RESUMO DA AULA

Instalando e configurando o Jest no projeto

Nesta se o, vamos aprender como instalar e configurar o Jest no nosso projeto para realizar testes automatizados de maneira pr tica e eficiente. Veremos como configurar o ambiente, criar o primeiro teste e rod -lo para validar o comportamento do c digo. Passo 1: Inicializar o Projeto com npm Primeiro, precisamos inicializar um projeto com o npm para gerenciar as depend ncias necess rias. Para isso, abra o terminal e navegue at a pasta onde deseja criar o projeto de testes. No nosso exemplo, vamos usar a pasta 05-primeiro-teste-com-jest. Inicialize o npm no projeto executando o comando abaixo no terminal: npm init -y Esse comando cria automaticamente um arquivo package.json com configura es padr o. Se voc preferir preencher cada campo manualmente, basta rodar npm init sem o -y, e o npm solicitar informa es como nome do projeto, vers o, ponto de entrada, descri o, entre outros. Ap s rodar o comando, o arquivo package.json ser criado na raiz do projeto. Esse arquivo o ponto central para gerenciar depend ncias, scripts e outras configura es do projeto. Verifique o arquivo package.json para garantir que ele foi gerado corretamente. Aqui est um exemplo de um package.json b sico: json { "name": "primeiro-teste-com-jest", "version": "1.0.0", "description": "", "main": "index.js", "scripts": {}, "author": "", "license": "ISC" } Passo 2: Instalar o Jest Agora vamos instalar o Jest como uma depend ncia de desenvolvimento no nosso projeto. Para isso, voc pode seguir as recomenda es da documenta o oficial do Jest. Instale o Jest com o seguinte comando: npm install --save-dev jest Esse comando adiciona o Jest como uma depend ncia de desenvolvimento (--save-dev), o que significa que ele ser instalado apenas para ambientes de desenvolvimento. Voc tamb m pode usar o yarn caso prefira: yarn add --dev jest Ap s a instala o, verifique se o package.json foi atualizado. Voc ver uma nova entrada nas depend ncias: json "devDependencies": { "jest": "^27.0.0" } Isso indica que o Jest foi instalado corretamente e est pronto para ser usado no projeto. Passo 3: Configurar o Jest no package.json Para facilitar a execu o dos testes, vamos adicionar um script no package.json. Esse script permite

rodar o Jest com um comando simples. Adicione o script de teste dentro do objeto scripts no package.json: json "scripts": { "test": "jest" } Isso define um atalho para rodar os testes usando npm test. Sempre que voc executar npm test no terminal, o Jest ser ativado e buscar por todos os arquivos de teste no projeto. Passo 4: Criar um arquivo .gitignore Antes de avan armos, importante adicionar um arquivo .gitignore para evitar que arquivos desnecess rios sejam comitados no Git. Crie um arquivo chamado .gitignore na raiz do projeto e adicione as seguintes linhas: node_modules/ coverage/ node_modules/: Essa pasta cont m todas as depend ncias instaladas pelo npm e n o deve ser enviada para o reposit rio. coverage/: uma pasta que ser gerada pelo Jest para armazenar informa es de cobertura de testes, e tamb m n o deve ser versionada. Passo 5: Exportar a fun o a ser testada Vamos criar uma fun o simples para testarmos. No nosso projeto, j temos um arquivo calcular-valor-pedido.js que cont m uma fun o para calcular o valor total de um pedido. Vamos garantir que essa fun o esteja exportada corretamente para que possamos import -la no arquivo de teste. No arquivo calcular-valor-pedido.js, adicione a seguinte linha para exportar a fun o: module.exports = calcularValorPedido; Isso permite que a fun o seja importada em outros arquivos para ser testada. Passo 6: Criar um arquivo de teste Agora vamos criar um arquivo de teste para validar o comportamento da nossa fun o. O Jest identifica automaticamente os arquivos de teste que terminam com .test.js. Vamos seguir essa conven o e criar um arquivo chamado calcular-valor-pedido.test.js. Crie o arquivo na mesma pasta do calcular-valor-pedido.js com o nome: calcular-valor-pedido.test.js Esse nome importante, pois o Jest usa o sufixo .test.js para identificar que esse arquivo cont m testes. Passo 7: Escrever um teste b sico Dentro do arquivo calcular-valor-pedido.test.js, vamos escrever um teste simples para validar o comportamento da nossa fun o. Importe a fun o que queremos testar e escreva um teste b sico usando a API do Jest: const calcularValorPedido = require('./calcular-valor-pedido'); test('Calcula o valor total do pedido corretamente', () => { const meuPedido = { itens: [{ nome: 'Po o de energia', valor: 100, quantidade: 2 }, { nome: 'Espada longa', valor: 3000, quantidade: 1 }] }; const resultado = calcularValorPedido(meuPedido); expect(resultado).toBe(3200); // Espera que o valor total seja 3200 }); Aqui: test(): Define um teste. Ele recebe dois par metros: uma descri o do teste e uma fun o

contendo a l gica do teste. expect(): Verifica o resultado. Usamos expect(resultado).toBe(3200) para garantir que o valor calculado seja 3200. Passo 8: Rodar o teste Para rodar o teste, basta executar o seguinte comando no terminal: npm test O Jest vai buscar automaticamente todos os arquivos que terminam com .test.js e rodar os testes que encontrar. No terminal, veremos um resumo com o resultado do teste: PASS ./calcular-valor-pedido.test.js Calcula o valor total do pedido corretamente (5 ms) Isso indica que o teste passou com sucesso. Se houver algum problema, o Jest vai mostrar o erro e a linha espec fica onde ele ocorreu. Conclus o Nesta se o, configuramos o Jest no projeto, criamos nosso primeiro teste e rodamos ele usando a API do Jest. Isso nos d uma base para come ar a aplicar testes automatizados nos nossos c digos. Na pr xima se o, vamos explorar como escrever testes mais detalhados, cobrindo diferentes cen rios para garantir que nosso c digo seja realmente robusto e confi vel.
