

CENTRO UNIVERSITARIO CATÓLICA DE SANTA CATARINA - ENGENHARIA DE SOFTWARE

Diogo Boege Ramuski

Gerenciamento de Inspeções de Infraestrutura Urbana

**Data de Entrega**: 04/2024.

# **Resumo**

## Este documento encontra-se o detalhamento da construção de um sistema para engenharia civil, abrangendo desde o front-end até a implementação de endpoints no back-end. O projeto visa reduzir o tempo de trabalho no escritório, acelerando o processo de carregamento de imagens e formulários. São discutidos os principais pontos do projeto, destacando sua rapidez no resultado dele, pelos aspectos de estarem tudo em módulos juntos,

## ficando mais fácil o usuário achar oque lê interessa naquele momento.

## **1. Introdução**

**1.1 Contexto**

No campo da engenharia civil, a realização de inspeções em terrenos, ruas, postes e sistemas de esgoto desempenha um papel crucial na garantia da segurança, qualidade e conformidade dos projetos de infraestrutura urbana. Essas inspeções são necessárias para identificar quaisquer problemas, irregularidades ou deficiências que possam comprometer a integridade estrutural, funcionalidade ou segurança das instalações. Para facilitar e aprimorar esse processo de inspeção e relatório, será desenvolvido um sistema web que permite aos engenheiros civis realizar inspeções detalhadas em diversos elementos urbanos. Este sistema não apenas agiliza o processo de coleta de dados no campo, mas também fornece uma plataforma para documentar e relatar qualquer problema encontrado durante as inspeções.

**1.2 Justificativa**

A relevância do projeto para o campo da engenharia de software reside na sua capacidade de automatizar e aprimorar o processo de inspeção e relatório. Ao utilizar tecnologias de software, como aplicativos móveis e plataformas online, o projeto simplifica a coleta de dados, padroniza os procedimentos de inspeção e facilita a geração de relatórios detalhados. Isso não apenas aumenta a eficiência do trabalho dos engenheiros civis, mas também melhora a qualidade das informações disponíveis para análise e tomada de decisão.

**1.3 Objetivos**

## **Automatização da Coleta de Dados:**

## Desenvolver um sistema que permita aos engenheiros registrarem informações detalhadas durante as inspeções por meio de dispositivos móveis como smartphones ou tablets.

## **Padronização de Procedimentos de Inspeção:**

## Criar formulários personalizados para diferentes tipos de inspeções, garantindo que os engenheiros sigam protocolos consistentes e documentem problemas de maneira uniforme.

## **Facilitação da Geração de Relatórios:**

## Desenvolver uma plataforma que gere automaticamente relatórios detalhados após a conclusão das inspeções, incluindo dados coletados, análises e recomendações para ações corretivas.

## **Integração de Tecnologias de Apoio:**

## Integrar tecnologias complementares como geolocalização, imagens de satélite e análise de dados para enriquecer os relatórios de inspeção e fornecer insights adicionais sobre as condições das infraestruturas urbanas.

## **2. Descrição do Projeto**

**2.1 Tema do Projeto**

O projeto visa desenvolver uma plataforma de gerenciamento de inspeções de infraestrutura urbana para engenheiros civis, utilizando as seguintes tecnologias: Frontend em Next.js, Backend API em Django, Docker para containerização, Jira para gestão de projetos, GitHub para controle de versão e Postgres como banco de dados.

**2.2 Problemas a Resolver**

**Processo Manual de Inspeção:** Automatizar o processo de coleta de dados em inspeções de terrenos, ruas, postes e sistemas de esgoto para eliminar a dependência de métodos manuais suscetíveis a erros.

**Padronização de Procedimentos:** Padronizar os procedimentos de inspeção para garantir consistência e uniformidade nas documentações de problemas encontrados.

Geração de Relatórios Eficientes: Desenvolver uma plataforma que permita a geração automática de relatórios detalhados após as inspeções, facilitando a análise de dados e a tomada de decisão a partir da coleta pela inspeção.

**Integração de Tecnologias:**

Integrar diversas tecnologias, como geolocalização e imagens de satélite, para enriquecer os relatórios de inspeção e fornecer insights adicionais sobre as condições das infraestruturas urbanas.

**Funcionalidades Extras:**

O projeto não abordará funcionalidades extras que não estejam diretamente relacionadas à coleta de dados de inspeção e geração de relatórios.

**Personalização Avançada:**

Não será implementada uma personalização avançada da interface de usuário além do escopo necessário para atender aos requisitos de inspeção e relatório.

**Integrações Adicionais:**

Limitar se as integrações a tecnologias essenciais, como geolocalização e imagens de satélite, e não incluir integrações com sistemas de terceiros além do necessário para o funcionamento da aplicação.

## **3. Especificação Técnica**

**Frontend:**

Desenvolvimento utilizando Next.js para criação de uma interface web responsiva e dinâmica.

Utilização de componentes reutilizáveis para facilitar a manutenção e escalabilidade do código.

Integração com a API backend para acesso e manipulação de dados.

Implementação de autenticação de usuários para garantir acesso seguro às funcionalidades da plataforma.

**Backend:**

Desenvolvimento de uma API RESTful utilizando Django para manipulação de dados de inspeção.

Implementação de endpoints para operações CRUD (Create, Read, Update, Delete) de inspeções, incluindo agendamento, registro e geração de relatórios.

Utilização de autenticação JWT (JSON Web Tokens) para autorização de acesso aos endpoints.

Conexão com banco de dados PostgreSQL para armazenamento persistente de dados.

**Docker:**

Utilização de contêineres Docker para empacotamento e implantação da aplicação.

Definição de Dockerfiles para construção dos contêineres frontend e backend.

Uso do docker-compose para orquestração dos contêineres e configuração de redes e volumes.

**Protocolos:**

Comunicação entre frontend e backend será realizada utilizando o protocolo HTTP/HTTPS para troca de dados de forma segura e eficiente.

Utilização do protocolo JWT para autenticação de usuários na API backend.

Algoritmos:

**Geolocalização:**

Utilização de algoritmos de geolocalização para determinar as coordenadas geográficas dos pontos de inspeção.

Integração com APIs de serviços de mapas para obtenção de informações adicionais, se necessário.

**Geração de Relatórios:**

Implementação de algoritmos para análise dos dados coletados durante as inspeções.

Utilização de templates pré-definidos para geração automática de relatórios em formato PDF ou HTML.

**Procedimentos:**

Padronização dos procedimentos de inspeção conforme normas e regulamentos aplicáveis.

Definição de fluxos de trabalho claros para agendamento, realização e conclusão das inspeções.

**Formatos de Dados:**

Troca de dados entre frontend e backend será realizada principalmente em formato JSON (JavaScript Object Notation) devido à sua simplicidade e ampla compatibilidade.

Armazenamento de dados no banco de dados PostgreSQL utilizando um modelo de dados bem definido e normalizado, seguindo as melhores práticas de modelagem de dados

## **3.1. Requisitos de Software**

**Requisitos Funcionais:1. Cadastro de Usuários**

Prioridade: MédiaDescrição: Permitir que administradores adicionem novos usuários à plataforma.

**2. Criação de projetos:**

Prioridade: AltaDescrição: Ver o projeto e nele conterá as informações das inspeções.

**3. Criar Inspeção**   
Prioridade: AltaDescrição: Será um formbuilder, onde poderá construir o formulário colocando o que foi passado em campo, com fotos, texto, descrição, data e hora, e será uma extensão na página de formulários.  
  
**4. Criação de formulários**   
Prioridade: Média  
Descrição: Página que exibirá os formulários criados, oferecendo a possibilidade de editá-los. Incluirá informações como ID, estado crítico, e-mail e valor gasto com processamento e análise dos dados.

**5. Criação de item**  
Prioridade: Alta  
Descrição: Será aonde você irá associar o formulário com o projeto , podendo responder o formulário e gerar um pdf.

**Requisitos Não Funcionais:6. Interface Responsiva**

Prioridade: AltaDescrição: Adaptar a interface de usuário a diferentes dispositivos e tamanhos de tela.

**7. Segurança da Informação**

Prioridade: AltaDescrição: Utilizar práticas de segurança para proteger os dados sensíveis armazenados.

**8. Desempenho**

Prioridade: AltaDescrição: Garantir um bom desempenho mesmo em situações de grande carga de usuários.

**9. Disponibilidade**

Prioridade: AltaDescrição: Assegurar alta disponibilidade da plataforma minimizando o tempo de inatividade.

**10. Escalabilidade**

Prioridade: AltaDescrição: Projetar a plataforma para ser facilmente escalável.

**11. Usabilidade**

Prioridade: AltaDescrição: Tornar a plataforma intuitiva e fácil de usar para usuários com diferentes níveis de experiência.

**Representação dos Requisitos:**

Diagrama

Descrição gerada automaticamente

**Fluxo Principal:**

**1. Iniciar Login:**

O usuário acessa a tela de login do sistema.

O usuário digita seu nome de usuário e senha nos campos designados.

O usuário clica no botão “Entrar”.

**2. Verificação de Credenciais:**

O sistema recebe as credenciais do usuário (nome de usuário e senha).

O sistema valida as credenciais em relação ao banco de dados de usuários.

**3. Resultado da Validação:**

Credenciais Válidas:

O sistema faz a autenticação do usuário.

O sistema direciona o usuário para a tela principal do sistema.

**3.2 Usuário Inválido:**

O sistema exibe uma mensagem de erro informando que as credenciais estão incorretas.

O sistema impede o login do usuário no sistema.

**4. Acessar Tela Principal:**

O usuário acessa a tela principal com a listagem de todos os projetos criados e pode acessar as inspeções.

**Inspeção:**

O usuário entra na tela do formbuilder.

O usuário irá montar o formulário.

O usuário pode salvar o formulário.

**Pós-condições:**

O formulário será criado e armazenado no banco de dados.

O formulário aparecera na página de formulários.

**Fluxos Alternativos:**

Caso esqueça de preencher algum campo o sistema ira dar uma mensagem de erro dizendo : Preencha todos os campos .

**6. Formulário:**

**Fluxo Principal:**

O usuário irá ver os formulários criado no formbuilder em uma tabela.

O usuário poderá alterar o, id, email , dado crítico da inspeção e valor .

**Pós-condições:**

O formulário poderá ser exportado para word

**Fluxos Alternativos:**

Falta de informações: Caso haja alguma irregularidade na hora de editar o formulário e for para exportar irá dar um erro para colocar algum campo faltante.

**Fluxos de Exceção:**

Erro em salvar: O sistema exibe uma mensagem de erro informando o motivo do erro.

**Considerações de Design:**

**Visão Geral:**

O projeto prevê a criação de uma plataforma de automação de coleta de dados e geração de relatórios, utilizando uma stack tecnológica robusta que combina Next.js para o frontend, Django para a API backend, Docker para conteinerização e PostgreSQL como banco de dados. Esta plataforma será projetada para ser eficiente, segura e usável, tornando mais fácil o trabalho dos engenheiros civis ao fornecer funcionalidades como criação de formulários personalizados, geração automática de relatórios e integração com tecnologias de suporte como geolocalização.

**Padrões de Arquitetura:**

Será utilizada uma combinação dos padrões arquiteturais MVC (Model-View-Controller), Clean Architecture e os princípios SOLID (Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion). Essa abordagem visa criar um sistema robusto, escalável e fácil de manter.

**Modelo C4:  
  
Nivel 1:**Mostra o sistema como um todo e suas interações com usuários e sistemas externos. É uma visão de alto nível.  
  
  
Diagrama

Descrição gerada automaticamente

**Nivel 2:**

Detalha os principais contêineres (aplicações, serviços, bases de dados) que compõem o sistema e como eles se comunicam.

Diagrama

Descrição gerada automaticamente

**Nivel 3:**

Foca nos componentes internos de cada contêiner, detalhando suas responsabilidades e interações.  
  
Modelos de contêiner:  
  
Texto branco sobre fundo azul

Descrição gerada automaticamente com confiança baixa

Diagrama

Descrição gerada automaticamente

**Frameworks e Bibliotecas:**

**Front-end:**

**Next.js:**

Biblioteca JavaScript para construção de interfaces de usuário.

**Shadcs:**

Biblioteca de componentes UI para Next.js, que tem muitos componentes prontos para serem usados

**Back-end:  
  
  
Next.js 16.4:**Versão mais recente, já que é uma biblioteca do react.

**Django:**

Framework utilizado para a criação do back-end.

**Django-Api:**

Criado para conectar o beck-end com o front-end   
  
  
**Banco de dados:**   
  
PostgreSQL: Sistema de gerenciamento de banco de dados relaciona, conhecido pelos seus recursos avançados.

**Considerações de Segurança:**

**Autenticação e Autorização Seguras:**

Implementaremos um sistema de autenticação robusto, utilizando JWT (JSON Web Tokens) para autorização de acesso aos endpoints da API. Apenas usuários autorizados terão permissão para acessar e manipular dados sensíveis.  
  
**Proteção contra-ataques de Força Bruta:**

Implementaremos medidas para proteger contra-ataques de força bruta, como limitação de tentativas de login e bloqueio temporário de contas após várias tentativas malsucedidas.

**Próximos Passos:**-Criação das telas em algum ambiente de desing.

- Criação das telas no front-end.  
- Integrações da api.

Referencias :  
  
Django: <https://www.djangoproject.com>

Docker: <https://www.docker.com>

Next: <https://nextjs.org>

Postgresql: <https://www.postgresql.org>

Shadcs: <https://ui.shadcn.com>

Considerações Professor/a:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Considerações Professor/a:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Considerações Professor/a: