**UNIVERSIDADE PAULISTA – UNIP**

GUILHERME SANTOS GAMA - G254937
LENNEKER INDEMBERG SILVA NASCIMENTO - G090JI6
LUIZ GUSTAVO SILVA SOUSA - N724AD0
VINÍCIUS MARQUES DE OLIVEIRA - G24BBB2

# ALÔ, PREFEITURA

ATIVIDADE PRÁTICA SUPERVISIONADA

SÃO PAULO
2024

GUILHERME SANTOS GAMA
LENNEKER INDEMBERG SILVA NASCIMENTO
LUIZ GUSTAVO SILVA SOUSA
VINÍCIUS MARQUES DE OLIVEIRA

# ALÔ, PREFEITURA

ATIVIDADE PRÁTICA SUPERVISIONADA

Atividade Prática Supervisionada submetida ao Curso de Graduação em Ciência da Computação da Universidade Paulista, campus Paulista, como parte dos requisitos para a conclusão do semestre

SÃO PAULO
2024

GUILHERME SANTOS GAMA
LENNEKER INDEMBERG SILVA NASCIMENTO
LUIZ GUSTAVO SILVA SOUSA
VINÍCIUS MARQUES DE OLIVEIRA

# ALÔ, PREFEITURA

ATIVIDADE PRÁTICA SUPERVISIONADA

Atividade Prática Supervisionada submetida ao Curso de Graduação em Ciência da Computação da Universidade Paulista, campus Paulista, como parte dos requisitos para a conclusão do semestre

**BANCA EXAMINADORA**

Prof. Lauro Tomiatti

SÃO PAULO
2024

À nossa jornada compartilhada de aprendizado, dedicamos este trabalho a Guilherme Santos Gama, Lenneker Indemberg Silva Nascimento, Luiz Gustavo Silva Souza e Vinícius Marques de Oliveira. Que cada linha escrita represente não apenas o nosso esforço conjunto, mas também a nossa amizade e colaboração. Juntos, enfrentamos desafios, celebramos conquistas e construímos memórias inesquecíveis. Que esta Atividade Prática Supervisionada seja um testemunho do nosso comprometimento e trabalho em equipe. Expressamos nossa mais profunda gratidão por caminharem ao nosso lado nesta jornada acadêmica.

*"Falar é fácil. Mostre-me o código. Programar é sobre transformar ideias em realidade e garantir que a solução funcione de maneira eficiente e clara."*
*(Linus Torvalds)*

## LISTA DE ILUSTRAÇÕES

Figura 1 - Padrão de codificação adotado no projeto.....34
Figura 2 - Estrutura HTML e CSS do componente de card.....36
Figura 3 - JavaScript aplicado para dinamismo e interatividade no componente de card.....38
Figura 4 - Código do endpoint responsável pela comunicação entre a aplicação e o microserviço.....40
Figura 5 - Código do formulário que recebe o CEP.....43
Figura 6 - Código de integração com a API ViaCEP.....44

## LISTA DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| API | Interface de Programação de Aplicações (Application Programming Interface) |
| APP | Aplicação Web (Application) |
| CEP | Código de Endereçamento Postal |
| CSS | Cascading Style Sheets (Folhas de Estilo em Cascata) |
| EndPoint | Ponto de acesso de rede no microserviço, utilizado para receber requisições e enviar respostas. |
| HTML | Hypertext Markup Language (Linguagem de Marcação de Hipertexto) |
| JS | JavaScript (Linguagem de Programação para Desenvolvimento Web) |
| JSON | JavaScript Object Notation (Notação de Objetos JavaScript) |
| PHP | Hypertext Preprocessor (Pré-processador de Hipertexto) |
| Porta 8000 | Refere-se ao número da porta na qual o microserviço está configurado para rodar no ambiente local, permitindo sua comunicação isolada e sem interferir no sistema principal. |
| SMTP | Simple Mail Transfer Protocol (Protocolo Simples de Transferência de Correio) |
| SQL | Structured Query Language (Linguagem de Consulta Estruturada) |
| TI | Tecnologia da Informação |
| UI | Interface do Usuário (User Interface) |
| UX | Experiência do Usuário (User Experience) |
| XSS | Cross-Site Scripting |

**SUMÁRIO**

1 INTRODUÇÃO ..... 10
2 FUNDAMENTOS DAS TECNOLOGIAS PARA DISPOSITIVOS MÓVEIS ESCOLHIDAS ..... 12
2.1 HTML (Hypertext Markup Language) ..... 12
2.2 HTML (Hypertext Markup Language) ..... 12
2.3 JavaScript ..... 13
2.4 Bootstrap ..... 13
2.5 VIACEP – API de Consulta de CEP ..... 13
2.6 Microserviço para Envio de E-mails Automáticos ..... 15
2.7 Banco de Dados MySQL em Servidor Separado ..... 17
2.8 Criptografia e Segurança de Dados ..... 19
3 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO ..... 20
3.1 Tecnologias Fundamentais ..... 20
3.2 Integração com APIs e Microserviços ..... 24
3.3 Banco de Dados ..... 25
3.4 Implementação e Escalabilidade ..... 27
3.5 Ferramentas de Desenvolvimento ..... 27
3.6 Testes e Monitoramento ..... 27
3.7 Conclusão ..... 28
4 ESTRUTURA DO PROGRAMA ..... 29
4.1 Estrutura de Código da Aplicação Web ..... 29
4.2 Estrutura de Código do Microserviço de Envio de E-mail ..... 32
5 RELATÓRIO COM LINHAS DE CÓDIGO ..... 33
5.1 Front-End Modular ..... 33
5.2 Cartão Modular ..... 35
5.3 Endpoint de Recebimento de Dados e Envio de E-mail ..... 39

5.4 Uso da API ViaCEP .....41
6 CONCLUSÃO .....46
7 REFERÊNCIAS .....47

# 1 INTRODUÇÃO

A crescente degradação ambiental, gerada principalmente por ações humanas irresponsáveis, tem se tornado um dos maiores desafios enfrentados pelas cidades ao redor do mundo. Em resposta a essa situação, é essencial criar mecanismos que possibilitem o monitoramento, a denúncia e a correção dessas irregularidades de maneira eficiente e acessível. Pensando nisso, foi desenvolvida uma aplicação web (APP) para a abertura de denúncias de irregularidades ambientais, com foco no estado de São Paulo. Este sistema visa oferecer à população uma ferramenta prática e direta para reportar problemas ambientais, permitindo uma interação mais eficaz com os órgãos responsáveis pela fiscalização e ação corretiva.

A aplicação foi projetada com uma abordagem "mobile first", garantindo que ela seja totalmente acessível e funcional em dispositivos móveis (smartphones e tablets). Este enfoque é fundamental para atender à realidade atual, onde grande parte da população utiliza smartphones como principal meio de acesso à internet. Utilizando o framework Bootstrap, a aplicação oferece uma interface adaptativa (UI), que se ajusta a diferentes tamanhos de tela, garantindo uma experiência de usuário (UX) fluida e intuitiva, independentemente do dispositivo utilizado. A escolha de tecnologias modernas e responsivas facilita a construção de um sistema ágil, acessível e escalável.

O sistema permite que os cidadãos registrem denúncias de irregularidades ambientais, como poluição, desmatamento ilegal e outras práticas prejudiciais ao meio ambiente. Os usuários podem fornecer detalhes sobre o tipo de problema, a localização e uma descrição completa da situação, além de anexar imagens, quando necessário. Dessa forma, a plataforma não apenas coleta dados importantes, mas também organiza as informações de maneira a facilitar a análise e a tomada de ação pelos órgãos responsáveis. Além disso, o sistema envia automaticamente uma confirmação ao usuário, garantindo que a denúncia foi registrada com sucesso, e fornece um número de protocolo para acompanhamento.

Embora a aplicação tenha sido desenvolvida com o objetivo de atender às necessidades de monitoramento ambiental no estado de São Paulo, ela foi criada para

ser cedida gratuitamente à prefeitura, caso desejem adotá-la em suas operações. O sistema oferece uma maneira de fortalecer a participação cidadã na proteção do meio ambiente, possibilitando que a população se envolva de forma ativa na identificação e resolução de problemas ambientais. A plataforma representa uma importante ferramenta para facilitar a comunicação entre os cidadãos e os órgãos públicos, promovendo maior transparência, eficiência e agilidade no processo de fiscalização ambiental.

Este trabalho descreve o processo de desenvolvimento da aplicação, abordando as tecnologias utilizadas, os desafios enfrentados durante a implementação e as soluções adotadas para garantir a usabilidade, segurança e desempenho do sistema. A importância de uma aplicação como essa é refletida não apenas na eficiência operacional, mas também no fortalecimento das políticas públicas voltadas à preservação ambiental, proporcionando uma plataforma acessível para o engajamento da comunidade no enfrentamento dos desafios ambientais. Assim, a aplicação proposta visa contribuir significativamente para a melhoria da qualidade de vida e a proteção do meio ambiente, oferecendo à população uma ferramenta moderna e eficaz para reportar irregularidades e colaborar ativamente na construção de um futuro sustentável.

# 2 FUNDAMENTOS DAS TECNOLOGIAS PARA DISPOSITIVOS MÓVEIS ESCOLHIDAS

O desenvolvimento de uma aplicação web voltada para dispositivos móveis exige uma compreensão das tecnologias que possibilitam o acesso e a interação do usuário de maneira eficiente e responsiva. No caso deste projeto, que envolve a criação de uma plataforma de denúncias ambientais, utilizamos uma combinação de tecnologias fundamentais para garantir uma experiência de uso adequada em dispositivos móveis. Entre as principais tecnologias utilizadas, destacam-se o HTML, CSS, JavaScript, o framework Bootstrap, o uso de uma API de consulta de CEP, além de um microserviço para o envio automático de e-mails.

## 2.1 HTML (Hypertext Markup Language)

O HTML é a espinha dorsal de qualquer aplicação web, fornecendo a estrutura necessária para apresentar conteúdo na web. Para dispositivos móveis, o HTML precisa ser implementado de forma semântica e eficiente, garantindo que a aplicação carregue rapidamente e seja acessível em diferentes tamanhos de tela. Utilizamos a versão mais recente do HTML, que permite maior flexibilidade para construir layouts responsivos e adaptáveis, utilizando elementos como formulários, tabelas e links, essenciais para a funcionalidade da aplicação de denúncias.

## 2.2 HTML (Hypertext Markup Language)

O CSS é responsável pela apresentação da aplicação, incluindo o design, layout e a estética visual. Em um ambiente móvel, a utilização de CSS adequado é crucial para garantir que a interface do usuário seja legível, atraente e funcional, independentemente do dispositivo. Utilizamos o conceito de design responsivo, adaptando o layout da aplicação a diferentes tamanhos de tela. Através de media queries, conseguimos ajustar os elementos da interface para que se adequem a celulares e tablets, melhorando a navegação e a experiência do usuário.

## 2.3 JavaScript

O JavaScript é uma linguagem fundamental para a criação de interatividade nas páginas da web. No contexto de dispositivos móveis, o JavaScript é essencial para proporcionar uma experiência dinâmica, como a validação de formulários em tempo real, o preenchimento automático de campos com base na consulta de CEP, e a comunicação assíncrona com o servidor para garantir que os dados sejam enviados ou recebidos sem a necessidade de recarregar a página. A biblioteca AJAX, utilizada em conjunto com o JavaScript, permite a troca de dados entre o cliente e o servidor de maneira rápida e eficiente, o que é fundamental para a performance de uma aplicação móvel.

## 2.4 Bootstrap

O Bootstrap é um framework front-end amplamente utilizado para criar layouts responsivos e otimizados para dispositivos móveis. Ele foi escolhido neste projeto por sua facilidade de uso e por fornecer uma série de componentes prontos (como botões, menus, modais e formularios) que se adaptam automaticamente a diferentes tamanhos de tela. O uso do Bootstrap permitiu que o desenvolvimento da interface fosse mais ágil e que a aplicação tivesse um design limpo, moderno e funcional, especialmente focado na usabilidade em dispositivos móveis.

## 2.5 VIACEP – API de Consulta de CEP

Em sistemas distribuídos, a integração entre componentes é essencial para garantir a funcionalidade do sistema de forma eficiente, escalável e confiável. No contexto do desenvolvimento de uma aplicação para abertura de denúncias ambientais, a integração de diferentes serviços, como uma API de consulta de CEP, é um exemplo claro de como a comunicação entre sistemas distintos pode ser utilizada para melhorar a experiência do usuário e otimizar processos.

A API VIACEP foi escolhida para permitir a consulta e preenchimento automático de dados de endereço com base no CEP informado pelo usuário. Em um sistema distribuído, a VIACEP se destaca como uma solução de microserviço que

oferece uma funcionalidade específica (consulta de CEP) sem sobrecarregar o sistema principal, facilitando a escalabilidade e a modularidade do projeto.

**Funcionamento da API VIACEP em um Sistema Distribuído:**

A API VIACEP é um serviço externo, o que a torna parte de um sistema distribuído. A comunicação entre a aplicação de denúncias e o serviço VIACEP ocorre por meio de requisições HTTP, utilizando o protocolo RESTful, o que é uma característica comum em sistemas distribuídos. A aplicação envia uma requisição contendo o CEP desejado e recebe uma resposta contendo os dados do endereço, que é então processada e exibida para o usuário.

Em um sistema distribuído, os serviços geralmente são independentes, podendo ser mantidos, escalados e gerenciados de forma autônoma. No caso da VIACEP, ela atua como um microserviço que não depende do código da aplicação principal, permitindo que a funcionalidade de consulta de CEP seja utilizada por outros sistemas ou interfaces, sem necessidade de integração direta com o banco de dados local de endereços.

**Benefícios no Contexto de Sistemas Distribuídos:**

1. **Desacoplamento e Escalabilidade**: A integração com a VIACEP permite que o sistema de denúncias se concentre em suas funcionalidades principais, como o registro de denúncias e a comunicação com os usuários, enquanto o serviço de consulta de CEP é isolado em seu próprio domínio. Isso permite que a aplicação principal seja mais leve e escalável, pois o processo de consulta ao CEP não impacta diretamente as operações do sistema de denúncias, como o envio de dados para o banco de dados ou a geração de relatórios.

2. **Redundância e Alta Disponibilidade**: Em um sistema distribuído, o uso de APIs externas como a VIACEP pode ser vantajoso devido à alta disponibilidade do serviço. Como o serviço de consulta de CEP é gerido

por uma infraestrutura externa, ele está frequentemente configurado para garantir uma boa performance e redundância, o que reduz a chance de falhas e melhora a confiabilidade da aplicação como um todo.

3. **Eficiência e Desempenho**: Em dispositivos móveis, onde a conexão de internet pode ser instável ou limitada, a utilização de um microserviço como a VIACEP contribui para a eficiência do sistema. A resposta rápida da API VIACEP, com dados de CEP precisos e formatados, permite que o preenchimento de formulários seja ágil e sem erros, otimizando o tempo de uso do aplicativo e melhorando a experiência do usuário.

4. **Manutenção Simplificada**: Como a VIACEP é um serviço externo, não há necessidade de realizar atualizações ou manutenção no código da aplicação principal para garantir a funcionalidade de consulta de endereços. Caso haja alterações no serviço da VIACEP, o sistema principal poderá se adaptar com poucas modificações, mantendo o sistema distribuído modular e de fácil manutenção.

**Implementação da API no Sistema de Denúncias:**

A API VIACEP foi implementada de forma assíncrona no sistema, permitindo que, ao digitar um CEP, o sistema consulte automaticamente os dados de endereço sem recarregar a página. Isso torna o preenchimento do formulário mais rápido e preciso. A consulta ao CEP é realizada por um serviço separado, o que facilita a manutenção e escalabilidade do sistema, sem impactar outras funcionalidades.

Ao integrar a VIACEP, o sistema ganha eficiência e modularidade, oferecendo uma experiência de usuário intuitiva, especialmente em dispositivos móveis, ao preencher automaticamente os campos de endereço com base no CEP. Isso contribui para a melhoria da qualidade e da satisfação do usuário no processo de denúncia.

## 2.6 Microserviço para Envio de E-mails Automáticos

A utilização de um microserviço para o envio de e-mails automáticos oferece uma solução eficiente e escalável, permitindo que o envio de notificações sobre as denúncias seja realizado de forma independente e sem sobrecarregar o sistema principal. No caso do nosso sistema, esse microserviço foi desenvolvido em PHP, utilizando a biblioteca PHPMailer, e foi configurado para operar em um ambiente distribuído, separado do código principal da aplicação web.

Quando o representante da prefeitura fecha uma denúncia ou atualiza o status de uma denúncia, o sistema envia um payload contendo as variáveis necessárias para a atualização do estado da denúncia para o microserviço. O microserviço, que pode ser configurado para se comunicar com qualquer endpoint, no nosso caso, está rodando localmente em uma porta separada, especificamente na porta 8000, permitindo que o envio de e-mails aconteça de forma isolada, sem interferir nas outras funcionalidades da aplicação.

Esse modelo de microserviço oferece diversas vantagens em sistemas distribuídos, como:

1. **Desacoplamento e Independência**: O envio de e-mails ocorre de forma independente, o que torna o sistema mais modular e facilita a manutenção e escalabilidade. Isso significa que, caso o microserviço precise ser atualizado ou escalado, isso pode ser feito sem impactar diretamente o funcionamento da aplicação web principal.

2. **Escalabilidade e Flexibilidade**: Por estar desacoplado da aplicação principal, o microserviço pode ser facilmente escalado de acordo com a demanda, garantindo que o envio de e-mails não afete o desempenho do sistema principal, especialmente em períodos de grande volume de atualizações.

3. **Desempenho e Eficiência**: O envio de e-mails é feito de maneira assíncrona, ou seja, não bloqueia outras operações do sistema. O código PHP envia os dados para o microserviço, que processa e envia os e-mails em segundo plano, sem afetar a experiência do usuário.

4. **Experiência do Usuário**: O microserviço é projetado para oferecer uma experiência de usuário moderna e fluida. O design dos e-mails enviados é cuidadosamente desenvolvido para ser visualmente atraente e intuitivo, transmitindo uma sensação de profissionalismo e cuidado com o usuário, além de ser responsivo, garantindo boa aparência tanto em dispositivos móveis quanto em desktops.

Com a arquitetura de microserviços, nosso sistema consegue manter o foco em suas funções principais, enquanto delega tarefas específicas, como o envio de e-mails, a um serviço especializado, garantindo alta performance e uma melhor experiência para os usuários.

## 2.7 Banco de Dados MySQL em Servidor Separado

Em sistemas distribuídos, é comum a utilização de múltiplos servidores para separar os diferentes componentes da aplicação, incluindo a separação entre a aplicação web e o banco de dados. Neste contexto, a decisão de manter o MySQL em um servidor diferente do servidor que hospeda a aplicação web pode trazer diversos benefícios, especialmente em termos de escalabilidade, desempenho e segurança. Embora essa abordagem introduza algumas complexidades, ela é altamente vantajosa em sistemas que demandam alta disponibilidade e capacidade de escalar conforme o volume de dados e usuários cresce.

### Vantagens da Separação entre a Aplicação Web e o Banco de Dados:

A principal vantagem de se utilizar o MySQL em um servidor distinto está na escalabilidade. Com a separação dos componentes, o banco de dados pode ser dimensionado de forma independente, sem impactar a performance da aplicação web. Em um sistema distribuído, onde há uma constante troca de informações entre diferentes partes da aplicação, essa separação pode reduzir o risco de sobrecarga no servidor da aplicação, permitindo que cada serviço seja otimizado e dimensionado conforme sua necessidade.

Outra vantagem significativa é a segurança. Ao manter o banco de dados em um servidor separado, é possível implementar estratégias de isolamento de rede, criptografia de dados e controle de acesso mais rigoroso. O banco de dados se torna menos suscetível a ataques, já que a comunicação entre a aplicação web e o banco de dados é realizada através de uma rede controlada, com autenticação e autorização adequadas.

Além disso, a alta disponibilidade é facilitada em sistemas distribuídos com servidores separados. O banco de dados pode ser configurado com replicação de dados, permitindo que os dados sejam sincronizados entre diferentes instâncias, o que aumenta a resiliência e garante que a aplicação continue operando mesmo em caso de falhas em uma das instâncias do banco de dados.

**Desafios e Considerações:**

Apesar das vantagens, o uso de um banco de dados MySQL em servidor separado também apresenta alguns desafios. Um dos principais é a latência nas comunicações entre a aplicação web e o banco de dados. Como os dados precisam ser transferidos através da rede, a distância física entre os servidores ou problemas de rede podem impactar a velocidade de resposta da aplicação. Esse fator deve ser monitorado, especialmente em sistemas com grandes volumes de dados ou alto tráfego.

Além disso, a complexidade na configuração e monitoramento do sistema aumenta. A separação entre os servidores exige uma infraestrutura mais elaborada, com a necessidade de um gerenciamento contínuo dos recursos, bem como a implementação de ferramentas de monitoramento e automação para garantir o bom funcionamento do sistema.

O custo operacional também pode ser um fator a ser considerado. Manter servidores separados implica em um aumento nos custos com infraestrutura e recursos de TI (Tecnologia da Informação). No entanto, esse custo é frequentemente justificado pela necessidade de garantir a escalabilidade e a performance de um sistema distribuído, especialmente em ambientes de produção com alta demanda.

A utilização de um banco de dados MySQL em um servidor separado é uma prática comum em sistemas distribuídos, especialmente em aplicações que exigem

alta escalabilidade e disponibilidade. Ao separar o banco de dados da aplicação web, é possível otimizar o desempenho, aumentar a segurança e garantir a resiliência do sistema. No entanto, essa abordagem deve ser adotada com cautela, considerando as necessidades específicas do sistema, o volume de dados e as exigências de performance. A decisão de usar servidores separados deve ser fundamentada na análise dos benefícios e desafios que essa configuração pode trazer para o projeto.

O arquivo de exportação dump do banco de dados será disponibilizado para testes, caso necessário. O login de **admin** é **root** com senha **123**, e o usuário normal é **user** com senha **123**.

No caso da aplicação atual, o banco de dados está hospedado no mesmo servidor da aplicação, dado que se trata de um ambiente de desenvolvimento e teste. Contudo, à medida que a aplicação evoluir, a separação entre o banco de dados e o servidor da aplicação poderá ser considerada para otimizar a escalabilidade e o desempenho.

## 2.8 Criptografia e Segurança de Dados

No contexto da aplicação, a segurança dos dados sensíveis, como senhas de usuários, é uma prioridade. Para garantir a proteção dessas informações, utilizamos a função de hashing do PHP, especificamente a função password_hash(), que gera um valor criptografado irreversível a partir da senha fornecida pelo usuário. Isso significa que, mesmo que o banco de dados seja acessado de maneira indevida, as senhas armazenadas estarão protegidas.

A função password_hash() aplica um algoritmo seguro (como o BCRYPT) para transformar a senha em um valor único e de difícil reversão. Para verificar a autenticidade de uma senha fornecida durante o login, utilizamos a função password_verify(), que compara a senha informada com o hash armazenado no banco de dados, sem nunca armazenar a senha original.

Essa abordagem garante que as senhas dos usuários sejam tratadas de maneira segura, mesmo no caso de um possível comprometimento do banco de dados.

# 3 PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

O desenvolvimento da aplicação de denúncias ambientais será realizado com o objetivo de criar uma interface intuitiva e acessível, otimizada para dispositivos móveis. A seguir, apresentamos os principais elementos e ferramentas que serão utilizadas no desenvolvimento, bem como a arquitetura do sistema, estratégias de integração e aspectos relacionados à segurança, escalabilidade e usabilidade da aplicação.

## 3.1 Tecnologias Fundamentais

A escolha das tecnologias para o desenvolvimento da aplicação é crucial para garantir que o sistema atenda aos requisitos funcionais e não funcionais de forma eficiente e eficaz. A seguir, detalhamos as tecnologias principais que serão utilizadas.

**HTML (Hypertext Markup Language):**

O HTML será utilizado como a base estrutural da aplicação, permitindo a criação de páginas web. Utilizaremos as últimas versões da linguagem para aproveitar recursos como elementos semânticos (como <header>, <footer>, <article>, <section>), que melhoram a acessibilidade e a manutenção do código. A semântica do HTML ajuda não só na organização do conteúdo, mas também no desempenho da página, proporcionando uma melhor leitura e indexação por motores de busca (SEO).

Além disso, serão utilizadas tags de formulários modernas e eficientes, como <input>, <select>, e <textarea>, para garantir uma boa experiência de usuário, especialmente em dispositivos móveis. Cada formulário será projetado de forma a ser intuitivo, com validações simples e feedback imediato.

**CSS (Cascading Style Sheets):**

O CSS será responsável pela parte visual da aplicação. Utilizaremos o Bootstrap para garantir a responsividade geral da aplicação, aproveitando seus componentes e o grid system, que já oferecem suporte nativo para layouts

responsivos. No entanto, para melhorar a experiência visual e adaptar o design a necessidades específicas, utilizaremos CSS personalizado.

A aplicação fará uso de media queries para garantir que o layout se ajuste adequadamente a diferentes tamanhos de tela, especialmente para dispositivos móveis. A crescente utilização de smartphones e tablets exige que a aplicação seja otimizada para telas menores. Através de ajustes finos com CSS, garantiremos que todos os elementos, como menus, tabelas e formulários, sejam redimensionados de forma eficiente, proporcionando uma visualização confortável em dispositivos com diferentes resoluções.

Além do uso de media queries, a aparência será modernizada com a utilização de propriedades avançadas do CSS, como Flexbox e Grid. Essas tecnologias permitirão o posicionamento eficiente de elementos, criando layouts fluidos que se adaptam de forma dinâmica ao espaço disponível na tela. Isso elimina a necessidade de manipulações excessivas com floats ou posicionamento absoluto, garantindo uma estrutura mais limpa e de fácil manutenção.

Embora o Bootstrap forneça uma base sólida, algumas áreas específicas da interface exigem ajustes personalizados para melhorar a aparência e a usabilidade. Por exemplo, cards, tabelas e menus receberão estilos adicionais para se alinharem melhor ao visual da aplicação e à identidade do projeto. O uso de sombras sutis, transições e animações em elementos interativos garantirá que a interface seja não apenas funcional, mas também atraente.

A combinação do Bootstrap com o CSS customizado resultará em uma interface de usuário fluida, moderna e altamente responsiva, melhorando a experiência geral do usuário em diferentes dispositivos e ambientes.

**JavaScript:**

O JavaScript será essencial para adicionar interatividade à aplicação. Ele será utilizado para manipular o DOM (Document Object Model), permitindo que os formulários sejam validados em tempo real e que dados sejam enviados e recebidos do servidor sem a necessidade de recarregar a página, utilizando AJAX

(Asynchronous JavaScript and XML). Isso garantirá uma navegação mais rápida e uma experiência de uso mais fluida.

Além disso, o JavaScript será responsável pela integração com a API de CEP, permitindo que os campos de endereço sejam preenchidos automaticamente com base no CEP informado pelo usuário. A comunicação com a API será realizada por meio de requisições HTTP, utilizando o método fetch() para garantir um processo rápido e assíncrono, sem que o usuário precise esperar pela resposta do servidor.

**Bootstrap:**

O Bootstrap será uma das principais ferramentas para garantir que a aplicação tenha uma interface visual atraente e responsiva desde o início. Como um dos frameworks front-end mais populares e amplamente utilizados, o Bootstrap oferece uma ampla gama de componentes prontos, como botões, barras de navegação, formulários, modais e muito mais. Além de permitir um design consistente e funcional, o uso de classes predefinidas do Bootstrap acelera o desenvolvimento, permitindo que a interface tenha uma aparência moderna e otimizada para uma grande variedade de dispositivos, especialmente móveis.

A aplicação será desenvolvida seguindo o conceito **mobile first**, que visa garantir que a experiência do usuário seja priorizada em dispositivos móveis, com uma adaptação natural para telas maiores. O Bootstrap se alinha perfeitamente a essa abordagem, pois já possui um sistema de grid flexível que ajusta automaticamente os layouts para diferentes tamanhos de tela. Dessa forma, o design é projetado para ser funcional e esteticamente agradável em smartphones e tablets desde o início, com a ampliação para telas maiores (como desktops) tratada em etapas posteriores com o uso de media queries.

Com essa estratégia, a aplicação oferecerá uma experiência de navegação fluida e eficiente, independentemente do dispositivo utilizado, garantindo que os usuários possam interagir com a interface de maneira intuitiva, rápida e sem frustrações. Além disso, o Bootstrap oferece suporte para temas claros e escuros, o que permite uma maior personalização de acordo com o público-alvo da aplicação, podendo ser ajustado conforme a preferência dos usuários.

Embora o Bootstrap forneça uma base sólida, a personalização será aplicada onde necessário, como em cards, tabelas e menus, para garantir que a identidade visual da aplicação seja mantida. O uso de tecnologias modernas como Flexbox e CSS Grid também será adotado para ajustar o layout de maneira fluida, criando uma experiência mais envolvente e visualmente agradável, sem a necessidade de manipulações complexas de CSS.

**PHP: Controle de Acesso e Segurança nas Operações de Banco de Dados**

O PHP desempenha um papel crucial no back-end da aplicação, sendo responsável por gerenciar as operações de acesso, autenticação de usuários e interação com o banco de dados. Através de suas funcionalidades robustas, o PHP garante que a aplicação tenha um controle de acesso eficiente e que as operações realizadas no banco de dados sejam seguras e confiáveis.

**Controle de Acesso:**

A aplicação utiliza o PHP para gerenciar o controle de acesso, tanto para usuários comuns quanto para administradores. Ao acessar o sistema, os usuários devem passar por um processo de autenticação, no qual suas credenciais são verificadas antes de permitir o acesso. Para isso, o PHP utiliza sessões, garantindo que, uma vez autenticado, o usuário tenha acesso apenas às funcionalidades pertinentes ao seu perfil.

As credenciais dos usuários, como nome de usuário e senha, são processadas com segurança. As senhas são armazenadas de forma criptografada no banco de dados utilizando a função password_hash(), garantindo que não possam ser recuperadas diretamente em caso de violação de segurança. No momento do login, o PHP utiliza a função password_verify() para comparar a senha fornecida com a versão criptografada armazenada, validando assim o acesso do usuário.

**Operações de Banco de Dados (CRUD):**

As operações de criação, leitura, atualização e exclusão (CRUD) são realizadas através do PHP, que interage diretamente com o banco de dados MySQL. O PHP é responsável por receber as entradas dos usuários (como dados dos formulários), processá-las e enviar as consultas SQL ao banco de dados.

Uma das principais preocupações ao lidar com dados enviados pelo usuário é a segurança. Para proteger a aplicação contra ataques como o SQL Injection, o PHP utiliza prepared statements. Com essa técnica, as consultas SQL são preparadas e os dados fornecidos pelo usuário são vinculados a parâmetros previamente definidos, impedindo que qualquer entrada maliciosa seja executada como parte da consulta SQL. Isso assegura que os dados fornecidos pelo usuário sejam tratados apenas como valores, sem alterar a estrutura da consulta.

Além disso, o PHP realiza a validação dos dados antes de enviá-los para o banco de dados, verificando se estão no formato esperado (como validação de e-mail, números de telefone ou categorias de denúncia), minimizando o risco de falhas ou inconsistências. Essa camada de validação é fundamental para garantir que os dados armazenados no banco de dados sejam sempre válidos e seguros.

**Segurança e Boas Práticas:**

A segurança é uma prioridade no desenvolvimento da aplicação. Além do uso de prepared statements, o PHP também é configurado para proteger contra outras ameaças comuns, como Cross-Site Scripting (XSS). A validação de entrada e a filtragem de dados são implementadas para garantir que qualquer dado fornecido pelo usuário seja seguro antes de ser exibido na interface ou armazenado no banco de dados.

Para reforçar a segurança da comunicação com o banco de dados, as credenciais de acesso são armazenadas de forma segura, e a conexão com o MySQL é feita utilizando métodos que evitam o envio de dados sensíveis sem a devida proteção.

## 3.2 Integração com APIs e Microserviços

A integração com APIs e microserviços é uma parte fundamental da aplicação, permitindo a comunicação com sistemas externos e a divisão de responsabilidades, o que melhora a escalabilidade e a flexibilidade do sistema.

**VIACEP - API de Consulta de CEP:**

A API VIACEP será utilizada para preencher automaticamente os campos de endereço com base no CEP inserido pelo usuário. Esse recurso tornará o processo de denúncia mais rápido e conveniente, evitando que o usuário tenha que preencher manualmente todos os campos de endereço. A integração será feita utilizando requisições HTTP, de forma assíncrona, para não interromper a experiência do usuário.

A resposta da API será processada e os campos de endereço serão preenchidos automaticamente na interface da aplicação, com validações adicionais para garantir que o CEP seja válido e que as informações fornecidas pela API sejam corretas.

**Microserviço para Envio de E-mails Automáticos:**

O envio de e-mails automáticos será realizado por meio de um microserviço dedicado, desenvolvido em PHP com a biblioteca PHPMailer. Este microserviço será responsável por enviar confirmações de recebimento das denúncias para os usuários, bem como notificações para os administradores do sistema.

A separação do envio de e-mails em um microserviço tem várias vantagens, incluindo a possibilidade de escalabilidade independente e maior flexibilidade na implementação de melhorias ou ajustes na parte de envio de e-mails, sem afetar o restante da aplicação. O microserviço será hospedado separadamente da aplicação principal, garantindo que o desempenho da aplicação não seja impactado pelas requisições de envio de e-mail.

## 3.3 Banco de Dados

A escolha do banco de dados é uma parte crucial para garantir a integridade e a segurança das informações armazenadas. A seguir, detalhamos a arquitetura do banco de dados e como ele será projetado.

**Estrutura do Banco de Dados:**

A estrutura do banco de dados é composta por três tabelas principais: users, tickets e comments. A tabela users armazenará informações dos usuários, como nome, e-mail e senha (devidamente criptografada). A tabela tickets será responsável por armazenar os dados das denúncias, incluindo categoria, descrição, prioridade e status, além de informações sobre o endereço (como o CEP). A tabela comments ainda está em desenvolvimento, mas sua finalidade será registrar as mensagens trocadas entre os cidadãos e os representantes da prefeitura, possibilitando o acompanhamento das interações de forma eficiente.

**Relacionamento entre as Tabelas:**

As tabelas serão interligadas por chaves estrangeiras, garantindo a integridade referencial entre os dados. A tabela tickets estará relacionada à tabela users por meio de uma chave estrangeira que indicará o usuário que fez a denúncia. Já a tabela comments estará ligada a tickets e users, permitindo que as mensagens de acompanhamento sejam associadas a denúncias específicas. Essa estrutura de relacionamento visa organizar os dados de maneira eficiente e facilitar a consulta e o gerenciamento das informações.

**Segurança dos Dados:**

A segurança dos dados é uma prioridade na construção da aplicação. Para proteger as senhas dos usuários, será utilizada a função password_hash() do PHP para criptografá-las antes de armazená-las no banco de dados. Dessa forma, mesmo em um cenário de ataque, as senhas estarão protegidas contra recuperação.

As informações trocadas entre a aplicação e o banco de dados serão manipuladas de forma segura, com foco na integridade e proteção dos dados sensíveis. A aplicação

adotará boas práticas de segurança na camada de código e banco de dados, garantindo que a comunicação entre o cliente e o servidor seja realizada de maneira eficiente e protegida.

## 3.4 Implementação e Escalabilidade

A aplicação será projetada para ser escalável desde o início. A utilização de microserviços, por exemplo, permite que partes específicas da aplicação, como o envio de e-mails, possam ser escaladas de forma independente, sem afetar a performance do restante do sistema.

À medida que a aplicação evoluir, consideraremos a utilização de containers Docker para facilitar a implementação de ambientes consistentes e a orquestração com Kubernetes, permitindo que o sistema seja facilmente dimensionado em um ambiente de produção.

## 3.5 Ferramentas de Desenvolvimento

**Controle de Versão com Git:**

O Git será utilizado como sistema de controle de versão, permitindo o versionamento eficiente do código-fonte. O uso de branches permitirá o desenvolvimento paralelo de novas funcionalidades, enquanto o uso de pull requests garantirá que o código seja revisado antes de ser integrado ao repositório principal. O código será hospedado em uma plataforma de repositório remoto, como GitHub ou GitLab, para facilitar a colaboração entre desenvolvedores e garantir que o histórico do projeto seja bem documentado.

## 3.6 Testes e Monitoramento

A aplicação será sujeita a um conjunto abrangente de testes para garantir sua confiabilidade e desempenho. Testes unitários serão realizados para garantir que cada componente individual da aplicação funcione corretamente. Além disso, serão

realizados testes de integração para verificar a interação entre diferentes partes do sistema, como a comunicação com a API de CEP e o envio de e-mails.

## 3.7 Conclusão

Este plano de desenvolvimento visa criar uma aplicação robusta, escalável e segura, com foco em uma experiência de usuário otimizada para dispositivos móveis. A utilização de tecnologias modernas e abordagens como microserviços permitirá que a aplicação evolua de forma modular e eficiente, enquanto as práticas de segurança garantirão que os dados sensíveis dos usuários sejam protegidos. A aplicação será projetada para atender às necessidades da prefeitura, proporcionando uma solução eficiente para o gerenciamento de denúncias de irregularidades ambientais.

# 4 ESTRUTURA DO PROGRAMA

A aplicação de denúncias de irregularidades ambientais para a prefeitura de São Paulo é organizada de forma a separar claramente suas funcionalidades, arquivos de configuração e recursos visuais. A estrutura principal do projeto está contida na pasta htdocs, que é dividida da seguinte forma:

## 4.1 Estrutura de Código da Aplicação Web

A aplicação web é responsável por fornecer a interface que permite aos usuários submeterem e acompanharem denúncias de irregularidades ambientais. A estrutura do código da aplicação web é organizada da seguinte forma:

```
htdocs
│
├── admin-area
│   ├── admin-home.php
│   ├── admin-nav.php
│   ├── admin-open-tickets-full.php
│   ├── admin-opened-tickets.php
│   ├── admin-ticket-closed-details.php
│   ├── admin-ticket-history.php
│   ├── admin-ticket-info-ro.php
│   ├── buttons-open-closed-admin.php
│   ├── button-create-user.php
│   ├── create-user.php
│   ├── mailer-closed-ticket.php
│   ├── mailer-create-user.php
│   ├── receive-post.php
│   ├── ticket-details-admin.php
│   └── ticket-info-ro.php
│
├── assets
```

```
│   ├── css
│   │   ├── login.css
│   │   └── tickets-table.css
│   │
│   ├── images
│   │   ├── favicon
│   │   └── log.svg
│   │
│   └── js
│       └── login.js
│
├── config
│   ├── variables
│   ├── card_data.php
│   ├── connection.php
│   ├── save-edit.php
│   └── send-post.php
│
├── includes
│   ├── admin-nav.php
│   ├── admin-open-tickets-full.php
│   ├── admin-opened-tickets.php
│   ├── admin-ticket-closed-details.php
│   ├── button-create-user.php
│   ├── buttons-open-closed-admin.php
│   ├── create-user.php
│   ├── hat.php
│   ├── mailer-closed-ticket.php
│   ├── mailer-create-user.php
│   ├── receive-post.php
│   ├── ticket-details-admin.php
│   ├── ticket-info-ro.php
│   ├── admin-home.php
```

```
│   ├── admin-open-tickets.php
│   ├── admin-ticket-history.php
│   ├── admin-ticket-info-ro.php
│   └── create-user.php
│
├── node_modules
│
├── pages
│   ├── home.php
│   ├── open-ticket.php
│   ├── ticket-info.php
│   ├── tickets-history.php
│   └── tickets-home.php
│
├── vendor
│
├── composer.json
├── composer.lock
├── index.php
├── package-lock.json
└── package.json
```

A pasta *admin-area* contém todas as funcionalidades voltadas para os administradores, incluindo a gestão de tickets e a criação de usuários. Os outros diretórios, como *assets*, armazenam dados importantes, como arquivos de estilo e scripts necessários para a interface do usuário.

A pasta *config* contém arquivos de configuração que devem ser mantidos em diretórios privados para evitar acesso direto pelo navegador. Essa separação de segurança é aplicada no momento da hospedagem da aplicação em um servidor web, garantindo que apenas arquivos necessários sejam expostos.

A pasta includes abriga componentes reutilizáveis e funções auxiliares que suportam tanto a parte administrativa quanto a parte dos usuários, otimizando a

manutenção e a consistência do código. As *pages* representam a parte da interface voltada para os usuários finais, permitindo navegação e interação com o sistema.

## 4.2 Estrutura de Código do Microserviço de Envio de E-mail

O microserviço é responsável pelo envio de e-mails de confirmação das denúncias submetidas. Desenvolvido em PHP, ele foi projetado para garantir a integração eficiente com a aplicação principal. A escolha do PHP para esse microserviço foi motivada pela necessidade de manter a coesão e a simplicidade da estrutura de código, aproveitando as funcionalidades nativas da linguagem. Para otimizar o envio de mensagens e garantir maior flexibilidade, a parte de envio de e-mails utiliza a biblioteca PHPMailer, permitindo o controle detalhado sobre o processo de envio e autenticação."

```
htdocs
│
├── config
│   └── smtp-data.php
│
├── mailer
│   └── send-email.php
│
└── receive-post.php
```

Com essa estrutura, o microserviço foi desenvolvido de forma modular e eficiente, garantindo a separação clara de responsabilidades entre as configurações, funções de envio de e-mails e o ponto de entrada para as requisições. A pasta *config* contém as informações sensíveis de configuração do servidor SMTP, enquanto o diretório *mailer* centraliza a lógica do envio de e-mails. O arquivo *receive-post.php* serve como o ponto de integração, recebendo as requisições e acionando as funções necessárias para o envio das mensagens. Essa organização facilita a manutenção e a escalabilidade do sistema, permitindo futuras melhorias ou a inclusão de novos serviços relacionados ao envio de notificações.

# 5 RELATÓRIO COM LINHAS DE CÓDIGO

Devido à limitação de páginas estabelecida para este trabalho, optou-se por apresentar apenas os trechos mais relevantes do código que implementa o microserviço de envio de e-mails. Essas partes do código foram selecionadas com o objetivo de ilustrar os principais processos envolvidos, como a configuração do servidor SMTP e a função de envio de e-mails. As demais funções e detalhes complementares, embora essenciais para o funcionamento do sistema, foram omitidos do corpo principal do relatório para manter a concisão e o foco nos pontos mais significativos.

## 5.1 Front-End Modular

A aplicação web foi projetada com foco na modularidade do front-end, permitindo fácil manutenção e expansão. A imagem a seguir mostra a estrutura de um dos componentes principais do front-end, destacando a divisão da interface em partes reutilizáveis, como cabeçalhos, formulários e botões de interação.

*Figura 1 - Padrão de codificação adotado no projeto*

```php
<?php

include('../includes/protect.php');

$userid = $_SESSION["userid"];

?>

<!doctype html>
<html lang="pt-br">

<head>
    <title>Denúncias | Prefeitura de São Paulo</title>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta name="viewport" content=
"width=device-width, initial-scale=1, shrink-to-fit=no" />

    <!-- Bootstrap CSS -->
    <link href="../node_modules/bootstrap/dist/css/bootstrap.min.css"
rel="stylesheet" />
    <link rel="stylesheet" href="../assets/css/index.css">
    <!-- Bootstrap Icons CSS -->
    <link href=
"../node_modules/bootstrap-icons/font/bootstrap-icons.css" rel=
"stylesheet" />
</head>

<body>

    <div class="d-flex flex-column wrapper">

        <!-- Include Navbar -->
        <?php include '../includes/navbar.php'; ?>

        <main class="flex-fill">

            <!-- Include Cards -->
            <?php include '../includes/card.php'; ?>

        </main>

        <!-- Include Footer -->
        <?php include '../includes/footer.php'; ?>

    </div>

    <script src=
"../node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>
```

A imagem apresentada nesta seção ilustra uma das páginas da aplicação de denúncias, desenvolvida com foco na modularidade do front-end. A modularidade permite que os componentes da interface sejam reutilizados em diferentes partes da aplicação, facilitando a manutenção e a escalabilidade do sistema.

A estrutura da página é baseada em um layout flexível, utilizando o framework Bootstrap para garantir uma interface responsiva e adaptável a diferentes dispositivos. O código exibido é um exemplo da página inicial de denúncias, onde são integrados componentes essenciais da aplicação, como a barra de navegação, o conteúdo principal e o rodapé. Esses componentes são modularizados e incluídos por meio de arquivos PHP, como *navbar.php*, *card.php* e *footer.php*. Essa abordagem modular permite que alterações em qualquer parte da interface sejam feitas de forma centralizada, sem a necessidade de modificar o código de cada página individualmente.

Além disso, o uso da variável de sessão *$_SESSION["userid"]* no início do código reflete a lógica de controle de acesso, garantindo que apenas usuários autenticados possam visualizar ou interagir com as páginas da aplicação. Isso é importante para assegurar a segurança e a privacidade dos dados, pois a aplicação verifica se o usuário está logado antes de carregar os componentes da interface.

## 5.2 Cartão Modular

A imagem apresentada nesta seção mostra um exemplo de um “card” utilizado na aplicação de denúncias. O card é um componente modular que exibe informações de forma compacta e visualmente atraente, sendo uma das principais unidades de conteúdo na interface. Este tipo de estrutura é altamente reutilizável, permitindo que a mesma formatação e estilo sejam aplicados em diferentes partes da aplicação, promovendo consistência no design e facilitando a manutenção do sistema.

*Figura 2 - Estrutura HTML e CSS do componente de card*

```
<style>
    * {
        box-sizing: border-box;
    }

    .icon {
        color: black;
        font-size: 2rem;
    }

    .card-text {
        color: gray;
    }

    .card:hover {
        background-color: #AC1B2F;
        color: #ffffff;
    }

    .card:hover .icon,
    .card:hover .card-text {
        color: #ffffff;
    }

    .card {
        border-radius: 5px !important;

        transition: all 0.5s;
        overflow: hidden;
        padding-bottom: 20px;
    }
</style>

<div class="container p-5">
    <div class="row justify-content-center">
        <!-- CARD 1 -->
        <div class=
"col-12 col-sm-6 col-md-6 col-lg-5 col-xl-4 col-xxl-4 p-2">
            <a href="../pages/open-ticket.php" class=
"text-decoration-none">
                <div class=
"card text-center border-0 shadow rounded-0 p-1">
                    <div class="icon">
                        <i class="bi bi-person-raised-hand"></i>
                    </div>
                    <div class="card-body">
                        <h4 class="card-title fw-bold">Denúnciar</h4>
                        <p class="card-text">
Denuncie irregularidades ambientais para a Prefeitura de São Paulo.</p>
                        <link rel="stylesheet"
                            href=
"https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icon
s.min.css"
>
                    </div>
                </div>
            </a>
        </div>

    </div>
```

Este card, em particular, tem como objetivo permitir ao usuário submeter uma denúncia. Ele contém um ícone, um título e uma descrição, proporcionando um acesso rápido à página de registro de irregularidades ambientais. Essa estrutura modular facilita a manutenção e a expansão da aplicação, visto que os cards podem ser facilmente atualizados ou duplicados sem a necessidade de mudanças significativas no código.

**Comportamento Dinâmico com JavaScript:**

Para garantir que todos os cards na página tenham a mesma altura, mesmo quando seus conteúdos variam, um código JavaScript é utilizado. Este código, também presente no mesmo arquivo, calcula dinamicamente a altura máxima dos cards e aplica essa altura uniformemente a todos os cards da página. Isso assegura que a disposição dos elementos seja visualmente consistente e esteticamente agradável, independentemente do conteúdo.

O código JavaScript que realiza essa função é o seguinte:

*Figura 3 - JavaScript aplicado para dinamismo e interatividade no componente de card*

```
<script>
    window.addEventListener('load', function () {
        const cards = document.querySelectorAll('.card');
// Seleciona todos os elementos com a classe "card"
        let maxHeight = 0;

        // Calcula a altura máxima dos cards
        cards.forEach(card => {
            let cardHeight = card.offsetHeight;
            if (cardHeight > maxHeight) {
                maxHeight = cardHeight;
            }
        });

        // Aplica a altura máxima a todos os cards
        cards.forEach(card => {
            card.style.height = maxHeight + 'px';
        });
    });

    // Atualiza os cards quando a tela é redimensionada
    window.addEventListener('resize', function () {
        const cards = document.querySelectorAll('.card');
        let maxHeight = 0;

        // Remove a altura definida anteriormente
        cards.forEach(card => {
            card.style.height = 'auto';
        });

        // Recalcula a altura máxima
        cards.forEach(card => {
            let cardHeight = card.offsetHeight;
            if (cardHeight > maxHeight) {
                maxHeight = cardHeight;
            }
        });

        // Reaplica a altura máxima
        cards.forEach(card => {
            card.style.height = maxHeight + 'px';
        });
    });
</script>
```

Este código ajusta a altura de todos os cards da página para que possuam a mesma altura, independentemente do conteúdo exibido, melhorando a apresentação visual da interface.

### 5.3 Endpoint de Recebimento de Dados e Envio de E-mail

O endpoint apresentado é responsável por receber um payload em formato JSON enviado a partir de uma requisição HTTP. O objetivo principal deste endpoint é processar os dados recebidos, extrair as informações relevantes e, em seguida, acionar uma função que realiza o envio de um e-mail de notificação para o usuário. Este fluxo é crucial para a comunicação do sistema com o usuário, garantindo que ele seja informado sobre interações em suas solicitações.

O arquivo PHP, que processa o payload, executa os seguintes passos:

1. **Recebimento e Leitura do Payload**: O endpoint começa capturando os dados brutos enviados via método POST. Esses dados são extraídos utilizando a função *file_get_contents('php://input'),* que lê o corpo da requisição.

2. **Decodificação do JSON**: O conteúdo recebido em formato JSON é então decodificado utilizando a função *json_decode(),* convertendo-o em um *array* associativo PHP. Isso permite que as informações sejam facilmente acessadas e manipuladas dentro do código.

3. **Validação dos Dados**: Caso a decodificação do JSON seja bem-sucedida, os dados são extraídos do array e armazenados em variáveis. Caso contrário, uma resposta de erro é retornada com o código HTTP 400, indicando que o JSON recebido não é válido.

4. **Envio do E-mail**: Após a extração dos dados, o e-mail de notificação é enviado através da função *sendTicketClosedEmail()*. Esta função, que é chamada com os dados do ticket (como ID (protocolo), descrição,

justificativa de resolução, entre outros), gera e envia um e-mail informando o usuário sobre o status do seu ticket.

**Código do Endpoint:**

*Figura 4 - Código do endpoint responsável pela comunicação entre a aplicação e o microserviço*

```php
<?php

include_once('mailer-closed-ticket.php');

// Retrieve the raw POST data
$jsonData = file_get_contents('php://input');
print_r($jsonData);

// Decode the JSON data into a PHP associative array
$data = json_decode($jsonData, true);
error_log(print_r("No servidor, recebendo post, respondendo com : ".
$jsonData, TRUE));

// Check if decoding was successful
if ($data !== null) {
   // Access the data and perform operations

   $user_email = $data["user_email"];
   $id = $data["ticket_id"];
   $description = $data["description"];
   $resolution_justification = $data["resolution_justification"];
   $attendant = $data["attendant"];
   $status = $data["status"];

   sendTicketClosedEmail($user_email, $id, $description,
$resolution_justification, $attendant);

   // Perform further processing or respond to the request
} else {
   // JSON decoding failed
   http_response_code(400); // Bad Request
   echo "Invalid JSON data";
}
?>
```

## 5.4 Uso da API ViaCEP

O formulário foi projetado para permitir que os usuários insiram um CEP (Código de Endereçamento Postal) e, com isso, os campos de endereço, como logradouro, bairro, cidade e estado, sejam preenchidos automaticamente. Para isso, o código utiliza a API pública ViaCEP (https://viacep.com.br/), que oferece informações de endereço com base no CEP fornecido. O processo ocorre da seguinte maneira:

1. **Captura do CEP**: Quando o usuário insere o CEP no campo de entrada e sai do campo (evento onblur), a função *buscarCEP()* é chamada. A função começa por limpar qualquer caracter não numérico do CEP, utilizando uma expressão regular (/\D/g).

2. **Validação do CEP**: Se o CEP tem exatamente 8 caracteres (o número padrão de dígitos para CEPs no Brasil), o código continua com a solicitação para a API ViaCEP. Caso contrário, um alerta de "CEP inválido" é mostrado.

3. **Chamada para a API ViaCEP**: A função *fetch()* é usada para realizar uma requisição GET para a API ViaCEP, passando o CEP inserido na URL. A API retorna um objeto JSON com as informações do endereço associado ao CEP, como logradouro, bairro, localidade (cidade) e uf (estado). Se a API retornar dados válidos (sem erro), os campos de endereço (logradouro, bairro, cidade, estado) são automaticamente preenchidos com as informações obtidas. Caso a API retorne um erro, indicando que o CEP não foi encontrado, um alerta é exibido para o usuário.

4. **Preenchimento Automático dos Campos**: O código preenche os campos visíveis e invisíveis (campos escondidos) do formulário com os valores recebidos da API. Isso garante que o usuário tenha os campos automaticamente preenchidos, mas ainda assim possam ser usados

para o envio do formulário. O preenchimento é feito nos campos de logradouro, bairro, cidade e estado.

5. **Tratamento de Erros**: Caso haja algum erro na chamada da API ou no retorno dos dados, um erro é capturado com o método *catch()* e um alerta de erro é mostrado ao usuário.

**Código usando a API:**

*Figura 5 - Código do formulário que recebe o CEP*

```
<div class=
"container d-flex justify-content-center align-items-center pt-5">
    <form class="row g-3 needs-validation" method="POST" novalidate>

        <!-- Campos preenchidos automaticamente (Código resumido) -->
        <div class=
"col-12 col-sm-6 col-md-6 col-lg-3 col-xl-3 col-xxl-3 p-1">
            <div class="form-floating shadow">
                <input type="text" class="form-control" id="logradouro"
name="logradouro" placeholder="Logradouro"
                    readonly>
                <label for="logradouro">Logradouro</label>
            </div>
        </div>

        <div class=
"col-12 col-sm-6 col-md-6 col-lg-3 col-xl-3 col-xxl-3 p-1">
            <div class="form-floating shadow">
                <input type="text" class="form-control" id="bairro" name
="bairro" placeholder="Bairro" readonly>
                <label for="bairro">Bairro</label>
            </div>
        </div>

        <div class=
"col-12 col-sm-6 col-md-6 col-lg-3 col-xl-3 col-xxl-3 p-1">
            <div class="form-floating shadow">
                <input type="text" class="form-control" id="cidade" name
="cidade" placeholder="Cidade" readonly>
                <label for="cidade">Cidade</label>
            </div>
        </div>

        <div class=
"col-12 col-sm-6 col-md-6 col-lg-3 col-xl-3 col-xxl-3 p-1">
            <div class="form-floating shadow">
                <input type="text" class="form-control" id="estado" name
="estado" placeholder="Estado" readonly>
                <label for="estado">Estado</label>
            </div>
        </div>

<!-- Fim dos campos preenchidos automaticamente (Código resumido) -->
    </form>

</div>
```

*Figura 6 - Código de integração com a API ViaCEP*

```
<script>


// Atualiza os campos visíveis e escondidos no preenchimento automático
do CEP
    function buscarCEP() {
        const cep = document.getElementById('cep').value.replace(/\D/g,
'');

        if (cep.length === 8) {
            fetch(`https://viacep.com.br/ws/${cep}/json/`)
                .then(response => response.json())
                .then(data => {
                    if (!data.erro) {
                        // Preenche os campos visíveis
                        document.getElementById('logradouro').value =
data.logradouro;
                        document.getElementById('bairro').value = data.
bairro;
                        document.getElementById('cidade').value = data.
localidade;
                        document.getElementById('estado').value = data.
uf;

                        // Atualiza os campos escondidos
                        document.querySelector(
'input[name="logradouro"]').value = data.logradouro;
                        document.querySelector('input[name="bairro"]').
value = data.bairro;
                        document.querySelector('input[name="cidade"]').
value = data.localidade;
                        document.querySelector('input[name="estado"]').
value = data.uf;
                    } else {
                        alert('CEP não encontrado.');
                    }
                })
                .catch(error => {
                    console.error('Erro ao buscar o CEP:', error);
                    alert('Erro ao buscar o CEP');
                });
        } else {
            alert('CEP inválido.');
        }
    }

</script>
```

Este processo simplifica a experiência do usuário, pois ao inserir apenas o CEP, ele obtém automaticamente os dados completos de endereço, sem precisar digitá-los manualmente. Isso também melhora a precisão e a consistência dos dados inseridos.

# 6 CONCLUSÃO

A aplicação web desenvolvida, foi projetada com foco na experiência do usuário em dispositivos móveis, seguindo o modelo "mobile-first". A escolha desse modelo de desenvolvimento permitiu garantir uma interface intuitiva e adaptável, com o design sendo otimizado para telas menores e, ao mesmo tempo, funcional em dispositivos maiores.

Ao adotar o Bootstrap como framework CSS, conseguimos garantir uma estrutura responsiva, que se ajusta automaticamente a diferentes tamanhos de tela, sem comprometer a usabilidade. A utilização de PHP no backend possibilitou a implementação de funcionalidades dinâmicas, como o processamento de formulários e o gerenciamento de dados no banco de dados MySQL, garantindo a eficiência e a escalabilidade da aplicação.

A integração com SQL proporcionou um armazenamento seguro e eficiente de dados, permitindo o gerenciamento das informações dos usuários e das denúncias de irregularidades ambientais, com consultas rápidas e otimizadas. A aplicação oferece uma experiência fluida, tanto no envio quanto no recebimento de dados, garantindo que os usuários possam interagir facilmente com o sistema, independentemente do dispositivo utilizado.

Com a adoção do modelo "mobile-first", conseguimos priorizar a acessibilidade e a usabilidade, especialmente em um cenário onde o acesso a internet por meio de dispositivos móveis é cada vez mais predominante. A aplicação está preparada para atender de forma eficaz às necessidades dos usuários, oferecendo um ambiente seguro, confiável e fácil de usar.

# 7 REFERÊNCIAS

**Alura**. *Boas práticas no desenvolvimento de APIs RESTful*. Disponível em: https://www.alura.com.br/artigos/boas-praticas-no-desenvolvimento-de-apis-restful. Acessado em: 02 de out. de 2024.

**Alura**. *Boas práticas de desenvolvimento para front-end*. Disponível em: https://www.alura.com.br/artigos/boas-praticas-no-front-end. Acessado em: 07 de set. de 2024.

**Caelum**. *Curso de PHP para Iniciantes*. Disponível em: https://www.caelum.com.br/curso/php-para-iniciantes/. Acessado em: 13 de set. de 2024.

**Código Fonte**. *API ViaCEP em Aplicações Web*. Disponível em: https://www.codigofonte.com.br/artigos/api-viacep-em-aplicacoes-web. Acessado em: 15 de set. de 2024.

**DevMedia**. *Introdução ao JavaScript e suas funcionalidades*. Disponível em: https://www.devmedia.com.br/introducao-ao-javascript/28307. Acessado em: 28 de ago. de 2024.

**DevMedia**. *Manipulação de Data e Hora no PHP*. Disponível em: https://www.devmedia.com.br/manipulacao-de-data-e-hora-no-php/28458. Acessado em: 22 de set. de 2024.

**DigitalOcean.** *Como Criptografar Senhas em PHP com bcrypt*. Disponível em: https://www.digitalocean.com/community/tutorials/how-to-hash-passwords-in-php-with-bcrypt. Acessado em: 17 de set. de 2024.

**Google Developers**. *Melhores práticas de performance com JavaScript*. Disponível em: https://developers.google.com/web/fundamentals/performance. Acessado em: 30 de set. de 2024.

**Google Developers**. *Mobile-First Approach*. Disponível em: https://developers.google.com/web/fundamentals/design-and-ux/responsive. Acessado em: 12 de set. de 2024.

**JavaTpoint**. *Introdução ao JavaScript com exemplos*. Disponível em: https://www.javatpoint.com/javascript-tutorial. Acessado em: 05 de out. de 2024.
MDN Web Docs. *API Fetch e Promises*. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API/Using_Fetch. Acessado em: 29 de set. de 2024.

**MDN Web Docs**. *Introdução ao SQL*. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/SQL. Acessado em: 25 de ago. de 2024.

**MDN Web Docs**. *JavaScript Fetch API*. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API. Acessado em: 15 de ago. de 2024.

**Microsoft Docs**. *Introdução ao JavaScript*. Disponível em: https://learn.microsoft.com/pt-br/samples/azure/quickstart-javascript/. Acessado em: 10 de set. de 2024.

**MySQL**. *Documentação do MySQL*. Disponível em: https://dev.mysql.com/doc/. Acessado em: 05 de set. de 2024.

**PHP Manual**. *Documentação Oficial do PHP*. Disponível em: https://www.php.net/manual/pt_BR/. Acessado em: 20 de ago. de 2024.

**PHP Manual**. *Funções de Criptografia em PHP*. Disponível em: https://www.php.net/manual/pt_BR/book.openssl.php. Acessado em: 20 de set. de 2024.

**Rocketseat.** *Aula de Formatação de Input e Validação com JavaScript*. Disponível em: https://rocketseat.com.br. Acessado em: 20 de ago. de 2024.

**SecurityTrails**. *Melhores práticas de segurança no desenvolvimento de PHP*. Disponível em: https://securitytrails.com/blog/php-security-best-practices. Acessado em: 15 de set. de 2024.

**SitePoint**. *Como usar Gatilhos no MySQL para monitoramento e auditoria de dados*. Disponível em: https://www.sitepoint.com/using-triggers-mysql. Acessado em: 12 de set. de 2024.

**Stack Overflow**. *Como configurar o XAMPP para ambiente de desenvolvimento PHP*. Disponível em: https://stackoverflow.com/questions/17426743/how-to-configure-xampp-for-php-development. Acessado em: 17 de set. de 2024.

**Stack Overflow**. *Como utilizar a API ViaCEP*. Disponível em: https://stackoverflow.com/questions/52875872/como-utilizar-a-api-viacep. Acessado em: 03 de set. de 2024.

**TutsPlus**. *Como Implementar Segurança no Banco de Dados MySQL com Gatilhos e Criptografia*. Disponível em: https://code.tutsplus.com/tutorials/how-to-secure-your-mysql-database-with-triggers-and-encryption--cms-33833. Acessado em: 25 de ago. de 2024.

**Udemy**. *Curso de PHP para Desenvolvedores Web*. Disponível em: https://www.udemy.com/course/php-para-desenvolvedores-web. Acessado em: 11 de set. de 2024.

**W3C**. *HTML5 Specification*. Disponível em: https://www.w3.org/TR/html5/. Acessado em: 18 de set. de 2024.