



Documentação do Projeto — Conversor de Moedas

Escola: EEEP Deputado Roberto Mesquita

Disciplina: Qualidade e Teste de Software

Alunos(as): Alan Dyeison e Isac Rocha

Docente: Everson Sousa

18.06.2025

2. Introdução

A qualidade de software é essencial para garantir que aplicações funcionem corretamente, sejam fáceis de usar, eficientes e livres de erros. Neste contexto, a aplicação de testes automatizados se torna uma prática indispensável no desenvolvimento de software moderno.

O projeto foi escolhido por ser um problema do cotidiano: a conversão de moedas. É uma aplicação simples, porém com lógica suficiente para exercitar conceitos de qualidade, boas práticas de código e testes automatizados.

O objetivo da aplicação é permitir que usuários convertam valores entre diferentes moedas de maneira rápida e confiável.

3. Descrição da Aplicação

A aplicação é um Conversor de Moedas, que permite ao usuário selecionar uma moeda de origem, uma moeda de destino, inserir um valor e obter o valor convertido.

Público-alvo: Pessoas que desejam fazer conversões de moedas de forma prática, como estudantes, turistas ou qualquer usuário comum.

Funcionalidades principais:

- **Seleção de moeda de origem e destino.**
 - **Inserção de valor.**
 - **Conversão do valor com base nas taxas cadastradas no sistema.**
 - **Limpeza automática do resultado ao alterar os dados.**
-

4. Requisitos

Requisitos Funcionais (RF):

- RF01 — Permitir que o usuário insira um valor numérico.
- RF02 — Permitir selecionar a moeda de origem.
- RF03 — Permitir selecionar a moeda de destino.
- RF04 — Calcular e exibir o valor convertido.
- RF05 — Exibir alerta se o valor inserido for inválido.
- RF06 — Limpar o resultado automaticamente se houver mudança nos dados de entrada.

Requisitos Não Funcionais (RNF):

- RNF01 — A aplicação deve ser responsiva.
 - RNF02 — O sistema deve ter tempo de resposta imediato após clicar em "Converter".
 - RNF03 — Interface simples, intuitiva e de fácil navegação.
 - RNF04 — Código organizado, reutilizável e seguindo boas práticas de React.
-

5. Arquitetura da Aplicação

Tecnologias Utilizadas:

- **React**
- **Vite (para desenvolvimento)**
- **Jest + Testing Library (para testes)**
- **CSS (estilização)**
- **Node.js + npm (node_modules) — Gerenciador de pacotes que permite instalar bibliotecas necessárias para o projeto. A pasta node_modules contém todas as dependências instaladas.**

Estrutura de Pastas:

/src

|— **/components (não utilizado, pois o app é simples)**

|— **/tests**

| |— **App.test.jsx**

|— **App.jsx**

|— **App.css**

|— **main.jsx**

Componentização:

Por se tratar de um projeto pequeno, foi utilizado apenas um componente principal (App.jsx), que concentra toda a lógica e interface. Em projetos maiores, seria feita a separação em componentes específicos, como: InputField.jsx, SelectCurrency.jsx, ResultDisplay.jsx e Button.jsx.

6. Processo de Qualidade e Boas Práticas

- O código foi estruturado utilizando Hooks do React (useState).
 - Foram aplicadas boas práticas de legibilidade, como nomeação clara de variáveis (fromCurrency, toCurrency, amount, result).
 - Sempre que o usuário altera algum dado, o resultado anterior é apagado, prevenindo erros ou confusão na interface.
 - A aplicação atende requisitos de usabilidade: interface limpa, botões visíveis, campos acessíveis e respostas imediatas.
-

7. Testes Automatizados

Ferramentas Utilizadas:

- React — Biblioteca JavaScript para construção da interface de usuário.
- Vite — Ferramenta para desenvolvimento rápido e empacotamento da aplicação.

- **Jest + Testing Library** — Framework e biblioteca para criação de testes automatizados.
- **CSS** — Utilizado para a estilização da interface.
- **Node.js + npm (node_modules)** — Gerenciador de pacotes para instalar as dependências do projeto.

Tipos de Testes Aplicados:

- **Testes de unidade:** Funções de conversão e validações.
- **Testes de componentes:** Verificação da renderização dos elementos e comportamento dos inputs e botões.

Descrição dos Principais Testes

Teste	Descrição
Verificar se os inputs estão na tela	Testa se o campo de valor e os selects aparecem corretamente.
Verificar se o botão "Converter" existe	Garante que o botão foi renderizado na interface.
Permitir digitar no input de valor	Testa se o input aceita entrada de dados numéricos corretamente.

Alterar moedas de origem e destino **Garante que é possível mudar as moedas corretamente nos selects.**

Efetuar conversão correta **Insere um valor e verifica se o resultado da conversão é exibido corretamente.**

Impedir conversão com valor inválido **Garante que o sistema alerta ou impede se o valor inserido não for numérico ou inválido.**

Não exibir resultado antes da conversão **Testa que o resultado aparece apenas após clicar no botão "Converter", evitando confusão.**

Limpar resultado ao mudar valor ou moedas **Garante que o valor convertido desaparece se o usuário alterar qualquer campo antes de nova conversão.**

Verificar conversões específicas (ex.: BRL → USD) **Valida se taxas específicas estão sendo aplicadas corretamente.**

Testar troca de moeda após uma conversão **Garante que o resultado anterior não interfere na nova operação de conversão.**

8. Conclusão

O desenvolvimento deste projeto permitiu consolidar conhecimentos em:

- Construção de aplicações React.
- Aplicação de testes automatizados com Jest e Testing Library.
- Boas práticas de desenvolvimento e qualidade de software.

A aplicação atendeu aos objetivos propostos, funcionando de forma correta e com testes cobrindo os principais fluxos. Como melhoria futura, poderia ser adicionada uma integração com uma API de câmbio em tempo real, além da separação em mais componentes para aumentar a escalabilidade.

9. Referências

- Documentação React — <https://react.dev>
- Testing Library — <https://testing-library.com>
- Jest — <https://jestjs.io>
- Conversor de moedas — <https://www.xe.com/> (consultado para taxas)
- W3Schools CSS — <https://www.w3schools.com/css/>