Estrutura do Projeto:

```
projeto-clinica-psicopedagogica/
├── frontend/
│   ├── css/
│   │   ├── global.css
│   │   ├── cadastro.css
│   │   ├── login.css
│   │   ├── perfil.css
│   │   └── tela-inicial.css
│   ├── js/
│   │   ├── cadastro.js
│   │   ├── login.js
│   │   ├── perfil.js
│   │   └── tela-inicial.js
│   ├── assets/
│   │   ├── img/
│   │   │   ├── logo.png
│   │   │   └── ... (outras imagens)
│   │   ├── icons/
│   │   │   ├── user.svg
│   │   │   └── ... (outros ícones)
│   ├── pages/
│   │   ├── cadastro.html
│   │   ├── login.html
│   │   ├── perfil.html
│   │   └── tela-inicial.html
│   └── index.html
├── backend/
│   ├── controllers/
│   │   ├── pacienteController.js
│   │   ├── profissionalController.js
│   │   ├── agendamentoController.js
│   │   └── ... (outros controllers)
│   ├── models/
│   │   ├── pacienteModel.js
│   │   ├── profissionalModel.js
│   │   ├── agendamentoModel.js
│   │   └── ... (outros models)
│   ├── routes/
│   │   ├── pacienteRoutes.js
│   │   ├── profissionalRoutes.js
```

```
│   │   ├── agendamentoRoutes.js
│   │   └── ... (outras rotas)
│   ├── config/
│   │   └── database.js
│   ├── server.js
│   └── package.json
└── database/
    └── clinicapsicopedagogica.sql
```

Conteúdo dos Arquivos:

1. `frontend/index.html` (Página Inicial do Site):

- **HTML:**
  - Estrutura básica do HTML (`<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`).
  - Cabeçalho (`<header>`) com título do site.
  - Menu de navegação (`<nav>`) com links para as páginas de cadastro, login, perfil e tela inicial.
  - Rodapé (`<footer>`) com informações de copyright.
  - Link para o arquivo `css/global.css`.

2. `frontend/pages/cadastro.html` (Página de Cadastro):

- **HTML:**
  - Estrutura básica do HTML.
  - Formulário de cadastro com campos para dados pessoais e de acesso.
  - Link para o arquivo `css/cadastro.css`.
  - Link para o arquivo `js/cadastro.js`.

3. `frontend/pages/login.html` (Página de Login):

- **HTML:**
  - Estrutura básica do HTML.
  - Formulário de login com campos para nome de usuário/e-mail e senha.
  - Link para o arquivo `css/login.css`.
  - Link para o arquivo `js/login.js`.

4. `frontend/pages/perfil.html` (Página de Perfil):

- **HTML:**
  - Estrutura básica do HTML.
  - Exibição dos dados do perfil do usuário.
  - Formulário para editar os dados do perfil.

    - Link para o arquivo `css/perfil.css`.
    - Link para o arquivo `js/perfil.js`.

5. `frontend/pages/tela-inicial.html` (Tela Inicial do Sistema):

- **HTML:**
    - Estrutura básica do HTML.
    - Cabeçalho com informações do usuário e menu de navegação principal.
    - Conteúdo principal com seções para próximas consultas, avisos, informações da clínica, etc.
    - Link para o arquivo `css/tela-inicial.css`.
    - Link para o arquivo `js/tela-inicial.js`.

6. `frontend/css/global.css` (Estilos Globais):

- **CSS:**
    - Estilos gerais para o site (fontes, cores, layout básico).
    - Estilos para elementos comuns (botões, links, etc.).

7. `frontend/css/cadastro.css`, `login.css`, `perfil.css`, `tela-inicial.css` (Estilos das Páginas):

- **CSS:**
    - Estilos específicos para cada página.

8. `frontend/js/cadastro.js`, `login.js`, `perfil.js`, `tela-inicial.js` (Scripts das Páginas):

- **JavaScript:**
    - Lógica para validar os campos dos formulários.
    - Lógica para enviar os dados para o backend (usando `fetch` ou `XMLHttpRequest`).
    - Lógica para buscar e exibir os dados do perfil e da tela inicial.
    - Lógica para manipular o DOM e lidar com eventos de usuário.

9. `backend/server.js` (Servidor Node.js):

- **JavaScript (Node.js):**
    - Configuração do servidor Express.
    - Definição das rotas da API.
    - Conexão com o banco de dados.
    - Lógica para lidar com as requisições do frontend.

10. `backend/controllers/*.js` (Controllers):

- **JavaScript (Node.js):**
    - Lógica de controle das rotas.
    - Interação com os models para realizar operações no banco de dados.

11. `backend/models/*.js` (Models):

- **JavaScript (Node.js):**
    - Definição dos modelos de dados (estrutura das tabelas do banco de dados).
    - Lógica para realizar operações no banco de dados (CRUD).

12. `backend/routes/*.js` (Rotas):

- **JavaScript (Node.js):**
    - Definição das rotas da API (endpoints).
    - Associação das rotas aos controllers correspondentes.

13. `backend/config/database.js` (Configuração do Banco de Dados):

- **JavaScript (Node.js):**
    - Configuração da conexão com o banco de dados (credenciais, etc.).

14. `database/clinicapsicopedagogica.sql` (Script SQL do Banco de Dados):

- **SQL:**
    - Comandos SQL para criar as tabelas e definir a estrutura do banco de dados.

15. `backend/package.json` (Configuração do Node.js):

- **JSON:**
    - Informações sobre o projeto Node.js.
    - Lista de dependências do projeto.

1. Estrutura de Pastas:

- **`frontend/css/`**:
    - Esta pasta deve conter todos os arquivos CSS do seu projeto.
    - Dentro dela, você pode ter:
        - **`global.css`**: Estilos globais que se aplicam a todo o site.
        - Arquivos CSS específicos para cada página (por exemplo, `cadastro.css`, `login.css`, `perfil.css`, `tela-inicial.css`).
        - Pastas para organizar estilos por módulos ou componentes (opcional, dependendo da complexidade do projeto).

2. Conteúdo dos Arquivos CSS:

- `global.css`:
    - Defina estilos básicos para elementos HTML (body, headings, paragraphs, links, etc.).
    - Defina variáveis de cores, fontes e outros estilos reutilizáveis.
    - Defina estilos para elementos de layout comuns (containers, grids, etc.).
    - Utilize um reset CSS ou normalize.css para garantir consistência entre navegadores.
- **Arquivos CSS Específicos de Página:**
    - Contenha estilos específicos para a página correspondente.
    - Utilize seletores CSS que sejam específicos para os elementos da página.
    - Evite a duplicação de estilos que já foram definidos no `global.css`.
    - Organize o codigo de forma que fique facil a leitura e a manutenção.
- **Organização do Código CSS:**
    - Utilize comentários para documentar o código e facilitar a compreensão.
    - Organize as regras CSS por ordem alfabética ou por ordem de importância.
    - Utilize indentação e espaçamento consistente para melhorar a legibilidade.
    - Utilize classes CSS descritivas e semânticas.
    - Evite o uso de estilos inline.

3. Pré-processadores CSS (Opcional):

- Considere o uso de pré-processadores CSS como SASS, LESS ou Stylus.
    - Eles oferecem funcionalidades como variáveis, mixins, nesting e funções que facilitam o desenvolvimento e a manutenção do CSS.

4. Metodologias CSS (Opcional):

- Considere o uso de metodologias CSS como BEM (Block Element Modifier) ou Atomic CSS.
    - Elas ajudam a organizar o CSS e a evitar conflitos de estilos.

Exemplo de Organização de Arquivo CSS:

CSS

```
/* global.css */

body {
  font-family: sans-serif;
  color: #333;
}

a {
  color: #007bff;
```

```
}

/* cadastro.css */

.cadastro-form {
  width: 400px;
  margin: 0 auto;
}

.cadastro-form input {
  width: 100%;
}
```

1. Estrutura de Pastas:

- **`frontend/HTML`**:
  - Esta pasta contém todo o código do frontend.
  - **`frontend/pages/`**:
    - Esta pasta contém os arquivos HTML das páginas principais do seu sistema (cadastro, login, perfil, tela inicial, etc.).
  - **`frontend/index.html`**:
    - Este é o arquivo HTML da página inicial do seu site.

2. Conteúdo dos Arquivos HTML:

- **Estrutura Básica:**
  - Todos os arquivos HTML devem seguir a estrutura básica do HTML5:
    - `<!DOCTYPE html>`: Declara o tipo de documento como HTML5.
    - `<html>`: Elemento raiz do documento.
    - `<head>`: Contém metadados da página (título, links para CSS e JavaScript, etc.).
    - `<body>`: Contém o conteúdo visível da página.
- **Organização do Conteúdo:**
  - Utilize tags HTML semânticas para estruturar o conteúdo (header, nav, main, section, article, footer, etc.).
  - Utilize headings (h1, h2, h3, etc.) para organizar o conteúdo por hierarquia.
  - Utilize listas (ul, ol, li) para agrupar itens relacionados.
  - Utilize formulários (form, input, label, button, etc.) para coletar dados do usuário.
  - Utilize imagens (img) e outros elementos multimídia de forma adequada.
- **Links para CSS e JavaScript:**
  - Utilize a tag `<link>` para incluir arquivos CSS.
  - Utilize a tag `<script>` para incluir arquivos JavaScript.

- **Comentários:**
  - Utilize comentários para documentar o código e facilitar a compreensão.
- **Indentação:**
  - Utilize indentação consistente para melhorar a legibilidade do código.

3. Nomeação de Arquivos:

- Utilize nomes de arquivos descritivos e consistentes (por exemplo, `cadastro.html`, `login.html`, `perfil.html`, `tela-inicial.html`).
- Utilize letras minúsculas e hífens para separar palavras nos nomes dos arquivos.

4. Arquivo `index.html`:

- Este arquivo deve conter a estrutura básica da página inicial do seu site.
- Ele pode conter um menu de navegação com links para as páginas principais do sistema.
- Ele pode conter informações básicas sobre a clínica.

5. Arquivos nas pastas `pages`:

- **`cadastro.html`**:
  - Formulário de cadastro.
- **`login.html`**:
  - Formulário de login.
- **`perfil.html`**:
  - Exibição e edição dos dados do perfil do usuário.
- **`tela-inicial.html`**:
  - Tela inicial do sistema com informações relevantes para o usuário.

1. Estrutura de Pastas:

- **`frontend/js/`**:
  - Esta pasta deve conter todos os arquivos JavaScript do seu projeto.
  - Dentro dela, você pode ter:
    - Arquivos JavaScript específicos para cada página (por exemplo, `cadastro.js`, `login.js`, `perfil.js`, `tela-inicial.js`).
    - Pastas para organizar scripts por funcionalidades ou módulos (por exemplo, `utils`, `api`, `components`).
    - Um arquivo `main.js` ou `app.js` para inicializar a aplicação e importar os outros módulos.

2. Conteúdo dos Arquivos JavaScript:

- **Arquivos JavaScript Específicos de Página:**
    - Contenha a lógica específica para a página correspondente.
    - Utilize funções para organizar o código e facilitar a leitura.
    - Utilize eventos para interagir com o usuário e atualizar a interface.
    - Utilize requisições AJAX (`fetch` ou `XMLHttpRequest`) para se comunicar com o backend.
    - Valide os dados dos formulários antes de enviar para o backend.
    - Manipule o DOM para atualizar o conteúdo da página dinamicamente.
- **Módulos:**
    - Crie módulos para funcionalidades reutilizáveis (por exemplo, funções utilitárias, requisições à API, componentes da interface).
    - Utilize a sintaxe de módulos ES6 (`import` e `export`) para importar e exportar funcionalidades entre os módulos.
    - Mantenha os módulos pequenos e focados em uma única responsabilidade.
- **Arquivo `main.js` ou `app.js`:**
    - Inicialize a aplicação e importe os outros módulos.
    - Defina as rotas da aplicação (se estiver usando um roteador).
    - Configure as requisições à API.
    - Inicie a lógica principal da aplicação.

3. Organização do Código JavaScript:

- **Comentários:**
    - Utilize comentários para documentar o código e facilitar a compreensão.
- **Nomenclatura:**
    - Utilize nomes de variáveis e funções descritivos e consistentes.
- **Indentação:**
    - Utilize indentação consistente para melhorar a legibilidade do código.
- **Modularização:**
    - Divida o código em módulos reutilizáveis.
- **Padrões de Projeto:**
    - Considere o uso de padrões de projeto (por exemplo, MVC, MVVM) para organizar o código e facilitar a manutenção.
- **Boas Práticas:**
    - Siga as boas práticas de programação JavaScript (por exemplo, evitar variáveis globais, utilizar `const` e `let` em vez de `var`).

4. Bibliotecas e Frameworks (Opcional):

- Considere o uso de bibliotecas ou frameworks JavaScript como React, Angular ou Vue.js.
    - Eles oferecem funcionalidades avançadas para criar interfaces complexas e facilitar o desenvolvimento.

Exemplo de Organização de Arquivo JavaScript:

JavaScript

```
// cadastro.js

function validarFormulario() {
  // Lógica para validar os campos do formulário
}

function enviarDados() {
  // Lógica para enviar os dados para o backend
}

document.addEventListener('DOMContentLoaded', () => {
  const formulario = document.querySelector('#cadastro-form');
  formulario.addEventListener('submit', (evento) => {
    evento.preventDefault();
    if (validarFormulario()) {
      enviarDados();
    }
  });
});
```