### Documentação do Projeto

#### Arquitetura do Projeto

Este projeto utiliza uma arquitetura de aplicação dividida em Frontend e Backend, implementando uma abordagem moderna e escalável. A escolha de tecnologias e a estrutura da aplicação foram feitas para garantir a manutenção, a escalabilidade e a facilidade de desenvolvimento e testes.

#### Tecnologias Utilizadas

##### Frontend

1. \*\*React\*\*:

- \*\*Descrição\*\*: Uma biblioteca JavaScript para a construção de interfaces de usuário.

- \*\*Motivo da Escolha\*\*: React oferece uma abordagem declarativa para a criação de componentes reutilizáveis, facilitando a manutenção e o desenvolvimento de interfaces complexas.

2. \*\*Tailwind CSS\*\*:

- \*\*Descrição\*\*: Um framework CSS utilitário que permite a criação rápida de estilos diretamente nos componentes.

- \*\*Motivo da Escolha\*\*: Tailwind permite um desenvolvimento mais rápido e eficiente, evitando a necessidade de criar classes CSS personalizadas para cada elemento, promovendo a consistência e a reutilização de estilos.

3. \*\*Vite\*\*:

- \*\*Descrição\*\*: Um build tool que oferece um ambiente de desenvolvimento rápido e otimizado.

- \*\*Motivo da Escolha\*\*: Vite proporciona tempos de inicialização rápidos e uma experiência de desenvolvimento fluida, aproveitando o suporte nativo para ES Modules.

##### Backend

1. \*\*C# com ASP.NET Core MVC\*\*:

- \*\*Descrição\*\*: Um framework de desenvolvimento web que utiliza o padrão MVC (Model-View-Controller).

- \*\*Motivo da Escolha\*\*: ASP.NET Core MVC oferece uma estrutura robusta e escalável para o desenvolvimento de APIs e aplicações web, com forte suporte para injeção de dependência, middleware e testes unitários.

2. \*\*Entity Framework Core\*\*:

- \*\*Descrição\*\*: Um ORM (Object-Relational Mapper) para .NET que facilita o trabalho com banco de dados.

- \*\*Motivo da Escolha\*\*: EF Core permite interações eficientes e seguras com o banco de dados, utilizando uma abordagem orientada a objetos.

3. \*\*xUnit\*\*:

- \*\*Descrição\*\*: Um framework de testes unitários para .NET.

- \*\*Motivo da Escolha\*\*: xUnit é amplamente utilizado na comunidade .NET e oferece funcionalidades poderosas para a criação e execução de testes unitários.

#### Decisões de Design

1. \*\*Divisão de Responsabilidades\*\*:

- \*\*Frontend e Backend Separados\*\*: A separação clara entre frontend e backend promove uma arquitetura desacoplada, facilitando a manutenção e o desenvolvimento independente das duas partes.

- \*\*Componentização com React\*\*: A utilização de componentes reutilizáveis em React garante que a interface do usuário seja modular e fácil de gerenciar.

2. \*\*Utilização de Tailwind CSS\*\*:

- \*\*Design Consistente\*\*: Tailwind permite a criação de uma interface consistente e responsiva, sem a necessidade de definir estilos globais complexos.

- \*\*Desenvolvimento Rápido\*\*: A abordagem utilitária de Tailwind acelera o desenvolvimento de estilos, permitindo ajustes rápidos e eficientes.

3. \*\*Uso de Vite\*\*:

- \*\*Desenvolvimento Rápido\*\*: Vite oferece um ambiente de desenvolvimento rápido, com tempos de inicialização reduzidos e suporte a hot module replacement (HMR), proporcionando uma experiência de desenvolvimento mais ágil.

4. \*\*ASP.NET Core MVC\*\*:

- \*\*Padrão MVC\*\*: A adoção do padrão MVC no backend organiza a aplicação em três camadas distintas (Model, View e Controller), facilitando a separação de responsabilidades e a manutenção do código.

- \*\*Injeção de Dependência\*\*: A utilização da injeção de dependência em ASP.NET Core promove a criação de código mais testável e modular.

5. \*\*Entity Framework Core\*\*:

- \*\*Abstração do Banco de Dados\*\*: O uso de EF Core abstrai as operações de banco de dados, permitindo que os desenvolvedores trabalhem com dados utilizando objetos C# em vez de comandos SQL brutos.

- \*\*Migrações\*\*: As migrações do EF Core facilitam a evolução do esquema do banco de dados de maneira controlada e versionada.

6. \*\*Testes Unitários com xUnit\*\*:

- \*\*Cobertura de Testes\*\*: A implementação de testes unitários com xUnit garante que o código seja testado adequadamente, reduzindo a ocorrência de bugs e facilitando a refatoração segura.

- \*\*Automatização\*\*: A inclusão de testes unitários permite a automatização dos testes, promovendo um ciclo de desenvolvimento contínuo e de alta qualidade.

### Conclusão

A escolha das tecnologias e as decisões de design foram feitas para garantir que o projeto seja robusto, escalável e fácil de manter. A separação clara entre frontend e backend, combinada com a utilização de frameworks modernos e práticas de desenvolvimento ágeis, assegura que a aplicação possa evoluir de maneira eficiente, atendendo às necessidades dos usuários e facilitando o trabalho dos desenvolvedores.