# Documentação Técnica do Projeto Conversor de Moedas

## Estrutura do Projeto

O projeto consiste em vários componentes React e arquivos CSS correspondentes:

1. App.js e App.css
2. Converter.js e Converter.css
3. History.js e History.css

## Detalhamento dos Arquivos

### 1. App.js

#### Funcionalidades Principais:

* Componente principal que organiza a estrutura da aplicação.
* Gerencia o estado do histórico de conversões.
* Renderiza os componentes Converter e History.

#### Código Detalhado:

import React, { useState } from 'react';  
import Converter from './components/Converter';  
import History from './components/History';  
import './App.css'  
  
const App = () => {  
 const [history, setHistory] = useState([]); // Estado para armazenar o histórico  
  
 // Função para adicionar nova conversão ao histórico  
 const addToHistory = (conversion) => {  
 setHistory([...history, conversion]);  
 };  
  
 return (  
 <div className='App'>  
 <h1>Conversor de Moedas</h1>  
 <Converter addToHistory={addToHistory} />  
 <History history={history} />  
 </div>  
 );  
};  
  
export default App;

### 2. App.css

#### Funcionalidades:

* Define o estilo global da aplicação.
* Estabelece o layout principal usando Flexbox.
* Define estilos responsivos.

#### Código Detalhado:

\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}  
  
.App {  
 height: 100vh;  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 justify-content: center;  
 background: #2A2255;  
}  
  
h1, h2, p, span {  
 color: #BDBCDE;  
}  
  
h1 {  
 font-size: 50px;  
 margin-bottom: 30px;  
}  
  
#converterContainer, #historyContainer {  
 box-shadow: 3px 4px 27px 4px rgba(189, 188, 222, 0.38)  
}  
  
@media (max-width:425px) {  
 h1 {  
 font-size: 35px;  
 }  
}

### 3. Converter.js

#### Funcionalidades Principais:

* Gerencia a interface de conversão de moedas.
* Realiza chamadas à API para obter taxas de câmbio.
* Atualiza o histórico de conversões.

#### Código Detalhado:

import { useState } from "react";  
import "./Converter.css";  
  
const Converter = ({ addToHistory }) => {  
 const [amount, setAmount] = useState(1);  
 const [fromCurrency, setFromCurrency] = useState("USD");  
 const [toCurrency, setToCurrency] = useState("EUR");  
 const [result, setResult] = useState(null);  
  
 const convertCurrency = async () => {  
 try {  
 const response = await fetch(  
 `https://api.exchangerate-api.com/v4/latest/${fromCurrency}`  
 );  
 const data = await response.json();  
 const rate = data.rates[toCurrency];  
 const conversionResult = amount \* rate;  
 setResult(conversionResult.toFixed(2));  
  
 // Adiciona ao histórico  
 const conversion = {  
 amount,  
 fromCurrency,  
 toCurrency,  
 result: conversionResult.toFixed(2),  
 date: new Date().toLocaleString(),  
 };  
 addToHistory(conversion);  
 } catch (error) {  
 console.error("Erro ao converter moeda: ", error);  
 }  
 };  
  
 // Renderização do componente (omitido para brevidade)  
};  
  
export default Converter;

### 4. Converter.css

#### Funcionalidades:

* Define o layout e estilo do componente de conversão.
* Implementa design responsivo para diferentes tamanhos de tela.

#### Destaques do Código:

* Usa flexbox para organizar os elementos.
* Define estilos específicos para inputs, selects e botões.
* Inclui media queries para responsividade.

### 5. History.js

#### Funcionalidades Principais:

* Renderiza o histórico de conversões.
* Exibe detalhes de cada conversão realizada.

#### Código Detalhado:

import React from 'react';  
import "./History.css"  
  
const History = ({ history }) => {  
 return (  
 <div id='historyContainer'>  
 <h2>Histórico de Conversões</h2>  
 {history.length === 0 ? (  
 <p>Nenhuma conversão realizada.</p>  
 ) : (  
 <ul>  
 {history.map((item, index) => (  
 <li key={index}>  
 {item.amount} {item.fromCurrency} = {item.result} {item.toCurrency} em {item.date}  
 </li>  
 ))}  
 </ul>  
 )}  
 </div>  
 );  
};  
  
export default History;

### 6. History.css

#### Funcionalidades:

* Define o estilo do componente de histórico.
* Implementa layout responsivo.

#### Destaques do Código:

* Define largura e estilo do container de histórico.
* Estiliza a lista de conversões.
* Inclui media queries para adaptação a diferentes tamanhos de tela.

## Fluxo de Dados

1. O usuário insere dados no componente Converter.
2. Ao clicar em “Converter”, uma chamada à API é realizada.
3. O resultado é exibido e adicionado ao histórico.
4. O componente History é atualizado automaticamente com a nova conversão.

## Considerações Técnicas

1. **Estado**: Gerenciado localmente em cada componente usando useState.
2. **Props**: Usadas para passar funções e dados entre componentes.
3. **API**: Utiliza fetch para chamadas assíncronas à API de taxas de câmbio.
4. **Estilo**: Combina CSS modular por componente com estilos globais no App.css.
5. **Responsividade**: Implementada através de media queries em cada arquivo CSS.