**Metodologia de Teste para API de Gerenciamento de Posts**

Objetivo: o objetivo é garantir que todas as funcionalidades da API funcionem conforme esperado e identificar o máximo possível de bugs.

2. Escopo do Teste

Funcionalidades a serem testadas:

- Criação de usuários

- Criação de posts

- Listagem de posts

- Actualização de posts

- Exclusão de posts

Endpoints a serem testados:

- /users

- /posts

- /posts/{id}

- /token

3. Tipos de Teste

Testes Funcionais: Verificar se cada funcionalidade da API funciona corretamente.

Testes de Autenticação: Verificar se a geração e validação do JWT está correta.

Testes de Segurança: Garantir que ações só podem ser realizadas por usuários autenticados.

Testes de Integração: Verificar se a API funciona corretamente com a base de dados SQLite.

4. Ferramentas de Teste

Linguagem de Programação: Python

Framework de Teste: pytest, requests

Relatórios: pytest-html para gerar relatórios HTML

5. Plano de Teste

Etapas do Plano de Teste:

1. Configuração do ambiente de teste

2. Implementação dos casos de teste

3. Execução dos testes automatizados

4. Geração de relatórios de teste

5. Documentação de bugs encontrados

6. Casos de Teste

Exemplo de Caso de Teste:

ID: TC001

Descrição: Testar a criação de um novo usuário

Pré-condições: Nenhuma

Passos:

1. Enviar uma requisição POST para /users com dados válidos

Dados de Entrada: { "username": "user", "password": "pass123" }

Resultado Esperado: Resposta com status 201 e dados do usuário criado

7. Critérios de Aceitação

Todos os testes devem passar sem falhas

Todos os bugs críticos devem ser resolvidos

8. Relatórios de Teste

Formato do Relatório:

Resumo dos testes executados

Número de testes passados e falhados

Lista detalhada de bugs encontrados

Tempo total de execução dos testes

9. Anexos

Plano de Teste (xlsx)

Relatório de Bugs (xlsx)

Código dos Testes Automatizados