Documento de Estrutura de Pastas

Visão Geral

Este documento descreve a estrutura de pastas do projeto, explicando a finalidade de cada diretório. A organização do projeto segue a arquitetura MVC (Model-View-Controller) para manter o código bem estruturado e fácil de manter.

Estrutura de Diretórios

A estrutura de diretórios do projeto é organizada da seguinte maneira:

```
heavenly/
├── node_modules/
├── public/
├── src/
│   ├── controllers/
│   ├── model/
│   ├── routes/
│   ├── services/
│   ├── utils/
│   └── view/
│       ├── components/
│       ├── pages/
│       ├── styles/
│       └── assets/
├── .gitignore
├── package.json
├── README.md
└── ...
```

Descrição das Pastas

`controllers/`

- **Utilidade**: Esta pasta contém a lógica de controle do aplicativo. Os controladores são responsáveis por manipular a entrada do usuário e interagir com os modelos. Eles atuam como intermediários entre os modelos e as visualizações.

`model/`

- **Utilidade**: Esta pasta contém a lógica de dados e de negócios do aplicativo. Os modelos representam os dados e as regras de negócios da aplicação, incluindo a interação com bancos de dados.

`routes/`

- **Utilidade**: Esta pasta contém a configuração das rotas do aplicativo. As rotas definem quais páginas são chamadas com base nos URLs acessados.

`services/`

- **Utilidade**: Esta pasta contém serviços que realizam operações não diretamente relacionadas à interface do usuário, como interações com APIs externas, autenticação, e outras operações de lógica de negócios que não se enquadram diretamente em modelos ou controladores.

`utils/`

- **Utilidade**: Esta pasta contém funções utilitárias e helpers que são reutilizados em diferentes partes do aplicativo. Estes utilitários ajudam a evitar a duplicação de código e a manter o código limpo e organizado.

`view/`

- **Utilidade**: Esta pasta contém toda a interface do usuário do aplicativo. Ela é subdividida em:
    - **components/**: Contém componentes reutilizáveis da UI. Estes são blocos de construção independentes que podem ser usados em várias partes da aplicação.
        - **Exemplo**: `Header.js`, `Footer.js`
    - **pages/**: Contém as diferentes páginas do aplicativo. Cada página é composta por vários componentes e representa uma tela ou vista específica.
        - **Exemplo**: `HomePage.js`, `AboutPage.js`
    - **styles/**: Contém arquivos de estilo CSS usados para estilizar os componentes e páginas do aplicativo.
        - **Exemplo**: `App.css`, `index.css`
    - **assets/**: Contém arquivos estáticos como imagens e fontes que são utilizados na aplicação.
        - **Exemplo**: `images/`, `fonts/`

Conclusão

A estrutura de pastas organizada conforme descrito acima facilita a manutenção e o desenvolvimento do projeto, garantindo que cada parte do código esteja bem isolada e modular. Seguir esta estrutura ajuda a manter o código limpo, reutilizável e fácil de entender por outros desenvolvedores.