**Jest Test Unitarios com Angular**

**1. Configurar Jest no Projeto Angular**

**Substituir o Karma pelo Jest**

1. **Instale o Jest e as dependências necessárias:**

bash

npm install --save-dev jest @types/jest jest-preset-angular

**2. Criar um Teste Unitário com Jest**

Um teste unitário no Angular verifica uma função, método ou componente isolado.

**Exemplo de Teste Unitário para um Serviço**

Vamos supor que você tenha o seguinte serviço math.service.ts:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

jest.spyOn

O jest.spyOn é uma função do Jest usada para monitorar (ou "espionar") chamadas feitas a um método ou função específica de um objeto. Ele é útil em testes unitários para verificar se um método foi chamado, quantas vezes foi chamado e com quais argumentos.

**Como Funciona?**

O jest.spyOn cria um "spy" (espião) em um método existente, permitindo que você:

1. Intercepte chamadas ao método.
2. Valide argumentos e comportamentos.
3. Substitua a implementação do método, se necessário.

**Sintaxe Básica**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

jest.spyOn(obj, 'method');

* **obj**: O objeto que contém o método que você quer espiar.
* **method**: O nome do método a ser espionado.

Uso do EXPECT

**Verificar Valores Simples**

Texto, Carta

Descrição gerada automaticamente

**Validar Comportamentos e Funções**

const mockFn = jest.fn();

mockFn('argumento');

expect(mockFn).toHaveBeenCalledTimes(1); // Verifica que a função foi chamada 1 vez

expect(mockFn).toHaveBeenCalledWith('argumento'); // Verifica que foi chamada com o argumento correto

Uma imagem contendo Texto

Descrição gerada automaticamente

**Trabalhar com Objetos e Arrays**

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

**Testando o Browser**

it('should call clicarCardProduto and redirect to an external link', () => {

    const ramoMock = {

      codigo: 'auto',

      nome: 'teste',

      linkRedirecionar:

        'https://www.bradescoseguros.com.br',

      contratado: true,

      direcionamentoInterno: false,

      produtosExterno: [],

      contratadoExterno: false,

    };

    const browserSpy = jest.spyOn(Browser, 'open');

    component.clicarCardProduto(ramoMock);

    expect(browserSpy).toHaveBeenCalledWith({

      url: ramoMock.linkRedirecionar,

    });

**Redirecionando Link Interno**

it('should call clicarCardProduto', () => {

const routerSpy = jest

.spyOn(routerMock, 'navigate')

.mockImplementation(() => null);

component.clicarCardProduto({ ...ramoMock, codigo: 'auto' });

expect(routerSpy).toHaveBeenCalledWith(['tabs/inicio'], {

replaceUrl: true,

queryParams: { produto: 'auto' },

});

});**Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente**