

Roteiro para uso do Jest em React.

Esse roteiro descreve um projeto de calculo da formula de bhaskara. Implementando a ele teste utilizando **Jest**. As seguintes funcionalidades: calculo do delta e as duas raizes, e um formulario de entrada para a, b e c.

Passo A: Configuração inicial:

1. Crie um novo projeto React usando `npx create-react-app bhaskara`, ou navegue até o diretório do projeto existente.
2. Instale as dependências necessárias, incluindo o Jest, usando o comando `npm install --save-dev jest` ou `yarn add --dev jest`.

Passo B: Estrutura de arquivos:

1. Crie um diretório chamado `src` na raiz do projeto.
2. Dentro do diretório `src`, crie os seguintes arquivos:
 - a. **App.jsx**: Contém o componente principal que renderiza o formulário de entrada e exibe o resultado do cálculo de Bhaskara.
 - b. **Bhaskara.jsx**: Contém as funções que calculam o delta e as raízes de uma equação de segundo grau.
 - c. **App.test.jsx**: Arquivo de teste que contém os testes para o componente **App.jsx** e as funções em **Bhaskara.jsx**.


Passo C: Implementação: (arquivos ao final do roteiro)

1. No arquivo **App.test.jsx**, escreva os testes para o componente **App.jsx** e as funções em **Bhaskara.jsx**.
 - a. Teste a função de cálculo do delta para diferentes valores de a, b e c e verifique se o resultado está correto.
 - b. Teste a função de cálculo das raízes para diferentes valores de a, b e c e verifique se o resultado está correto.
2. No arquivo **Bhaskara.jsx**, implemente as funções para calcular o delta e as raízes com base nos valores de a, b e c.
3. No arquivo **App.jsx**, crie um componente que renderize um formulário de entrada para os valores de a, b e c, e exiba o resultado do cálculo de Bhaskara.

Passo D: Execução dos testes:

1. Execute o comando `npm test` ou `yarn test` no terminal para executar os testes.
2. Os resultados dos testes serão exibidos no terminal, indicando se os testes passaram ou falharam.

App.test.jsx



```
1 import React from 'react';
2 import { render, fireEvent } from '@testing-library/react';
3 import App from './App';
4 import { calcularDelta, calcularRaizes } from './Bhaskara';
5
6 test('calculates delta correctly', () => {
7   expect(calcularDelta(1, 2, 1)).toBe(0);
8   expect(calcularDelta(1, -5, 6)).toBe(1);
9   expect(calcularDelta(2, -7, 3)).toBe(25);
10  expect(calcularDelta(1, 3, 4)).toBe(-7);
11 });
12
13 test('calculates roots correctly', () => {
14   expect(calcularRaizes(1, 2, 1)).toEqual([-1]);
15   expect(calcularRaizes(1, -5, 6)).toEqual([3, 2]);
16   expect(calcularRaizes(2, -7, 3)).toEqual([3, 0.5]);
17   expect(calcularRaizes(1, 3, 4)).toEqual([]);
18 });
```

App.jsx (parte 1)

```
1 import React, { useState } from 'react';
2 import { calcularDelta, calcularRaizes } from './Bhaskara';
3
4 function App() {
5   const [a, setA] = useState('');
6   const [b, setB] = useState('');
7   const [c, setC] = useState('');
8   const [raizes, setRaizes] = useState([]);
9
10  const handleInputChange = (e) => {
11    const { name, value } = e.target;
12    if (name === 'a') setA(value);
13    if (name === 'b') setB(value);
14    if (name === 'c') setC(value);
15  };
16
17  const handleSubmit = (e) => {
18    e.preventDefault();
19    const raizesCalculadas = calcularRaizes(Number(a), Number(b), Number(c));
20    setRaizes(raizesCalculadas);
21  };
22
23  const renderRaizes = () => {
24    console.log(raizes)
25    if (raizes[0] === "na") {
26      return <p>NÃO Existem raízes reais.</p>;
27    } else if (raizes.length === 1) {
28      return <p>Existem duas raízes iguais: {raizes[0]}</p>;
29    } else {
30      return (
31        <p>
32          Existem duas raízes reais diferentes: {raizes[0]} e {raizes[1]}
33        </p>
34      );
35    }
36  };
37
```

App.jsx (parte 2)

```
1   return (
2     <div>
3       <form onSubmit={handleSubmit}>
4         <label>
5           a:
6           <input type="number" name="a" value={a} onChange={handleInputChange} />
7         </label>
8         <label>
9           b:
10          <input type="number" name="b" value={b} onChange={handleInputChange} />
11        </label>
12        <label>
13          c:
14          <input type="number" name="c" value={c} onChange={handleInputChange} />
15        </label>
16        <button type="submit">Calcular</button>
17      </form>
18      {raizes.length > 0 && (
19        <div>
20          <p>Raíces:</p>
21          {renderRaizes()}
22        </div>
23      )}
24    </div>
25  );
26 }
27
28 export default App;
```

Bhaskara.jsx

```
1 export function calcularDelta(a, b, c) {
2   return b ** 2 - 4 * a * c;
3 }
4
5 export function calcularRaizes(a, b, c) {
6   const delta = calcularDelta(a, b, c);
7   if (delta < 0) {
8     return [];
9   } else if (delta === 0) {
10    const raiz = -b / (2 * a);
11    return [raiz];
12  } else {
13    const raiz1 = (-b + Math.sqrt(delta)) / (2 * a);
14    const raiz2 = (-b - Math.sqrt(delta)) / (2 * a);
15    return [raiz1, raiz2];
16  }
17 }
18
```