**Plano de Teste – Software**

**Nome: Danilo, Carlos, Ana, Gui**

**1- Objetivo do Plano de Testes**

**O principal objetivo é reduzir erros e tornar o aplicativo mais seguro, oferecendo mais recursos para os usuários.**

**2 Escopo do Plano de Testes**

**Vamos definir o que vamos testar e o que não está no nosso radar:**

**Frontend Testaremos a validação das interfaces, responsividade e como o app se comporta em diferentes navegadores. Isso é essencial para garantir uma navegação fluida e eficiente com nosso banco de dados.**

**Backend Focaremos nas funcionalidades das APIs, desempenho, segurança e no gerenciamento do banco de dados. Isso nos ajudará a evitar problemas e proteger os dados dos usuários.**

**3. Abordagem de Testes**

**Vamos explicar como faremos os testes em cada parte do sistema:**

**Frontend**

**Testes Funcionais Vamos verificar se a aplicação funciona corretamente como um todo, integrando todas as partes.**

**Testes de Interface Conferiremos se cores e imagens estão corretas e se a interface é intuitiva para o usuário.**

**Backend**

**Testes função ou módulo será testado individualmente, usando ferramentas ou**

**Testes de Integração Vamos garantir que diferentes partes do backend funcionem bem juntas, testando APIs e banco de dados .**

**4. Casos de Teste**

**Aqui estão alguns exemplos do que vamos testar:**

**Verificar se o layout está certo em diferentes tamanhos de tela.**

**- Testar interações com botões e formulários para garantir que tudo funcione como esperado.**

**- Conferir se as mensagens de erro aparecem corretamente.**

**- backend - Validar as respostas de cada endpoint (como /login e /create) para garantir a segurança dos dados.**

**- Testar a integração com o banco de dados e a consistência dos dados.**

**- Verificar se as permissões e autenticações estão funcionando corretamente.**

**5Ambiente de Testes**

**Precisamos especificar onde faremos os testes:**

**- Sistemas operacionais, navegadores e versões do Python que usaremos.**

**- Bancos de dados de teste (MySQL, PostgreSQL, SQLite).**

**- Infraestrutura necessária (servidores locais ou em nuvem).**

**6.Ferramentas de Testes**

**Usaremos algumas ferramentas legais para facilitar nosso trabalho:**

**- Frontend** framework de testes para JavaScript

**- Backend**

**- Integração Contínua, GitHub.**

**7. Critérios de Aceitação**

**Vamos definir o que significa que um teste foi bem-sucedido:**

**- O tempo de resposta das APIs deve ser inferior a 5 segundos.**

**- Não pode haver erros visuais em diferentes resoluções.**

**- A autenticação deve funcionar para todos os perfis de usuário.**

**8. Relatórios de Testes**

**Como vamos documentar os resultados dos testes:**

**- Usaremos ferramentas de gerenciamento**

**- Registraremos logs e relatórios gerados pelas ferramentas de testes automáticos.**

**\*\*Frontend\*\***

**| ID Teste | Descrição | Esperado |**

**|----------|----------------------------------|----------------------|**

**| F001 | Teste de Responsividade | Layout correto em 1024x768 |**

**| F002 | Teste de Formulário de Login | Mostrar mensagem de erro para credenciais incorretas |**

**| F003 | Teste de Navegação | Navegação normal entre páginas |**

**\*\*Backend\*\***

**| ID Teste | Descrição | Esperado |**

**|----------|------------------------------------|----------------------------|**

**| B001 | Teste de API de Login | Autenticação correta para credenciais válidas e inválidas |**

**| B002 | Teste de Endpoint de Criação | Registro criado com sucesso |**

**| B003 | Teste de Carga API de Consulta | Consultar 1000 registros em menos de 5 segundos |**