# Documentação Técnica do Projeto Web

## Visão Geral

Este projeto consiste em uma interface web para visualização e análise de dados, com foco em:

1. **Estrutura da Interface:** Definida em `index.html`.
2. **Estilização:** Definida em `styles.css`.
3. **Lógica de Aplicação:** Implementada em `app.js`, que é um sistema de aquisição de dados em tempo real via Web Serial API.

---

## 1. Arquivo `index.html` (Estrutura da Interface)

O arquivo `index.html` define a estrutura principal da aplicação, utilizando o framework **Bootstrap** para layout e componentes.

| Seção Principal | Componentes Chave | Descrição |
| --- | --- | --- |
| **Hero/Branding** | `.brand-hero`, `.brand-switch` | Seção de destaque no topo, com título, descrição e um seletor para alternar entre modos (provavelmente Titulação/pH). |
| **Controles de Perfil** | `#instrument`, `#profile-actions` | Permite selecionar o instrumento (perfil de comunicação serial) e gerenciar perfis (importar/exportar). |
| **Controles de Conexão** | `#toggle-button`, `#status` | Botão para conectar/desconectar via Web Serial e área de status da conexão. |
| **Gráfico em Tempo Real** | `#real-time-chart` | Exibe os dados brutos recebidos em tempo real. Permite configurar limites do eixo Y (`#rt-y-min`, `#rt-y-max`) e o campo a ser plotado (`#rt-field-select`). |
| **Gráfico de Titulação** | `#experiment-chart` | Exibe o gráfico de titulação (volume vs. pH/outro parâmetro). |
| **Gráfico de Derivada** | `#derivative-chart` | Exibe o gráfico da primeira ou segunda derivada, útil para determinar o ponto de equivalência. |
| **Tabelas de Dados** | `#real-time-table`, `#experiment-table` | Exibem os dados brutos em tempo real e os pontos de titulação adicionados manualmente. |
| **Controles de Dados** | `#volume`, `#add-experiment-data-button` | Campo para inserir o volume de titulante e botão para registrar um ponto de titulação. |
| **Exportação** | Botões de download (`#download-*-data-button`) | Permitem exportar os dados brutos, de titulação e de derivada para arquivos CSV. |

## 2. Arquivo `styles.css` (Estilização)

O `styles.css` utiliza uma abordagem limpa e moderna, complementando o Bootstrap com estilos personalizados.

| Seção | Classes/Seletores Chave | Propósito |
| --- | --- | --- |
| **Variáveis** | `:root` ( `--brand-from` , `--brand-to` ) | Define cores primárias da marca para fácil manutenção. |
| **Tabelas** | `.scrollable-table` , `.table th` , `.table td` | Estilos para tornar as tabelas responsivas e com cabeçalhos fixos ( `position: sticky` ), melhorando a experiência de rolagem. |
| **Hero** | `.brand-hero` | Estiliza a seção de destaque com um gradiente de cores ( `linear-gradient` ) e sombras. |
| **Modal** | `.modal-overlay` , `.modal-panel` | Define a aparência e o comportamento de um modal de instruções, incluindo transições de opacidade e transformação. |
| **Formulários** | `.form-control:focus` | Estilos de foco aprimorados para campos de formulário. |
| **Gráficos** | `canvas` , `.derivative-chart-container` | Garante que os elementos de gráficos tenham bordas arredondadas e que o gráfico de derivada tenha dimensões fixas. |

## 3. Arquivo `app.js` (Lógica de Aplicação - Web Serial Data Acquisition)

O `app.js` é o núcleo da aplicação, responsável por gerenciar a comunicação serial, o processamento de dados, o armazenamento e a visualização.

## Fluxo de Trabalho e Componentes Chave

| Função Principal | Funções/Variáveis Chave | Descrição |
| --- | --- | --- |
| **Utilidades** | `beep()`, `csvFromRows()`, `downloadCSV()` | Funções auxiliares para feedback sonoro, formatação de dados para CSV e download de arquivos. |
| **Gerenciamento de Estado** | `rt`, `tit`, `der`, `connected`, `currentProfile` | Variáveis globais para armazenar dados em tempo real (`rt`), dados de titulação (`tit`), dados de derivada (`der`), status da conexão e o perfil de instrumento ativo. |
| **Persistência de Dados** | `persistSession()`, `restoreSession()` | Salva e restaura o estado da sessão (dados e configurações) no `localStorage` para evitar perda de dados em caso de desconexão ou fechamento acidental. |
| **Auto-Exportação** | `chooseExportFolder()`, `exportAllToFolder()` | Utiliza a File System Access API para permitir que o usuário escolha uma pasta para salvar automaticamente os dados exportados. |
| **Conexão Serial** | `connect()`, `disconnect()`, `startReading()` | Gerencia a abertura e fechamento da porta serial, e inicia o loop de leitura contínua dos dados do instrumento. |
| **Processamento de Dados** | `parseLine()`, `updateData()` | Analisa a linha de dados recebida do instrumento (baseado no perfil de instrumento) e adiciona o ponto de leitura aos arrays de dados. |
| **Cálculo de Derivada** | `calculateDerivative()` | Calcula a primeira e segunda derivada dos dados de titulação para ajudar a identificar o ponto de equivalência. |
| **Atualização de UI** | `updateRTChart()`, `updateTitChart()`, `renderTable()` | Atualiza os gráficos (Chart.js) e as tabelas HTML com os dados mais recentes. |

| | | |
|---|---|---|
| | | dados mais recentes. |

## Fluxo de Inicialização

1. **Inicialização:** Ao carregar a página ( `DOMContentLoaded` ), o script tenta restaurar uma sessão anterior ( `restoreSession()` ) e carrega os perfis de instrumentos ( `loadProfiles()` ).
2. **Persistência:** O armazenamento persistente é solicitado ( `requestPersistentStorage()` ) e a pasta de auto-exportação é restaurada ( `restoreExportFolder()` ).
3. **Validação:** O intervalo de leitura é validado ( `validateInterval()` ).

## Fluxo de Conexão

1. **Conexão:** O usuário clica em "Conectar". A função `connect()` solicita acesso à porta serial e abre a conexão com as configurações do perfil selecionado.
2. **Leitura:** `startReading()` inicia um `setInterval` que chama `reader.read()` para ler continuamente os dados brutos da porta serial.
3. **Parse:** Os dados são decodificados, o buffer é processado e as linhas são passadas para `parseLine()` para extração dos valores.
4. **Atualização:** Um segundo `setInterval` ( `setUpdateInterval()` ) chama `updateData()` para atualizar os gráficos e tabelas com os dados processados.

## Fluxo de Titulação

1. **Adição de Ponto:** O usuário insere o volume e clica no botão "Adicionar".
2. **Registro:** A função `addTitPoint()` registra o ponto ( `pH` , volume, etc.) no array `tit` .
3. **Derivada:** A função `calculateDerivative()` é chamada para recalcular a derivada com o novo ponto.
4. **Atualização:** Os gráficos de titulação e derivada são atualizados.
5. **Persistência:** O estado é salvo ( `persistSession()` ).

---