

Documentação parte 2 do projeto:

Sumário

Documentação parte 2 do projeto:	1
Explicação do código em Html5:	1
Explicação sobre o código em Css:	3

Explicação do código em Html5:

```
1  <!DOCTYPE html>
2  <html Lang="pt">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Èrè Moeda Digital</title>
7      <link rel="stylesheet" href="styles.css">
8  </head>
```

Essa é a estrutura inicial do nosso site, possui linguagem em português, título, escala e está diretamente linkado com a nossa folha de estilo css.

```
9  <body>
10     <header>
11         
12         <div class="header-content">
13             <h1>Èrè Moeda Digital</h1>
14             <p>A melhor moeda da atualidade!</p>
15             <a href="#compreAqui" class="btn-destaque">Comece Agora</a>
16         </div>
17     </header>
```

Para darmos início a folha criamos uma tag chamada body e dentro dela criamos outra chamada header que é onde está localizada a nossa logo e o título do nosso site.

```

19     <main>
20         <div class="container">
21             <div class="coluna-esquerda">
22                 <div class="menu-lateral">
23                     <h3>Menu</h3>
24                     <ul>
25                         <li><a href="#quemSomos">Quem Somos</a></li>
26                         <li><a href="#nossaVisao">Nossa Visão</a></li>
27                         <li><a href="#parceiros">Parceiros</a></li>
28                         <li><a href="#quemAjudamos">Quem Ajudamos</a></li>
29                         <li><a href="#compreAqui">Compre Moedas</a></li>
30                     </ul>
31                 </div>
32             </div>

```

Em seguida criamos uma tag main e dentro dela criamos alguns containers: Um container principal a coluna de navegação um menu e para o menu criamos uma lista não ordenada com itens de navegação que são links.

```

<div class="coluna-direita">
  <section id="inicio">
    <h2>Bem-vindo ao Èrè Moeda Digital</h2>
    <p>Selecione uma opção no menu ao lado para ver mais informações.</p>
  </section>

  <section id="quemSomos" class="conteudo-oculto">
    <h2>Quem Somos</h2>
    <p>Somos uma plataforma inovadora que oferece a moeda digital mais recompensadora do mercado.</p>
  </section>

  <section id="nossaVisao">
    <h2>Nossa Visão</h2>
    <p>Queremos transformar a forma como as pessoas interagem com moedas digitais, oferecendo benefícios para consumidores.</p>
  </section>

  <section id="parceiros">
    <h2>Nossos Parceiros</h2>
    <p>Empresar parceiras que nos ajudam a tornar esse projeto uma realidade.</p>
    <div class="grid-container">
      
      
      
      
    </div>
  </section>

```

Nesta parte do código criamos mais um container para abrigar seções com explicações sobre o nosso produto. E uma pequena explicação sobre os nossos parceiros com imagens das empresas.

```

<section id="quemAjudamos">
  <h2>Quem Ajudamos</h2>
  <p>Além de garantir produtos com preços acessíveis, você ajuda comunidades necessitadas na África.</p>
  <div class="grid-container">
    
    
    
    
  </div>
</section>

```

Nesta sessão abaixo da anterior, possui uma explicação de quem são as pessoas que ajudarão comprando a moeda èrè com imagens detalhadas.

```

<section id="compreAqui">
  <h2>Compre Moedas</h2>
  <div class="formContainer">
    <div class="quantidadeMoedas">
      <label for="quantidade">Quantidade de Moedas:</label>
      <input type="number" id="quantidade" name="quantidade" min="1" value="1">
      <p>Total: <span id="total">R$ 10,00</span></p>
    </div>
    <div class="dadosPagamento">
      <form>
        <label for="nome">Nome no Cartão:</label>
        <input type="text" id="nome" name="nome" required>

        <label for="numero">Número do Cartão:</label>
        <input type="text" id="numero" name="numero" required>

        <label for="validade">Validade:</label>
        <input type="text" id="validade" name="validade" placeholder="MM/AA" required>

        <label for="cvv">CVV:</label>
        <input type="text" id="cvv" name="cvv" required>

        <button type="submit">Finalizar Compra</button>
      </form>
      <div class="cartaoImagem">
        

```

Nesta parte criamos mais uma sessão para o usuário poder acrescentar os seus dados bancários e comprar a moeda digital erè. Fizemos isso criando alguns inputs, um botão para o envio das informações e uma imagem que futuramente acrescentaremos Javascript para aparecer dinamicamente a bandeira do cartão para passar credibilidade ao cliente.

```

<footer>
  <p>Todos os direitos reservados @</p>
  <p>Produzido por Eduardo Pirolo & Marco Aurélio</p>
</footer>

```

Criamos por ultimo uma tag footer para abordar o nome dos integrantes do site e os direitos sobre o produto.

Explicação sobre o código em Css:

```

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

```

Inicialmente acrescentamos as configurações iniciais para deixar tudo o mais cru possível para podermos estilizar a pagina.

```
body {
  font-family: 'Arial', sans-serif;
  line-height: 1.6;
  background-color: #f4f4f4;
  color: #09122C;
}
```

Aqui configuramos o body, a fonte, cor de fundo e cor das letras.

```
header {
  background: url('background.jpg') no-repeat center center/cover;
  color: #fff;
  padding: 60px 20px;
  text-align: center;
  position: relative;
  overflow: hidden;
}
```

No header acrescentamos uma imagem e estilizamos ela conforme o wireframe.

```
header::before {
  content: '';
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(9, 18, 44, 0.7);
  z-index: 1;
}

header .header-content {
  position: relative;
  z-index: 2;
}
```

O <header> terá um fundo escuro semitransparente cobrindo toda sua área, enquanto os elementos dentro de .header-content aparecerão acima desse fundo.

```
header h1 {
  font-size: 3em;
  margin-bottom: 10px;
  animation: fadeIn 2s ease-in-out;
}

header p {
  font-size: 1.5em;
  margin-bottom: 20px;
  animation: fadeIn 2.5s ease-in-out;
}

header .Logo {
  position: absolute;
  top: 20px;
  right: 20px;
  width: 100px;
  height: auto;
  z-index: 3;
}
```

estiliza o <header>, aplicando animações suaves ao título (h1, 2s) e ao parágrafo (p, 2.5s). O logotipo (.Logo) é posicionado no canto superior direito (absolute) com z-index: 3 para ficar acima de outros elementos.

```
.btn-destaque {
  display: inline-block;
  padding: 10px 20px;
  font-size: 1.2em;
  background-color: #BE3144;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  transition: background-color 0.3s ease, transform 0.3s ease;
}

.btn-destaque:hover {
  background-color: #872341;
  transform: scale(1.1);
}

.container {
  display: flex;
  gap: 20px;
  max-width: 1200px;
  margin: 20px auto;
  padding: 0 20px;
}

.coluna-esquerda {
  flex: 1;
}
```

Este código CSS estiliza um botão (.btn-destaque) com cor de fundo vermelha (#BE3144), texto branco, bordas arredondadas e efeito de transição suave ao passar o mouse, mudando a cor para um tom mais escuro (#872341) e aumentando ligeiramente o tamanho (scale(1.1)). Além disso, define um layout flexível para .container, com espaçamento (gap: 20px), largura máxima de 1200px, centralização automática e preenchimento lateral. A classe .coluna-esquerda recebe flex: 1, permitindo que ela ocupe espaço proporcional dentro do container flexível.

```

.coluna-direita {
  flex: 3;
}

.menu-lateral {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.menu-lateral h3 {
  font-size: 1.5em;
  color: #BE3144;
  margin-bottom: 15px;
}

.menu-lateral ul {
  list-style: none;
}

.menu-lateral ul li {
  margin-bottom: 10px;
}

```

Define estilos para duas classes principais: `.coluna-direita` e `.menu-lateral`. A classe `.coluna-direita` tem uma propriedade `flex` definida como 3, o que significa que ela ocupará três vezes mais espaço em um layout flexível em comparação com outros elementos flexíveis. A classe `.menu-lateral` estiliza um menu lateral com um fundo branco (`#fff`), padding de 20px, bordas arredondadas e uma sombra suave. O título dentro do menu (`h3`) tem um tamanho de fonte de 1.5em e uma cor vermelha (`#BE3144`), com uma margem inferior de 15px. A lista (`ul`) dentro do menu não tem marcadores (`list-style: none`), e cada item da lista (`li`) tem uma margem inferior de 10px para espaçamento entre os itens.

```
section {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
}

section h2 {
  font-size: 2em;
  color: #BE3144;
  margin-bottom: 15px;
}

section p {
  font-size: 1.1em;
  color: #09122C;
}

.grid-container {
  display: grid;
  object-fit: contain;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
  justify-content: center;
  align-items: center;
  padding: 20px;
}
```

O código CSS apresentado define estilos para elementos `section`, `h2`, `p` e uma classe `.grid-container`. As seções (`section`) têm um fundo branco (`#fff`), `padding` de 20px, bordas arredondadas, uma sombra suave e uma margem inferior de 20px para espaçamento. Os títulos (`h2`) dentro das seções têm um tamanho de fonte de

2em, cor vermelha (#BE3144) e uma margem inferior de 15px. Os parágrafos (p) têm um tamanho de fonte de 1.1em e cor azul escuro (#09122C). A classe .grid-container cria um layout de grade com duas colunas de largura igual (repeat(2, 1fr)), um espaçamento (gap) de 20px entre os itens, e centraliza o conteúdo tanto horizontal quanto verticalmente. O padding de 20px adiciona espaço interno ao contêiner da grade.

```
.grid-container img {
  width: 250px;
  height: 180px;
  object-fit: contain;
  border-radius: 15px;
  padding: 10px;
  background-color: #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.grid-container img:hover {
  transform: scale(1.05);
  box-shadow: 0 0 15px rgba(135, 35, 65, 0.3);
}

.cartaoImagem {
  text-align: center;
  margin-top: 20px;
}

.cartaoImagem img {
  width: 200px;
  height: auto;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
```

O código define estilos para imagens dentro de um contêiner de grade e para uma classe .cartaoImagem. As imagens dentro do .grid-container têm uma largura fixa de 250px e altura de 180px, com object-fit: contain para garantir que a imagem se ajuste sem distorção. Elas têm bordas arredondadas, padding interno, fundo branco e uma sombra suave. Há também uma transição suave para efeitos de escala e sombra ao passar o mouse sobre as imagens, aumentando ligeiramente o tamanho e a intensidade da sombra. A classe .cartaoImagem centraliza o texto e

adiciona uma margem superior de 20px. As imagens dentro desta classe têm uma largura fixa de 200px, altura automática, bordas arredondadas e uma sombra mais pronunciada. Esses estilos visam melhorar a apresentação visual e a interatividade das imagens.

```
.formContainer {
  display: flex;
  gap: 20px;
}

.quantidadeMoedas, .dadosPagamento {
  flex: 1;
  background-color: #f9f9f9;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.quantidadeMoedas label, .dadosPagamento label {
  display: block;
  margin-bottom: 10px;
  font-weight: bold;
  color: #09122C;
}

.quantidadeMoedas input, .dadosPagamento input {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

O código define estilos para um contêiner de formulário e seus elementos internos. O `_formContainer` utiliza um layout flexível com um espaçamento (`gap`) de 20px entre os itens. As classes `_quantidadeMoedas` e `_dadosPagamento` têm fundo cinza claro (`#f9f9f9`), padding de 20px, bordas arredondadas e uma sombra suave. Os rótulos (`label`) dentro dessas classes são exibidos em bloco, com margem inferior de 10px, texto em negrito e cor azul escuro (`#09122C`). Os campos de entrada (`input`) ocupam 100% da largura, têm padding de 10px, margem inferior

de 15px, borda cinza (#ccc) e bordas arredondadas. Esses estilos visam criar um formulário organizado e visualmente agradável.

```
.dadosPagamento button {
  background-color: #BE3144;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.dadosPagamento button:hover {
  background-color: #872341;
}

footer {
  text-align: center;
  padding: 20px;
  background-color: #09122C;
  color: #fff;
  margin-top: 20px;
}

footer p {
  margin: 5px 0;
}

@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

O código define estilos para um botão de pagamento, um rodapé e uma regra de mídia para telas menores. O botão dentro de `.dadosPagamento` tem um fundo vermelho (`#BE3144`), texto branco, padding de 10px 20px, bordas arredondadas e uma transição suave para a cor de fundo ao passar o mouse, mudando para um tom mais escuro de vermelho (`#872341`). O rodapé (footer) é centralizado, com padding de 20px, fundo azul escuro (`#09122C`), texto branco e uma margem superior de 20px. Os parágrafos (p) dentro do rodapé têm margens reduzidas. A regra de mídia ajusta o layout para telas com largura máxima de 768px, mudando a direção do `.container` para coluna, o que é útil para designs responsivos. Esses estilos visam melhorar a usabilidade e a aparência em diferentes dispositivos.

```
@media (max-width: 768px) {  
  .container {  
    flex-direction: column;  
  }  
  
  .menu-lateral {  
    margin-bottom: 20px;  
  }  
  
  .grid-container {  
    grid-template-columns: repeat(1, 1fr);  
  }  
  
  .grid-container img {  
    width: 100%;  
    max-width: 250px;  
    height: 180px;  
  }  
}
```

define regras de mídia para telas com largura máxima de 768px, visando melhorar a responsividade do layout. Quando a tela é menor que 768px, o `.container` muda sua direção para coluna (`flex-direction: column`), empilhando os elementos verticalmente. O `.menu-lateral` recebe uma margem inferior de 20px para espaçamento. O `.grid-container` ajusta seu layout para uma única coluna (`grid-`

template-columns: repeat(1, 1fr)), garantindo que os itens sejam exibidos um abaixo do outro. As imagens dentro do .grid-container ocupam 100% da largura, mas têm uma largura máxima de 250px e altura fixa de 180px, mantendo a proporção e a consistência visual em dispositivos menores. Essas alterações visam garantir uma experiência de usuário mais adequada em dispositivos móveis.

```
@media (max-width: 480px) {  
  .grid-container {  
    grid-template-columns: repeat(1, 1fr);  
  }  
  
  .grid-container img {  
    width: 100%;  
    max-width: 200px;  
    height: 150px;  
  }  
  
  .cartaoImagem img {  
    width: 150px;  
  }  
}
```

O código define regras de mídia para telas com largura máxima de 480px, visando otimizar a exibição em dispositivos móveis pequenos. Quando a tela é menor que 480px, o .grid-container ajusta seu layout para uma única coluna (grid-template-columns: repeat(1, 1fr)), garantindo que os itens sejam exibidos um abaixo do outro. As imagens dentro do .grid-container ocupam 100% da largura, mas têm uma largura máxima de 200px e altura fixa de 150px, mantendo a proporção e a consistência visual. Além disso, as imagens dentro da classe .cartaoImagem são redimensionadas para uma largura de 150px, adaptando-se melhor ao espaço disponível em telas menores. Essas alterações visam melhorar a usabilidade e a aparência em dispositivos móveis de pequeno porte.

Explicação do código em Javascript:

Esse código JavaScript faz duas funções principais:

1. **Atualizar o valor total de moedas** com base na quantidade informada pelo usuário.
2. **Atualizar a exibição de um cartão virtual** conforme o usuário preenche os campos do formulário.

```
const quantidadeInput = document.getElementById('quantidade');  
const totalElement = document.getElementById('total');  
const valorUnitario = 1.50;
```

Pega a referência dos elementos HTML:

- quantidadeInput: Campo onde o usuário digita a quantidade de moedas.
- totalElement: Onde será mostrado o valor total.
- valorUnitario: Valor unitário da moeda (R\$ 1,50).

```
function atualizarTotal() {  
    const quantidade = quantidadeInput.value;  
    const total = quantidade * valorUnitario;  
    totalElement.textContent = `R$ ${total.toFixed(2)}`;  
}
```

Quando o usuário altera a quantidade, essa função:

- Obtém o valor digitado.
- Calcula o total (quantidade * valorUnitario).
- Exibe o valor formatado (duas casas decimais) no elemento totalElement.

```
quantidadeInput.addEventListener('input', atualizarTotal);  
atualizarTotal();
```

- ❓ A função `atualizarTotal()` é chamada inicialmente para exibir o valor correto ao carregar a página.
- ❓ O evento `input` detecta qualquer alteração no campo `quantidadeInput` e chama a função para atualizar o total.

```
// Obter referências dos campos de entrada do formulário  
const nomeInput = document.getElementById('nome');  
const numeroInput = document.getElementById('numero');  
const validadeInput = document.getElementById('validade');  
const cvvInput = document.getElementById('cvv');  
  
const numeroCartao = document.querySelector('.numeroCartao');  
const nomeCartao = document.querySelector('.nomeCartao');  
const validadeCartao = document.querySelector('.validadeCartao');  
const cvvCartao = document.querySelector('.cvvCartao');  
const cartao = document.querySelector('.cartao');
```

- ❓ Obtém referências dos campos de entrada do formulário (nome, número, validade, cvv).
- ❓ Obtém referências dos elementos onde os dados serão exibidos no cartão virtual.

```
numeroInput.addEventListener('input', () => {  
  let numero = numeroInput.value.replace(/\s/g, '');  
  numero = numero.replace(/(\d{4})/g, '$1 ').trim();  
  numeroCartao.textContent = numero || '#### #### ####';  
});
```

- Remove espaços do número do cartão.
- Formata o número, adicionando um espaço a cada 4 dígitos.

- Exibe o número no cartão virtual, ou um valor padrão se o campo estiver vazio.

```
// Atualiza o nome do titular
nomeInput.addEventListener('input', () => {
  nomeCartao.textContent = nomeInput.value || 'NOME DO TITULAR';
});
```

? Atualiza o nome exibido no cartão.

? Se o campo estiver vazio, exibe um placeholder (NOME DO TITULAR).

```
// Atualiza a validade
validadeInput.addEventListener('input', () => {
  validadeCartao.textContent = validadeInput.value || 'MM/AA';
});
```

Atualiza a data de validade no cartão virtual.

```
// Atualiza o CVV e vira o cartão
cvvInput.addEventListener('focus', () => {
  cartao.classList.add('flip');
});
```

? Quando o campo CVV recebe foco (focus), adiciona a classe flip ao cartão.

? Essa classe pode ser usada no CSS para virar o cartão e mostrar o CVV.


```
cvvInput.addEventListener('blur', () => {  
  |   cartao.classList.remove('flip');  
});
```

- Quando o campo CVV perde o foco (blur), remove a classe flip, voltando à exibição normal.

js

```
cvvInput.addEventListener('input', () => {  
  |   cvvCartao.textContent = cvvInput.value || '***';  
});
```

Atualiza o CVV exibido no cartão virtual.