

Guia Detalhado: A Estrutura (HTML)

Este guia explica, em detalhes, a estrutura do arquivo telaAdmin.html. O HTML é o "esqueleto" da nossa página, definindo quais elementos existem e onde eles estão. Todo o nosso sistema de responsividade e grande parte do layout dependem da estrutura correta das classes do Bootstrap.

1. Configuração Inicial (<head>)

O <head> não é visível na página, mas fornece ao navegador informações cruciais:

- **<meta charset="UTF-8">**: Garante que o navegador leia caracteres especiais (como "ç" e "ã") corretamente.
- **<meta name="viewport" ...>**: A tag **mais importante** para responsividade. Ela diz aos dispositivos móveis: "use a largura real do seu dispositivo como a largura da tela e não aplique zoom". Sem isso, o Bootstrap não funciona.
- **Links do Bootstrap (CSS e JS)**:
 - **<link ... bootstrap.min.css ...>**: Importa todo o CSS do Bootstrap. É o que nos dá acesso a classes como .row, .col-md-6, .navbar, .btn, etc.
 - **<script ... bootstrap.bundle.min.js ...>**: Importa o JavaScript do Bootstrap. Isso é essencial para componentes interativos, como o menu "hambúrguer" (quando a tela encolhe) e, futuramente, tooltips ou popovers.
- **Google Fonts (Roboto e Roboto Slab)**:
 - Importamos duas fontes: Roboto (para o texto normal, por ser limpa e legível) e Roboto Slab (para títulos, por ser mais robusta e estilosa).
- **Bootstrap Icons**: Importa a biblioteca de ícones (<i class="bi bi-...">) que usamos nos cards e botões.
- **Nosso CSS (<link rel="stylesheet" href="telaAdmin.css">)**: É **crucial** que este link venha *depois* do link do Bootstrap. Isso permite que nossas regras customizadas (ex: .barra-navegacao-principal) possam sobrescrever os padrões do Bootstrap.

2. O Cabeçalho (<header>) - Responsividade com Flexbox

O cabeçalho foi um ponto crítico. A versão original quebrava em telas pequenas. A solução foi usar as **classes utilitárias de Flexbox do Bootstrap** em vez de CSS customizado.

```
<header class="cabecalho-inicial p-3 d-flex flex-wrap justify-content-between align-items-center">
```

- **p-3:** Adiciona um padding (espaçamento interno) de nível 3.
- **d-flex:** Transforma o <header> em um container flexbox.
- **flex-wrap: A CHAVE DA RESPONSIVIDADE AQUI.** Permite que os itens "quebrem" para a linha de baixo se não couberem lado a lado.
- **justify-content-between:** Tenta colocar o máximo de espaço possível *entre* os três blocos filhos (logo, etec-info, usuario-info).
- **align-items-center:** Alinha verticalmente todos os itens ao centro.

Os filhos do cabeçalho também usam classes flex para se auto-ajustarem:

- **flex-grow-1:** Aplicado no .etec-info. Diz a esse elemento: "ocupe todo o espaço livre que sobrar no meio".
- **d-flex align-items-center:** Usado no .container-logo e .container-usuario para garantir que seus itens internos (ex: logo e texto, ou nome e botão) fiquem alinhados.
- **justify-content-center** e **justify-content-md-end:** No .container-usuario, os itens ficam centralizados em telas pequenas (xs) e alinhados à direita (end) em telas médias (md) ou maiores.

3. A Barra de Navegação (<nav>) - O Componente Navbar

Usamos o componente <nav class="navbar ..."> do Bootstrap, que é feito para isso.

```
<nav class="navbar navbar-expand-lg barra-navegacao-principal">
```

```
<div class="container-fluid justify-content-center">
```

```
<!-- Botão "Hambúrguer" para telas pequenas -->
```

```
<button class="navbar-toggler" ...>
```

```
<span class="navbar-toggler-icon"></span>
```

```
</button>
```

```
<!-- Links -->
```

```
<div class="collapse navbar-collapse" id="navbarNavDropdown">
```

```
<div class="navbar-nav">
```

```

    <a class="nav-link active" data-target="dashboard">Dashboard</a>

    <a class="nav-link" data-target="usuarios">Usuários</a>

    <a class="nav-link" data-target="estoque">Estoque</a>

</div>

</div>

</div>

</nav>

```

- **navbar-expand-lg:** Diz ao Bootstrap: "expanda os links horizontalmente em telas grandes (lg) ou maiores. Em telas menores, recolha-os atrás do botão hambúrguer".
- **barra-navegacao-principal:** Nossa classe customizada (definida no CSS) para aplicar a cor de fundo vermelha e o estilo dos links.
- **justify-content-center:** Força os links a ficarem centralizados na barra.
- **Botão Toggler:** Este botão só é visível em telas pequenas (abaixo de lg). Ele controla a exibição da `div.collapse.navbar-collapse`.
- **data-target="...":** Este atributo *não* é do Bootstrap. É um "gancho" que nosso JavaScript (`telaAdmin.js`) usa para saber qual seção da página deve ser exibida quando o link é clicado.

4. O Conteúdo Principal (<main>) e o Grid

Esta é a área onde o sistema de Grid do Bootstrap brilha.

```

<main class="container py-4">

  <section id="dashboard" class="conteudo-secao ativo">

    <!-- Sistema de Grid para os Cards -->

    <div class="row g-4 mb-4">

      <!-- Coluna 1 -->

      <div class="col-lg-4 col-md-6 col-12">

        <div class="info-card"> ... </div>

      </div>

      <!-- Coluna 2 -->

      <div class="col-lg-4 col-md-6 col-12">

```

```

    <div class="info-card"> ... </div>

</div>

<!-- Coluna 3 -->

<div class="col-lg-4 col-md-6 col-12">

    <div class="info-card"> ... </div>

</div>

</div>

<!-- Tabela -->

<div class="card"> ... </div>

</section>

<section id="usuarios" class="conteudo-secao"> ... </section>

<section id="estoque" class="conteudo-secao"> ... </section>

</main>

```

- **py-4:** Um utilitário de espaçamento do Bootstrap que aplica padding (espaçamento interno) no eixo Y (topo e base).
- **div.row:** O container obrigatório para o sistema de grid. Ele define uma linha.
- **g-4:** Define um gap (vão ou "gutter") de nível 4 entre as colunas.
- **mb-4:** Adiciona uma margin-bottom (margem inferior) de nível 4.
- **col-lg-4 col-md-6 col-12:** Esta é a "lógica" da responsividade dos cards:
 - col-12: Em telas pequenas (mobile), cada coluna ocupa 12 de 12 espaços (ou seja, 100% da largura). Isso faz os cards empilharem.
 - col-md-6: Em telas médias (tablet), cada coluna ocupa 6 de 12 espaços (50% da largura). Isso mostra 2 cards por linha.
 - col-lg-4: Em telas grandes (desktop), cada coluna ocupa 4 de 12 espaços (33.3% da largura). Isso mostra 3 cards por linha.

5. Os Modais (Usuário e Estoque)

Os modais são estruturas HTML que ficam escondidas até serem ativadas. Eles não usam o modal padrão do Bootstrap, mas sim uma estrutura customizada (controlada pelo nosso CSS e JS) para maior simplicidade.

- **div.modal-container:** O fundo escuro semi-transparente que cobre a tela inteira.
- **div.modal-card:** O "cartão" branco (ou escuro, no modo escuro) que contém o formulário.
- **form-group, form-control, form-select:** Classes do Bootstrap que estilizam os formulários (campos de texto, labels, menus dropdown) automaticamente.
- **Lógica Condicional (Modal de Estoque):**
 - Repare nos campos de "Valor" no itemModal.
 - div#grupoValorUnidade e div#grupoValorPeso são dois grupos separados.
 - Nosso JavaScript (telaAdmin.js) irá mostrar ou esconder um desses grupos com base na seleção do dropdown tipoValorItem.

Guia Detalhado: O Estilo (CSS)

Este guia detalha o arquivo telaAdmin.css. O CSS dá "vida" ao HTML, definindo cores, fontes, espaçamentos e, o mais importante, as regras complexas de acessibilidade (Modo Escuro e Daltonismo). Nosso CSS funciona em *cascata* com o Bootstrap: primeiro o Bootstrap define um estilo, depois nós o sobrescrevemos.

1. Fontes (Roboto vs. Roboto Slab)

No início do arquivo, definimos as fontes que serão usadas:

- **body { font-family: 'Roboto', ... }:** Define Roboto (sans-serif) como a fonte padrão para todo o texto da página. Ela é limpa e ótima para leitura.
- **h1, h2,card-header, .info-card span, ...:** Um seletor agrupado que aplica Roboto Slab (serif) a todos os títulos, cabeçalhos de card e números de destaque. Isso cria uma hierarquia visual clara e elegante.

2. Barra de Navegação (.barra-navegacao-principal)

Esta é uma de nossas customizações mais importantes.

- **background-color: #a40000;** Define o fundo vermelho da ETEC.
- **.barra-navegacao-principal.navbar:** Removemos o padding-top e padding-bottom padrões do Bootstrap para que nossos links controlem totalmente a altura da barra.
- **.nav-link:**
 - padding: 18px 25px;; Define o espaçamento interno, que cria a altura da barra.
 - border-bottom: 5px solid transparent;; **TRUQUE IMPORTANTE.** Criamos uma borda *invisível* de 5px. Isso reserva o espaço para a barra de destaque do link ativo, impedindo que o layout "salte" quando um link é clicado.
 - position: relative;; Necessário para o truque do hover.
- **Efeito de Hover (Pseudo-elemento ::after):**
 - Não podíamos simplesmente mudar o border-bottom-color no hover, pois a cor de fundo (background-color) não se aplica à área da borda.
 - **Solução:** Criamos um pseudo-elemento ::after que tem exatamente a altura da borda (5px), o posicionamos sobre a borda invisível

(bottom: -5px;) e, no estado :hover do link, damos a ele a cor de fundo #8c0000. Isso preenche 100% da altura do item.

- **.nav-link.active:**
 - Define a cor de fundo (#8c0000) e a cor da borda (#303561) para o link que está selecionado (ativo).

3. A Lógica do Modo Escuro (body.dark-mode)

Esta é a regra mais poderosa do nosso CSS. Qualquer regra que começa com .dark-mode só será aplicada se a classe dark-mode estiver presente na tag <body> (o que o JavaScript faz).

- **.dark-mode .cabecalho-inicial, .dark-mode .card, ...:** Define o fundo (#343a40) e a cor da borda (#495057) para todos os "containers" principais (cabeçalho, cards, modais).
- **.dark-mode .container-logo img: TRUQUE DE FILTRO.** Para os logos escuros não desaparecerem no fundo escuro, aplicamos um filter: invert(1) brightness(1.5);. Isso inverte as cores (preto vira branco) e aumenta um pouco o brilho.
- **.dark-mode .table:**
 - **A CORREÇÃO DEFINITIVA.** Em vez de tentar forçar a cor em cada <tr> ou <td>, nós sobrescrevemos as **Variáveis CSS do Bootstrap**:
 - --bs-table-color: #dee2e6; (Define a cor do TEXTO da tabela)
 - --bs-table-bg: #343a40; (Define o FUNDO da tabela)
 - --bs-table-striped-bg: ...; (Define o fundo da linha listrada)
 - --bs-table-hover-bg: ...; (Define o fundo do hover)
- **.dark-mode .table thead th:** O cabeçalho da tabela (Lab, Professor, etc.) recebe um fundo ligeiramente diferente (#495057) para se destacar do corpo da tabela.
- **.dark-mode .modal-card .form-control, .dark-mode .modal-card .form-select:** Garante que os campos de formulário (inputs e selects) dentro dos modais também fiquem escuros.

4. A Lógica dos Modos de Daltonismo (body.protanopia, etc.)

Os modos de daltonismo aplicam um filtro de cor em *toda* a página. Esses filtros podem reduzir drasticamente o contraste entre o texto claro e o fundo claro.

- **A Estratégia:** Para garantir a legibilidade, decidimos que **ativar qualquer modo de daltonismo também força o tema escuro** nos elementos de conteúdo (como tabelas e modais), mesmo que o Modo Escuro principal não esteja ligado.
- **.body.protanopia .card, .body.deuteranopia .card, ...:** Este é um seletor agrupado. Ele aplica as *mesmas regras do modo escuro* (fundo #343a40, texto #dee2e6, etc.) aos cards e tabelas quando protanopia, deuteranopia, ou tritanopia estão ativos no <body>.
- **Sobrescritas de Alto Contraste:**
 - **Hover na Tabela:** Removemos o fundo do hover (background-color: transparent !important;) e adicionamos um border-bottom: 2px solid white; para indicar a seleção de forma clara.
 - **Hover no Dropdown de Acessibilidade:** Removemos o fundo e adicionamos text-decoration: underline; e color: white !important;. Isso faz o texto ficar branco e sublinhado, em vez de preto em um fundo cinza (que teria baixo contraste).
 - **Barra de Navegação Ativa:** .body.protanopia .nav-link.active e os outros modos recebem um border-bottom-color: #ffffff; (branco) para máximo contraste.
- **.body.protanopia { filter: ... }:** No final do arquivo, aplicamos o filtro de correção de cor real para cada modo.

Essa abordagem em cascata (Modo Base -> Modo Escuro -> Modos de Daltonismo) garante que a acessibilidade seja robusta e que as regras de alto contraste sempre vençam.

Guia Detalhado: A Interatividade (JavaScript)

Este guia detalha o arquivo telaAdmin.js. O JavaScript é o "cérebro" da página. Ele "ouve" as ações do usuário (cliques, seleções) e reage a elas, mudando o HTML e o CSS dinamicamente.

Nosso script está envolvido por um evento principal:

`document.addEventListener('DOMContentLoaded', function() { ... });` Isso é uma boa prática fundamental. Significa: "Não execute nenhum código JavaScript até que toda a estrutura HTML da página tenha sido completamente carregada". Isso evita erros de "elemento não encontrado".

1. Navegação Entre Seções (SPA "Fake")

O painel funciona como uma *Single Page Application* (SPA) simulada. Em vez de carregar novas páginas (como `usuarios.html`), nós apenas mostramos e escondemos seções (`<section>`) que já estão no `telaAdmin.html`.

1. **Seleção:** `const navLinks = document.querySelectorAll('.nav-link');` (pega todos os links da navegação) e `const sections = document.querySelectorAll('.conteudo-secao');` (pega todas as seções de conteúdo).
2. **Ouvinte de Clique:** Adicionamos um "ouvinte" de clique em *cada* `navLink`.
3. **Ação:** Quando um link é clicado:
 - `event.preventDefault();`: Impede a ação padrão do link (que seria tentar navegar para "#").
 - `const targetId = link.getAttribute('data-target');`: Lemos o atributo customizado `data-target` (ex: "dashboard", "usuarios").
 - `sections.forEach(s => s.classList.remove('ativo'));`: Escondemos *todas* as seções removendo a classe `.ativo`.
 - `document.getElementById(targetId).classList.add('ativo');`: Mostramos *apenas* a seção alvo (cujo id bate com o `data-target`) adicionando a classe `.ativo`.
 - (O CSS em `telaAdmin.css` define que `.conteudo-secao { display: none; }` e `.conteudo-secao.ativo { display: block; }`).
 - Por fim, removemos a classe `.active` de todos os links e a adicionamos apenas no link que foi clicado, para o CSS aplicar o destaque visual (fundo escuro e borda azul/branca).

2. Lógica dos Modais (Usuário e Estoque)

Temos dois modais, mas a lógica de abrir e fechar é quase idêntica e reutilizada.

A. Modal de Usuário (userModal)

1. **Seleção:** Pegamos todos os elementos pelo id: o container (userModal), o botão de abrir (abrirModalBtn) e os botões de fechar (fecharModalBtn, cancelarModalBtn).
2. **Funções:** Criamos funções simples: abrirModal (adiciona a classe .visivel) e fecharModal (remove a classe .visivel).
 - (O CSS define que .modal-container { opacity: 0; } e .modal-container.visivel { opacity: 1; }).
3. **Ouvintes:**
 - abrirModalBtn ouve por cliques e chama abrirModal.
 - fecharModalBtn e cancelarModalBtn ouvem por cliques e chamam fecharModal.
 - **Fechar ao Clicar Fora:** O ouvinte no modal (o container cinza) checa se o alvo do clique (event.target) é ele mesmo. Se for, o usuário clicou *fora* do cartão (modal-card), e o modal fecha.

B. Modal de Estoque (itemModal) - Lógica Condicional

1. **Seleção e Lógica de Abrir/Fechar:** Exatamente igual ao userModal, mas com seus próprios elementos (abrirItemModalBtn, itemModal, etc.).
2. **Lógica Condicional (Unidade vs. Peso):** Esta é a nova funcionalidade.
 - **Seleção:** Pegamos o dropdown (tipoValorItem) e os dois grupos de formulário (grupoValorUnidade, grupoValorPeso).
 - **Ouvinte:** Adicionamos um ouvinte de change (mudança) ao dropdown.
 - **Ação:** Quando o valor do dropdown muda:
 - Pegamos o valor selecionado (const valor = tipoValorItem.value;).
 - Usamos um if/else para checar se o valor é "unidade" ou "peso".
 - Se for "unidade", mostramos o grupoValorUnidade (removendo a classe d-none do Bootstrap) e escondemos o grupoValorPeso (adicionando d-none).

- Se for "peso", fazemos o oposto.

3. Lógica de Acessibilidade

Esta é a parte mais complexa e que manipula diretamente o `<body>` e o `<html>`.

1. Modo Escuro:

- Ouvimos o clique no `toggle-dark-mode`.
- Ação: `body.classList.toggle('dark-mode');`. O `.toggle()` é um atalho inteligente: se a classe existe, ele a remove; se não existe, ele a adiciona.
- **Persistência:** `localStorage.setItem('darkMode', ...)`; salva a escolha do usuário no navegador.
- **Carregamento:** No início do script, `if (localStorage.getItem('darkMode') === 'true') { ... }` checa se o usuário *já tinha* escolhido o modo escuro em uma visita anterior e o aplica automaticamente.

2. Controle de Fonte:

- Ouvimos cliques nos botões `increase-font` e `decrease-font`.
- **Ação:** Eles chamam a função `changeFontSize(valor)`.
- `changeFontSize`:
 - Lê o tamanho da fonte atual da variável CSS `--font-size-base` que definimos no `<html>` (usando `getComputedStyle`).
 - Calcula o novo tamanho (ex: $1.0\text{rem} + 0.1\text{rem} = 1.1\text{rem}$).
 - Aplica o novo tamanho de volta na variável CSS: `html.style.setProperty('--font-size-base', ...)`.
 - (Como todo o nosso site usa `rem` para fontes, mudar essa única variável no `<html>` redimensiona *todo* o texto da página proporcionalmente).

3. Modos de Daltonismo:

- Ouvimos cliques em todos os links que têm o atributo `data-mode`.
- **Ação:**
 - Lê o modo (ex: "protanopia") do atributo `data-mode`.

- Primeiro, remove *todas* as classes de daltonismo (protanopia, deuteranopia, tritanopia) para limpar o estado.
- Se o modo clicado *não* for "normal", ele adiciona a classe específica (ex: `body.classList.add('protanopia');`).
- (O CSS então cuida de aplicar o filtro de cor e forçar o tema escuro nas tabelas, como vimos no Guia CSS).