📚 Documentação Técnica do Projeto:

ONG Inclusão IA

1. Visão Geral do Projeto

O projeto **ONG Inclusão IA** é um website institucional e funcional desenvolvido para a Organização Não Governamental (ONG) focada em promover a inclusão social e digital através da Inteligência Artificial.

A arquitetura do site é baseada em um padrão **SPA (Single Page Application)** leve, utilizando apenas HTML, CSS e JavaScript Vanilla para roteamento e manipulação do DOM.

🎯 Principais Características

- **SPA Leve:** O carregamento do conteúdo principal é feito via **JavaScript (roteamento por *hash* da URL)**, sem recarregar a página, proporcionando uma experiência rápida ao usuário.
- **Design Responsivo:** Utiliza **CSS Grid** e *Media Queries* com breakpoints para Mobile (`max-width: 768px`), Tablet/Desktop Pequeno (`min-width: 768px`), Desktop Médio (`min-width: 992px`) e Desktop Grande (`min-width: 1200px`).
- **Sistema de Componentes:** Implementação de componentes modais (`modal-escondido`, `modal-visivel`), *toasts* de sucesso (`toast-escondido`, `toast-visivel`) e um menu *hamburger* customizado.
- **Validação de Formulário:** Possui um sistema robusto de validação de formulário em tempo real (on-blur e on-submit), incluindo máscaras de entrada (CPF, Telefone) e validação de consistência (CPF).

---

2. Estrutura de Arquivos e Componentes

O projeto é composto pelos seguintes arquivos principais:

Arquivo	Descrição
`index.html`	Estrutura principal, cabeçalho (`<header>`), rodapé (`<footer>`), modal de confirmação e link para os arquivos CSS e JS. Contém o container principal da SPA (`id="app-content-container"`).

Arquivo	Descrição
`style.css`	Folha de estilos. Inclui variáveis CSS, reset, layout principal (Grid), estilos de componentes (Nav, Modal, Toast) e *Media Queries* para responsividade.
`script.js`	Lógica principal da SPA. Contém o roteamento (`router()`), templates HTML via JavaScript e toda a lógica de interação (formulário, máscaras, validações, menu e modal).
`img/`	Diretório para arquivos de imagem, incluindo o favicon e os ativos do site.

3. Guia de Estilo (CSS)

O projeto segue um Guia de Estilo bem definido, centralizado em **Variáveis CSS** (Root Variables) para fácil manutenção e alteração de tema.

🎨 Variáveis de Cor (`:root`)

Variável	Valor	Descrição
`--cor-primaria`	`#007bff` (Azul)	Cor principal para ações e links.
`--cor-secundaria`	`#6c757d` (Cinza)	Cor para elementos secundários e botões de cancelamento.
`--cor-destaque`	`rgba(29, 179, 29, 0.781)` (Verde)	Para destaque em textos e links (hover).
`--cor-erro`	`#dc3545` (Vermelho)	Para mensagens de erro e bordas de input inválido.
`--cor-destaque-sucesso`	`#28a745` (Verde)	Para o componente Toast de sucesso.

📐 Sistema de Grid Customizado

O layout utiliza um sistema de **Grid de 12 colunas** dentro da classe `grid-container`.

Breakpoint	CSS (min-width)	Classes de Coluna
Mobile	`0`	`.col-1` a `.col-12` (Ocupam 12 colunas por padrão)
Tablet	`768px`	`.col-md-6`, `.col-md-4`, `.col-md-8` (Ex: 6/12)
Desktop Médio	`992px`	`.col-lg-3`, `.col-lg-6`, `.col-lg-9` (Ex: 3/12)
Desktop Grande	`1200px`	`.col-xl-2`, `.col-xl-4`, `.col-xl-8` (Ex: 2/12)

4. Lógica da Aplicação (JavaScript Vanilla)

⚙️ Roteamento (SPA)

A função `router()` é o coração da navegação, executada em `DOMContentLoaded` e no evento `hashchange`.

- **Mapeamento:** O objeto `routes` armazena os templates HTML de cada seção, indexados pelo *hash* da URL (ex: `'#cadastro'`, `'#missao'`).
- **Injeção de Conteúdo:** O conteúdo é injetado dinamicamente no `<main id="app-content-container">`.

```
const path = window.location.hash || '#inicial';

const template = routes[path];

appContentContainer.innerHTML = template;
```

📝 Formulários e Validação

A lógica do formulário de cadastro (`#cadastro`) é isolada e inicializada pela função `initializeFormLogic()` após a injeção do template.

Funcionalidade	Descrição	Funções JS (script.js)
Máscaras	Aplica formatação automática em tempo real para CPF (XXX.XXX.XXX-XX) e Telefone.	`maskCPF()`, `maskTelefone()`

Funcionalidade	Descrição	Funções JS (script.js)
Validação	Verifica se o campo está preenchido (obrigatório), formato de e-mail e validação de consistência do CPF (algoritmo básico).	`validateField()`, `validateAllFields()`, `validarCPF()`
Feedback de Erro	Adiciona classes CSS (`.input-error`, `.erro-mensagem.active`) para feedback visual.	`displayError()`, `clearError()`
Observer Alert	Um **Intersection Observer** exibe um `alert()` pop-up na primeira vez que o campo "Interesse Principal" entra na viewport, para garantir o preenchimento de um campo obrigatório chave.	`setupIntersectionObserver()`

🧭 Componentes de Navegação e Interação

- **Menu Hamburger:** No mobile (`max-width: 768px`), o menu é transformado em um *off-canvas* (fora da tela) acionado pelo botão `hamburger`.
  - **Classes de Controle:** `.menu-lista.active` e `.hamburger.is-active`.
- **Modal de Confirmação:** Utilizado para confirmar a ação de "Limpar Formulário".
  - **Classes de Controle:** `.modal-visivel` e `.modal-escondido`.
  - **Funções de Controle:** `mostrarModal()`, `esconderModal()`, `limparFormulario()`.
- **Toast de Sucesso:** Exibe uma notificação após o envio (simulado) bem-sucedido do formulário.
  - **Classes de Controle:** `.toast-visivel` e `.toast-escondido`.
  - **Função de Controle:** `mostrarToast()`.