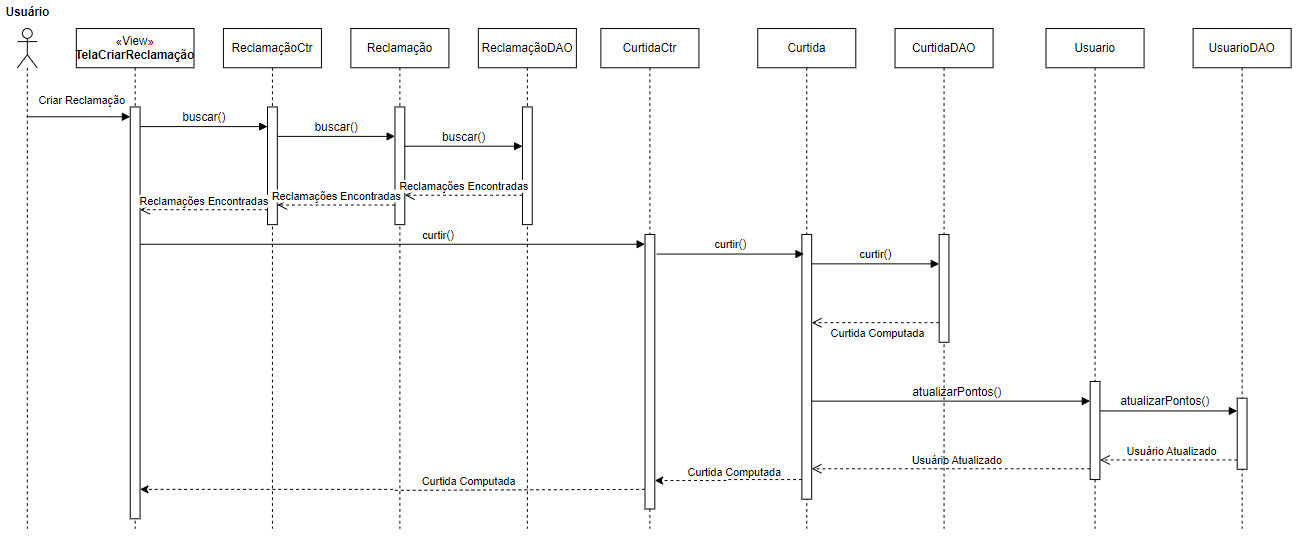
 Fonte: autores (2024).

Figura 7 - Diagrama de sequência criar reclamação. – Fluxo Principal CSU001

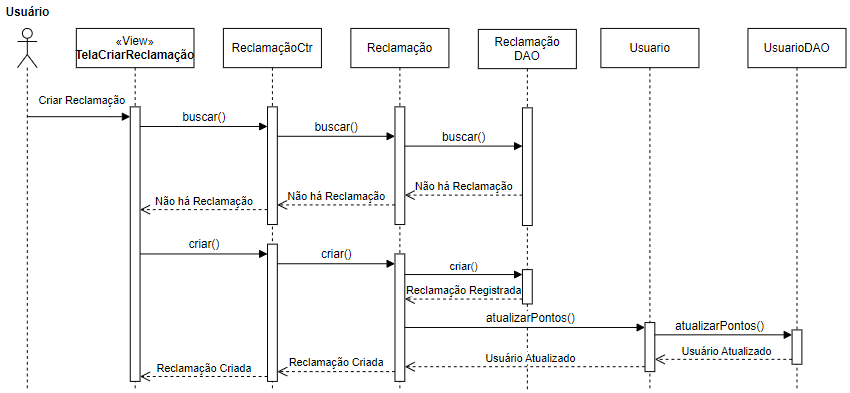


Fonte: autores (2024).

Figura 8 - Diagrama de sequência criar reclamação – Fluxo Alternativo 1 CSU001



Fonte: autores (2024).

Figura 9 - Diagrama de sequência criar reclamação – Fluxo Alternativo 2 CSU001  Fonte: autores (2024).

### Diagrama de classes do caso de uso comentar reclamação

Figura 10 - Diagrama de classes do caso de uso comentar reclamação

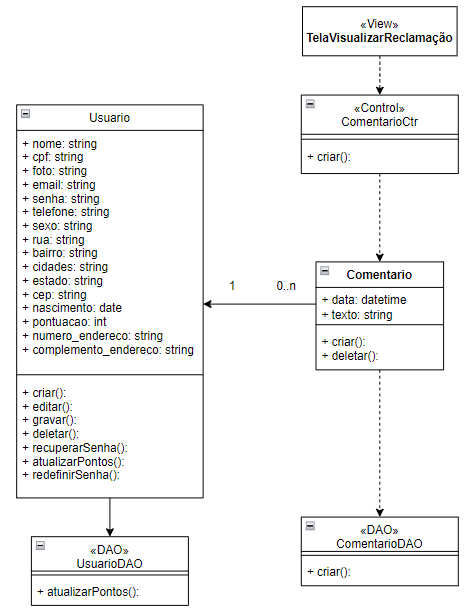
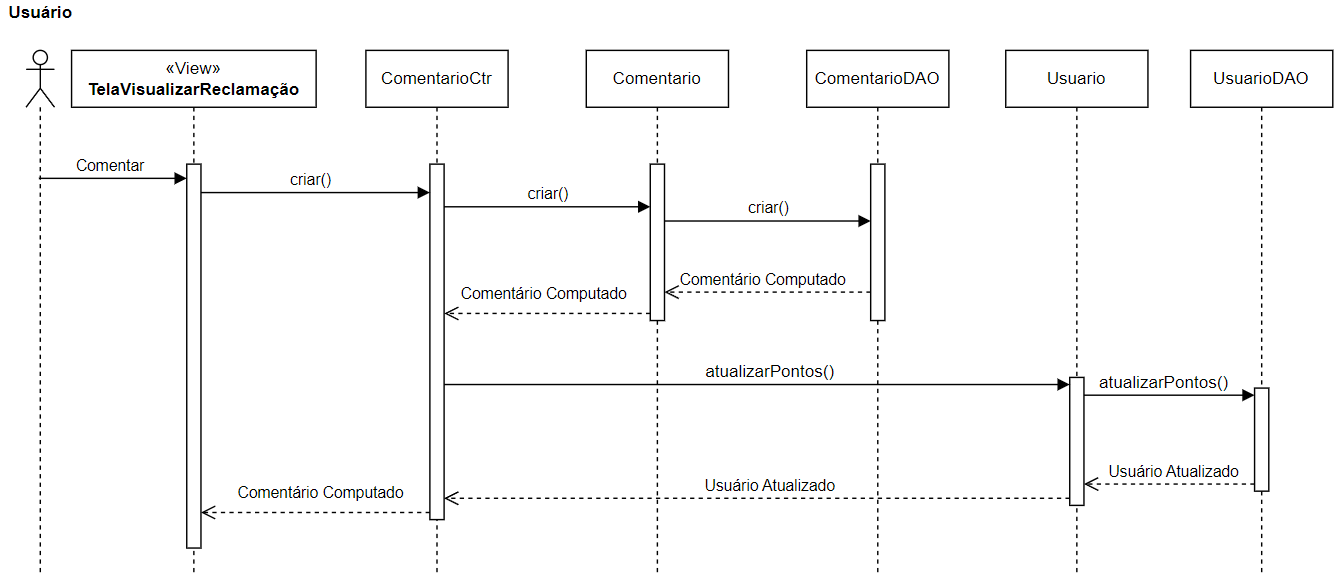
 Fonte: autores (2024).

Figura 11 - Diagrama de sequência comentar reclamação – Fluxo Principal CSU011  Fonte: autores (2024).

### Diagrama de classes do caso de uso curtir reclamação

Figura 12 - Diagrama de classes do caso de uso curtir reclamação

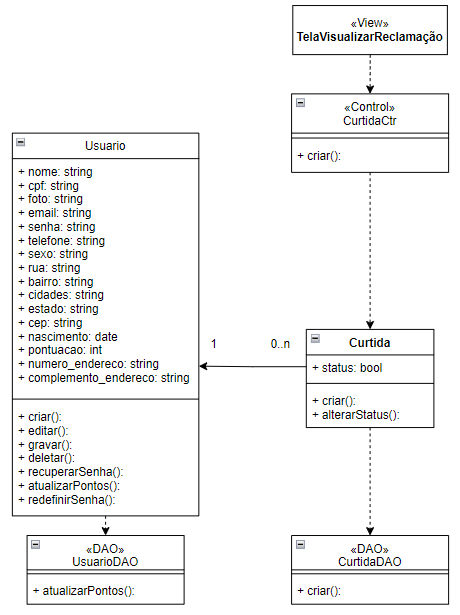
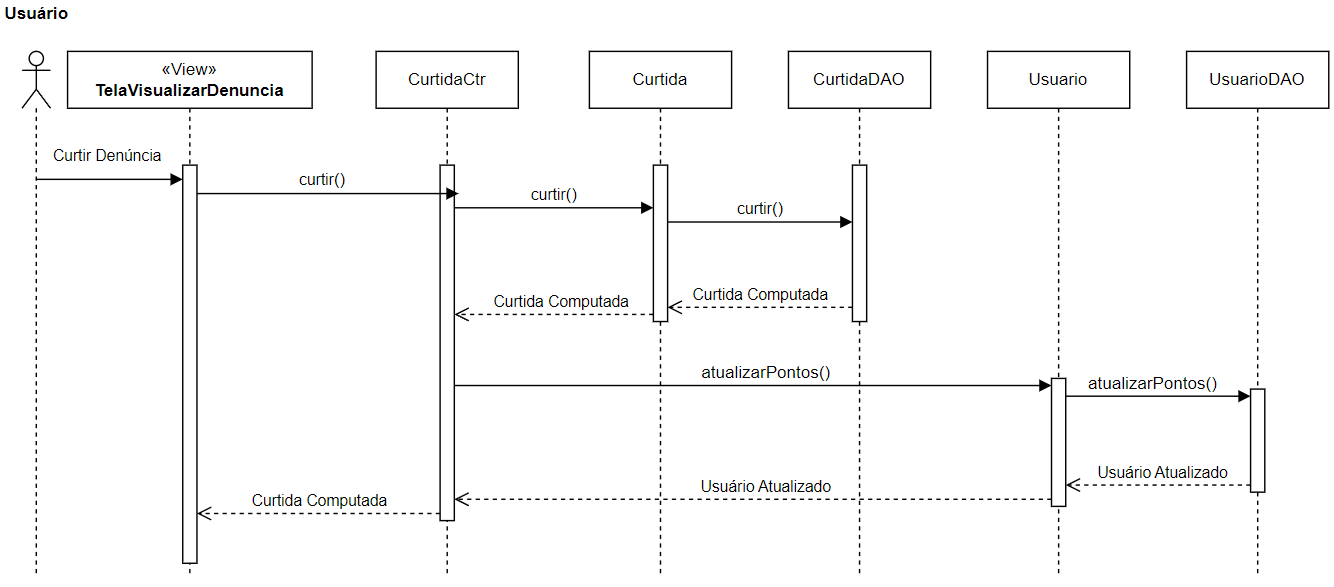
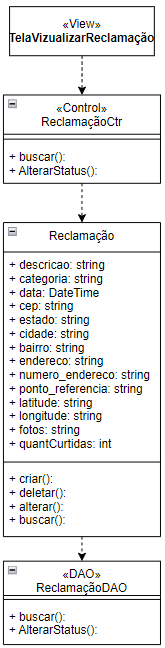
   
 Fonte: autores (2024).

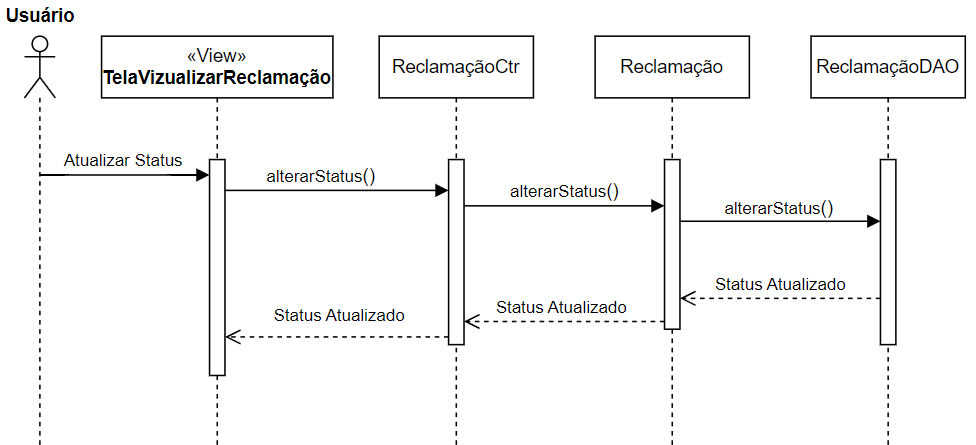
Figura 13 - Diagrama de sequência curtir reclamação – Fluxo Principal CSU010 Fonte: autores (2024).

### Diagrama de classes do caso de uso alterar status

Figura 14 - Diagrama de classes do caso de uso alterar status

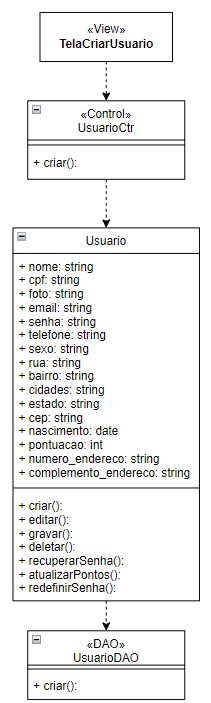


Fonte: autores (2024).

Figura 15 - Diagrama de sequência alterar status reclamação – Fluxo Principal CSU006 Fonte: autores (2024).

### Diagrama de classes do caso de uso cadastrar usuário

Figura 16 - Diagrama de classes do caso de uso cadastrar usuário



Fonte: autores (2024).

Figura 17 – Diagrama de sequência cadastrar usuário – Fluxo Principal CSU002



Fonte: autores (2024).

## Diagrama de atividades

O diagrama de atividades representa o detalhamento de tarefas e o fluxo de uma atividade para outra de um sistema.

Figura 18 - Diagrama de atividades

Diagrama

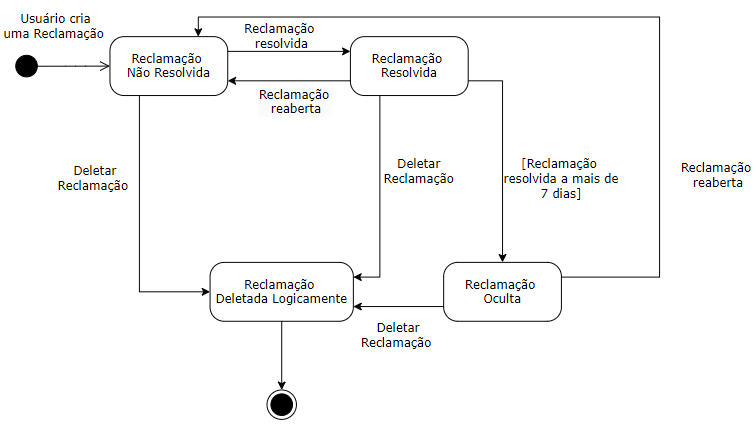
Descrição gerada automaticamente

Fonte: autores (2024).

## Diagrama de estados

O diagrama de estados especifica as sequências de estados pelas quais o objeto pode passar durante seu ciclo de vida em resposta a eventos.

Figura 19 - Diagrama de estados da classe reclamação

 Fonte: autores (2024).

# CONSIDERAÇÕES FINAIS

Ao encerrar o trabalho, devem-se retomar os elementos apresentados na introdução (objetivos, problema, justificativa,) e os dados levantados ao longo do desenvolvimento a fim de “fechar” o encadeamento lógico das ideias, constituindo a análise final do assunto. De acordo com o Centro Paula Souza (2008, p. 14), a conclusão “[...] é uma reafirmação das ideias principais tratadas no documento, a partir dos resultados da pesquisa.”

Em trabalhos realizados ao longo do curso é interessante que se denomine esta parte final de ’Considerações Finais‘, por se tratar de ’estudos exploratórios‘ que não possuem, necessariamente, uma conclusão definitiva.

# REFERÊNCIAS

BASTOS, Jennifer Ester de Sousa *et al*. O Uso do Questionário como Ferramenta Metodológica: potencialidades e desafios. **Brazilian Journal of Implantology and Health Sciences,** *[S. l.]*, v. 5, n. 3, p. 623–636, 20 jun. 2023. DOI: 10.36557/2674-8169.2023v5n3p623-636. Disponível em: <https://bjihs.emnuvens.com.br/bjihs/article/view/304>. Acesso em: 14 abr. 2024.

DATAFOLHA: 84% dos moradores na cidade de São Paulo reclamam de buracos no asfalto. **G1**, São Paulo, 16 mar. 2024. São Paulo, Disponível em: https://g1.globo.com/sp/noticia/2024/03/16/datafolha-84percent-dos-moradores-de-sp-reclamam-de-buracos-no-asfalto.ghtml. Acesso em: 14 abr. 2024.

Agência CNT Transporte Atual. [Piora a qualidade das rodovias brasileiras.](https://www.cnt.org.br/agencia-cnt/piora-a-qualidade-das-rodovias-brasileiras) **CNT**, 22 de out. 2019. Brasília, Disponível em: <https://www.cnt.org.br/agencia-cnt/piora-a-qualidade-das-rodovias-brasileiras.> Acesso em: 14 abr. 2024.

GARCIA, Lara Yamamura. O descarte incorreto de lixo no Brasil e o impacto causado na população. **Coisa pública**, 6 de set. 2023, Disponível em: https://wp.ufpel.edu.br/coisapublica/2023/09/06/o-descarte-incorreto-de-lixo-no-brasil-e-o-impacto-causado-na-populacao/. Acesso em: 14 abr. 2024.

# APÊNDICE A – INSTRUMENTO DE PESQUISA

# Levantamento dos Requisitos do Sistema de Software

O processo de desenvolvimento de software é complexo, que exige uma compreensão profunda das necessidades e expectativas dos usuários finais. A etapa crucial nesse processo é o levantamento de requisitos do sistema de software, pois é a base sobre a qual todo o projeto será construído. Este capítulo servirá como base sólida para o desenvolvimento do sistema de software que será abordado neste trabalho de conclusão de curso. Será Apresentado o levantamento dos requisitos do Sistema de Software e a forma de extração dos Requisitos.

## 1.1. Extração de Requisitos

No processo de extração de requisitos, foi escolhido a utilização de um questionário online, optando por essa abordagem devido à sua eficácia em coletar informações de forma estruturada e abrangente.

Link do questionário: <https://forms.gle/oTDUjvQfJZXWjbnz5>

**1.2. Análise da Coleta de Requisitos**

A extração de requisitos é uma fase crítica no ciclo de desenvolvimento de software, na qual as informações essenciais para o projeto são identificadas, documentadas e analisadas. Este capítulo se dedica a uma análise detalhada das 97 respostas obtidas do processo de extração de requisitos utilizado neste trabalho.

**Gráfico 1** – Pergunta 1



Fonte: próprios autores (2023).

Ao examinarmos o gráfico de respostas da Figura 1, que se refere à pergunta 1 "Você já enfrentou ou deparou-se com algum tipo de problema nas vias públicas?", fica evidente que 94,8% das respostas indicam uma experiência afirmativa, enquanto 5,2% responderam negativamente.

A constatação de que a maioria das pessoas enfrentou ou deparou-se com problemas nas vias públicas é de extrema relevância para o desenvolvimento de um software voltado para denunciar esses problemas, pois valida a necessidade do mesmo, demonstrando que há uma demanda real por soluções que auxiliem na resolução dessas questões.

**Gráfico 2** – Pergunta 2



Fonte: próprios autores (2023).

Analisando o gráfico da figura 2, que se refere a pergunta 2 “Você já teve a necessidade de registrar uma reclamação ou denúncia relacionada às vias públicas?”, podemos observar que 61,9% das respostas foram negativas e 38,1% foram positivas.

Isso pode nos mostrar que a maioria das pessoas não sentem à vontade/necessidade de chegar a registrar uma reclamação ou denuncia pelos canais atuais.

**Gráfico 3** – Pergunta 3



Fonte: próprios autores (2023).

Analisando o gráfico da figura 3, referente a pergunta 3 “Se sim, essa reclamação foi resolvida?” que se relaciona com a pergunta 2. fica claro que 70,8% nunca abriu uma denúncia/reclamação, 9,4% já abriram uma reclamação e seu problema foi resolvido e que 19,8% não tiveram a sua reclamação resolvida.

Podemos levar em consideração que a maioria das pessoas que fizeram uma reclamação não obtiveram resposta, nos indicando que o sistema atual tem uma “falha”, abrindo um possível espaço para desenvolvimento de um novo software na área.

**Gráfico 4** – Pergunta 4



Fonte: próprios autores (2023).

Ao examinarmos o gráfico da figura 4, referente a pergunta 4 “você acredita que uma maior visibilidade das suas reclamações ajudaria a resolver os problemas de forma mais rápida e eficiente?”, podemos concluir que 95,9% das pessoas acreditam que se suas reclamações tivessem maior visibilidade seus problemas seriam resolvidos de forma mais rápida. Essa conclusão seria mais uma validação da necessidade do software.

**Gráfico 5** – Pergunta 5



Fonte: próprios autores (2023).

Ao examinarmos o gráfico de respostas da Figura 5, que se refere à pergunta 5 "Como você avalia a qualidade da manutenção e fiscalização das vias públicas?", fica evidente que 71,1% de pessoas avaliam a qualidade da manutenção e fiscalização das vias públicas insatisfatória ou muito insatisfatória. Essa conclusão nos indica que as vias públicas são de má qualidade aos olhos da maioria do público da pesquisa, de maneira que valida a necessidade o nosso software e nos ajuda a garantir as reais necessidades dos usuários.

**Gráfico 6** – Pergunta 6



Fonte: próprios autores (2023).

Observando o gráfico de respostas para a Figura 6, que se refere à pergunta 6 "Você gostaria de ter acesso aos dados sobre problemas nas vias públicas do seu bairro?", podemos notar que 99% das pessoas responderam sim e 1% respondeu não. Isso nos auxilia a entender as funcionalidades e requisitos que poderemos adicionar no software de acordo com a necessidade dos usuários, como nesse caso, poderemos deixar os dados disponíveis para serem consultados pelos usuários.

**Gráfico 7** – Pergunta 7



Fonte: próprios autores (2023).

Analisando o gráfico da figura 7, que se refere a pergunta 7 “Você consideraria utilizar um site que permitisse denunciar problemas nas vias públicas de forma eficaz?”, podemos observar que 95,9% das respostas foram afirmativas e 4,1% foram negativas. Novamente a respostas dessa pergunta sendo uma ala porcentagem de pessoas que utilizariam um software de denúncia de problemas nas vias públicas valida a necessidade do mesmo, mostrando que há uma demanda real por soluções que auxiliam na resolução desses problemas.

**Gráfico 8** – Pergunta 8



Fonte: próprios autores (2023).

Ao examinarmos o gráfico da figura 8, referente a pergunta 8 “Você acha interessante termos um sistema de pontuação para as denúncias apresentadas na plataforma?”, podemos concluir que a forte concordância (97,9%) em relação à implementação de um sistema de pontuação para as denúncias apresentadas na plataforma tem implicações significativas para o desenvolvimento do software de denúncia de problemas nas vias públicas, sendo elas:

* + Engajamento e Incentivo: A alta taxa de aprovação indica que os usuários veem valor em um sistema de pontuação. Isso pode motivar mais pessoas a usar o software e a contribuir com denúncias, sabendo que seu engajamento será reconhecido e recompensado.
  + Melhoria da Qualidade das Denúncias: Um sistema de pontuação pode incentivar os usuários a fornecerem denúncias mais detalhadas e úteis. Isso pode resultar em informações de maior qualidade, facilitando a identificação e resolução de problemas pelas autoridades competentes.
  + Gamificação: A implementação de um sistema de pontuação pode introduzir elementos de gamificação, tornando a experiência de uso do software mais envolvente. Os usuários podem competir para obter pontuações mais altas, o que pode aumentar o envolvimento e a fidelidade à plataforma.
  + Transparência e Confiança: Um sistema de pontuação transparente e justo pode aumentar a confiança dos usuários na plataforma, pois eles percebem que as denúncias são tratadas de maneira objetiva e equitativa.
  + Feedback e Reconhecimento: A pontuação pode ser usada para reconhecer e premiar os usuários mais ativos e contributivos. Isso pode incluir certificados, distintivos ou até mesmo recompensas tangíveis, incentivando a participação contínua.

**Gráfico 9** – Pergunta 9



Fonte: próprios autores (2023).

Observando o gráfico de respostas para a Figura 9, que se refere à pergunta 9 "Em sua opinião, seria interessante um sistema de classificação que destacasse a relevância das denúncias?", podemos notar que 95,9% das pessoas responderam sim e 4,1% responderam não. Podemos notar que, a forte aceitação da ideia de um sistema de classificação de relevância indica que os usuários valorizam a priorização e a classificação. Isso deve ser levado em consideração no desenvolvimento do software, visando aprimorar a eficácia na resolução de problemas nas vias públicas e na satisfação dos usuários.

**Gráfico 10** – Pergunta 10



Fonte: próprios autores (2023).

Ao examinarmos o gráfico de respostas da Figura 10, que se refere à pergunta 10 “Você acha útil receber atualizações sobre o progresso das suas denúncias?", fica evidente que 96,9% responderam sim e que 3,1%responderam que não, isso nos mostra que a alta aceitação da ideia de receber atualizações sobre o progresso das denúncias indica que os usuários valorizam a comunicação transparente, a confirmação de que suas preocupações estão sendo tratadas e a oportunidade de se envolverem continuamente no processo de resolução de problemas. Isso deve ser considerado visando uma experiencia eficaz para os usuários.

**Gráfico 11** – Pergunta 11



Fonte: próprios autores (2023).

Analisando o gráfico de respostas da figura 11, que se refere a pergunta 11 “Quais problemas relacionados às vias públicas você se sentiria motivado a denunciar usando esta plataforma?”. Essa pergunta foi respondida livremente pelo público, utilizando as suas próprias palavras e podendo ser selecionada mais de uma resposta, assim podemos notar que a maioria das pessoas se sentiriam motivados a denunciar utilizando essa plataforma os seguintes problemas:

* + Buracos e irregularidades nas estradas – 86 pessoas
  + Falta de manutenção em calçadas – 78 pessoas
  + Problemas de iluminação pública – 74 pessoas

Essas respostas no ajudaram a decidir quais denúncias seriam possíveis de realizar dentro do site, priorizando sempre a experiencia e necessidades dos usuários.