

# Gerenciador de Conteúdo

## Banco de dados



# Gerenciador de Conteúdo Banco de dados

O que é um **CMS** (Sistema de Gerenciamento de Conteúdo)?

Um CMS (Content Management System), ou Sistema de Gerenciamento de Conteúdo, é uma plataforma que permite criar, gerenciar e modificar conteúdos digitais sem a necessidade de conhecimentos avançados em programação. Ele é amplamente utilizado para a construção de sites, blogs, lojas virtuais e outros tipos de **aplicações web**.



# Gerenciador de Conteúdo Banco de dados

Principais Características de um CMS

1. **Interface amigável** - Oferece um painel de controle intuitivo para gerenciar conteúdos com facilidade.
2. **Personalização** - Permite a escolha de temas, plugins e funcionalidades extras para atender às necessidades específicas de cada projeto.
3. **Gerenciamento de Usuários** - Controla permissões e funções de diferentes usuários que acessam o sistema.
4. **SEO e Otimização** - Muitas plataformas CMS possuem ferramentas para melhorar o posicionamento nos motores de busca.
5. **Segurança** - Atualizações frequentes para proteção contra vulnerabilidades.
6. **Acessibilidade e Colaboração** - Possibilidade de vários usuários trabalharem juntos na edição e publicação de conteúdos.



# Gerenciador de Conteúdo Banco de dados

## Exemplos Populares de CMS

- **WordPress** – O CMS mais popular do mundo, ideal para blogs, sites institucionais e e-commerces.
- **Joomla** – Oferece flexibilidade para sites mais robustos e com múltiplos recursos.
- **Drupal** – Recomendado para projetos complexos e que exigem alto nível de personalização.
- **Magento** – Especializado no desenvolvimento de lojas virtuais.
- **Wix e Squarespace** – Soluções mais simples e intuitivas para quem deseja criar um site sem conhecimentos técnicos.



# Gerenciador de Conteúdo com Banco de dados

## Vantagens de Usar um CMS

-  Facilidade de uso – Mesmo sem experiência em programação, é possível criar e manter um site.
-  Redução de custos – Elimina a necessidade de um desenvolvedor para tarefas básicas.
-  Manutenção e escalabilidade – Atualizações constantes permitem expandir funcionalidades sem precisar reconstruir o site do zero.
-  Comunidade e suporte – Plataformas populares contam com vasto suporte de usuários e desenvolvedores.



# Introdução ao CSS

O **CSS (Cascading Style Sheets)** é uma linguagem de estilização utilizada para definir a aparência e o layout de páginas HTML. Com o CSS, podemos alterar cores, fontes, tamanhos, espaçamentos e até criar animações, tornando os sites mais atraentes e organizados.

## Como o CSS Funciona?

O **CSS** funciona aplicando regras de estilo aos **elementos HTML**. Cada regra CSS é composta por um seletor (que indica qual elemento será estilizado) e um bloco de declarações, que contém propriedades e valores.



# Introdução ao CSS

## Exemplo de regra CSS:

```
p { color: blue; font-size: 16px; }
```

Neste exemplo:

- O seletor p seleciona todos os parágrafos da página.
- A propriedade color define a cor do texto como azul.
- A propriedade font-size define o tamanho da fonte como 16 pixels.



# Formas de Adicionar CSS

Existem três maneiras principais de aplicar CSS a um documento HTML:

1. **CSS Inline** (Dentro do próprio elemento)

Adicionamos a propriedade style diretamente

```
<p style="color: red; font-size: 20px;">Este é um parágrafo  
vermelho.</p>
```

e dentro da tag HTML:



# Formas de Adicionar CSS

Existem três maneiras principais de aplicar CSS a um documento HTML:

## 1. **CSS Inline** (Dentro do próprio elemento)

Adicionamos a propriedade style diretamente

```
<p style="color: red; font-size: 20px;">Este é um parágrafo  
vermelho.</p>
```

e dentro da tag HTML:

 Útil para pequenas alterações.

 Não recomendado para grandes projetos, pois dificulta a manutenção do código.



# Formas de Adicionar CSS

## 2. CSS Interno (Dentro da tag <style> no HTML)

Escrevemos o CSS dentro da seção <head> do documento HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            h1 { color: green; text-align: center; }
        </style>
    </head>
    <body>
        <h1>Este é um título centralizado e verde</h1>
    </body>
</html>
```

✓ Melhor organização do que o CSS inline.

✗ Ainda pode dificultar a manutenção se houver muitas regras.



# Formas de Adicionar CSS

## 3. CSS Externo (Arquivo separado .css)

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Este é um título estilizado por CSS
externo</h1></body>
</html>
```

```
style.css
h1 {
  color: purple;
  font-family: Arial, sans-serif;
}
```

✓ Melhor organização do que o CSS inline.

✗ Ainda pode dificultar a manutenção se houver muitas regras.



# Formas de Adicionar CSS

## 3. CSS Externo (Arquivo separado .css)

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Este é um título estilizado por CSS
externo</h1></body>
</html>
```

```
style.css
h1 {
  color: purple;
  font-family: Arial, sans-serif;
}
```

- ✓ A melhor abordagem para projetos grandes, pois mantém o HTML limpo e organizado.

- ✗ Exige um arquivo CSS separado



# Principais Propriedades do CSS Aqui estão algumas das propriedades CSS mais usadas:

Propriedade	Descrição	Exemplo
color	Define a cor do texto	color: blue;
background-color	Define a cor de fundo	background-color: yellow;
font-size	Tamanho da fonte	font-size: 18px;
text-align	Alinhamento do texto	text-align: center;
margin	Espaçamento externo	margin: 20px;
padding	Espaçamento interno	padding: 10px;
border	Borda do elemento	border: 2px solid black;
width	Largura do elemento	width: 100%;



# Exemplo Completo

Aqui está um exemplo de um site simples estilizado com CSS externo: Arquivo index.html

```
<!DOCTYPE html>

<html>
  <head>
    <link rel="stylesheet" href="style.css">
    <title>Meu Primeiro Site com CSS</title>
  </head>
  <body>
    <h1>Bem-vindo ao meu site!</h1>
    <p>Este é um site estilizado com CSS.</p>
  </body>
</html>
```



# Exemplo Completo

Aqui está um exemplo de um site simples estilizado com CSS externo: Arquivo style.css

```
body {  
    background-color: #f4f4f4;  
    font-family: Arial, sans-serif;  
    text-align: center;  
}  
  
h1 {  
    color: blue;  
}  
  
p {  
    font-size: 18px;  
    color: darkgray;  
}
```



# Revisando HTML/CSS

## Estilos globais

```
body {  
    margin: 0;  
    font-family: 'Open Sans', sans-serif;  
}
```

**margin: 0;** → Remove todas as margens padrão do corpo da página.

**font-family: 'Open Sans', sans-serif;** → Define a fonte padrão da página como "Open Sans" e, caso não esteja disponível, usa uma fonte genérica sem serifa (sans-serif).



# Revisando HTML/CSS

## Estilos para <main>

```
main {  
    text-align: center;  
    padding: 45px;  
}
```

- **text-align: center;** → Centraliza o texto dentro da <main>.
- **padding: 45px;** → Adiciona um espaço interno de 45px ao redor do conteúdo dentro de <main>



# Revisando HTML/CSS

## Estilização de tabelas

```
main table {  
    margin: auto;  
    font-size: 14px;  
}
```

- **margin: auto;** → Centraliza a tabela horizontalmente na <main>.
- **font-size: 14px;** → Define o tamanho da fonte para 14px dentro das tabelas.



# Revisando HTML/CSS

## Estilização de tabelas

```
main td, main th {  
    padding: 8px;  
    border-collapse: collapse;  
}
```

- **padding: 8px;** → Adiciona um espaço interno de 8px dentro de células (**<td>**) e cabeçalhos (**<th>**).
- **border-collapse: collapse;** → Faz com que as bordas das células da tabela sejam fundidas em uma única borda, criando um design mais uniforme.



# Revisando HTML/CSS

## Estilização de tabelas

```
main th {  
    border-bottom: 1px solid lightgrey;  
}
```

- **padding: 8px;** → Adiciona um espaço interno de 8px dentro de células (**<td>**) e cabeçalhos (**<th>**).
- **border-collapse: collapse;** → Faz com que as bordas das células da tabela sejam fundidas em uma única borda, criando um design mais uniforme.



# Revisando HTML/CSS

## Parágrafos dentro de <main>

```
main p {  
margin-top: 0px;  
}
```

- margin-top: 0px; → Remove o espaço superior dos parágrafos dentro da <main>, garantindo um alinhamento mais preciso.



# Revisando HTML/CSS

## Estilização de formulários

```
main form input, main form select {  
height: 24px; width: 240px;  
box-sizing: border-box;  
}
```

- margin-top: 0px; → Remove o espaço superior dos parágrafos dentro da <main>, garantindo um alinhamento mais preciso.



## Estilização de formulários

```
main form input, main form select {  
    height: 24px; width: 240px;  
    box-sizing: border-box;  
}
```

- **height: 24px;** → Define uma altura fixa de 24px para os campos de entrada (**<input>**) e seleção (**<select>**).
- **width: 240px;** → Define a largura dos campos como 240px.
- **box-sizing: border-box;** → Garante que a largura total do elemento inclua padding e border, evitando que os elementos ultrapassem o tamanho definido.



# Revisando HTML/CSS

## Estilização de formulários

```
main form label { font-size: 12px; }
```

- **font-size: 12px;** → Define o tamanho da fonte dos rótulos (**<label>**) para 12px, tornando-os mais compactos e discretos.



# Revisando HTML/CSS

## Seção .add-more

```
section.add-more {  
background-color: rgb(241, 241, 241);  
padding: 40px; margin-top: 40px;  
}
```

- **background-color: rgb(241, 241, 241);** → Define um fundo cinza claro para a seção com a classe .add-more.
- **padding: 40px;** → Adiciona um espaço interno de 40px em todos os lados da seção.
- **margin-top: 40px;** → Cria um espaçamento de 40px acima da seção .add-more, separando-a visualmente dos outros conteúdos.



# Introdução ao JavaScript

O **JavaScript (JS)** é uma linguagem de programação utilizada para tornar páginas da web interativas e dinâmicas. Enquanto o HTML estrutura a página e o CSS define seu estilo, o JavaScript adiciona funcionalidades como manipulação de eventos, animações, requisições de dados e muito mais.

## Como o JavaScript se Integra ao HTML e CSS?

Em seu arquivo index.html, adicione um trecho que faz referência a um arquivo **scripts.js**:

```
<script src="scripts.js"></script>
```

Isso indica que o JavaScript será carregado e executado assim que a página for carregada.

Agora, vamos entender como podemos usar o JavaScript para adicionar interatividade à sua página.



# Introdução ao JavaScript

Como Adicionar JavaScript ao Projeto?

O JavaScript pode ser adicionado de três formas principais:

## 1. **JavaScript Inline** (dentro do HTML)

Podemos adicionar código JS diretamente em elementos HTML, usando atributos como onclick:

```
<button onclick="alert('Botão clicado!')>Clique aqui</button>
```

- ✓ Rápido para pequenas interações.
- ✗ Não recomendado para código grande, pois dificulta a manutenção.



# Introdução ao JavaScript

## 2. JavaScript Interno (dentro da tag <script> no HTML)

Podemos adicionar o JavaScript dentro do próprio HTML, dentro da tag <script>:

```
<script>
    document.getElementById("app").style.backgroundColor = "lightblue";
</script>
```

- ✓ Melhor que o inline, pois mantém o código organizado.
- ✗ Ainda pode misturar estrutura e lógica, dificultando a manutenção.



# Introdução ao JavaScript

## 3. JavaScript Externo (arquivo separado .js)

O ideal é manter o JavaScript separado, em um arquivo .js, como scripts.js, e referenciá-lo no HTML:

```
<script src="scripts.js"></script>
```

Exemplo em **scripts.js**:

```
document.getElementById("app").style.backgroundColor = "lightgray";
```

- ✓ Melhor que o inline, pois mantém o código organizado.
- ✗ Ainda pode misturar estrutura e lógica, dificultando a manutenção.



# Introdução ao JavaScript

**Exemplo Prático:** Adicionando Interatividade ao Formulário Na sua página index.html, há um formulário para adicionar membros a uma banda. Podemos usar JavaScript para capturar os dados digitados e exibi-los na tabela.

Passo 1: Criar um Arquivo **scripts.js**, crie o arquivo scripts.js e adicione o seguinte código:

```
document.addEventListener("DOMContentLoaded", function() {  
    // Seleciona elementos do formulário e da tabela  
    const form = document.querySelector("form");  
    const tableBody = document.querySelector("tbody");  
    const btnAdicionar = document.querySelector("input[type=button]");  
  
    // Adiciona evento de clique ao botão  
    btnAdicionar.addEventListener("click", function() {  
        const nome = form.nome.value;  
        const instrumento = form.instrumento.value;  
        const novoTr = document.createElement("tr");  
        const novoTdNome = document.createElement("td");  
        const novoTdInstrumento = document.createElement("td");  
        novoTdNome.textContent = nome;  
        novoTdInstrumento.textContent = instrumento;  
        novoTr.appendChild(novoTdNome);  
        novoTr.appendChild(novoTdInstrumento);  
        tableBody.appendChild(novoTr);  
    });  
});
```



# Arquivo scripts.js

```
document.addEventListener("DOMContentLoaded", function() {  
    // Seleciona elementos do formulário e da tabela  
    const form = document.querySelector("form");  
    const tableBody = document.querySelector("tbody");  
    const btnAdicionar = document.querySelector("input[type=button]");  
  
    // Adiciona evento de clique ao botão  
    btnAdicionar.addEventListener("click", function() {  
        // Obtém valores dos campos  
        const nome = document.getElementById("fname").value;  
        const sobrenome = document.getElementById("lname").value;  
        const instrumento = document.getElementById("instrument").value;
```



# Arquivo scripts.js

```
// Verifica se os campos foram preenchidos
if (nome === "" || sobrenome === "") {
    alert("Por favor, preencha todos os campos.");
    return;
}
// Cria uma nova linha na tabela
const newRow = document.createElement("tr");
newRow.innerHTML = `<td>${nome}</td><td>${sobrenome}</td><td>${instrumento}</td>`;
// Adiciona a nova linha à tabela
tableBody.appendChild(newRow);

// Limpa os campos do formulário
form.reset();
});
});
```



# Arquivo scripts.js

## Explicação do Código:

1. **document.addEventListener("DOMContentLoaded", function() {})**
  - Aguarda o carregamento completo do HTML antes de executar o script.
2. Seleciona elementos do formulário e da tabela
  - Obtém os inputs (fname, lname, instrument) e o botão de adicionar.
3. Adiciona um evento de clique no botão
  - Quando o botão "Adicionar" é clicado, o script:
    - Obtém os valores dos inputs.
    - Cria uma nova linha na tabela.
    - Adiciona a linha na <tbody>.
    - Limpa os campos do formulário.
4. Validação Simples
  - Se os campos "Nome" ou "Sobrenome" estiverem vazios, exibe um alerta.

