

## Material de apoio

Site: [Geração Tech](#)  
Curso: Formação em Desenvolvedor Web - Online  
Livro: Material de apoio

Impresso por: JOÃO VITOR DE MELO FREITAS  
Data: sábado, 17 ago. 2024, 15:01

# Índice

## **1. comunicar o frontend com o backend**

1.1. Retomando o Teste

1.2. Vídeo Aula

1.3. Teste de Autenticação e Acesso à Tela de Posts

1.4. Vídeo Aula

# 1. comunicar o frontend com o backend

## Podemos comunicar o frontend com o backend?

Sim, podemos! Para isso, precisamos ajustar a rota da API no frontend para garantir que ela se comunique corretamente com o backend.

## Passo a Passo para Configuração

### 1. Instalação do Cypress:

- Primeiro, pare a aplicação [React](#) que está rodando.
- No terminal, execute o comando:

```
bash
```

```
Copiar código
```

```
npm install cypress --save-dev
```

- Isso instalará o Cypress como uma dependência de desenvolvimento.

### 2. Verificando a Instalação:

- Após a instalação, verifique no arquivo `package.json` se o Cypress foi adicionado corretamente nas dependências de desenvolvimento.

### 3. Configurando Scripts no `package.json`:

- Adicione um script no `package.json` para abrir o Cypress:

```
json
```

```
Copiar código
```

```
"scripts": { "cypress:open": "cypress open" }
```

### 4. Executando o Cypress:

- No terminal, execute o comando:

```
bash
```

```
Copiar código
```

```
npm run cypress:open
```

- Isso abrirá a interface do Cypress, onde você poderá configurar os testes.

### 5. Escolhendo o Tipo de Teste:

- O Cypress perguntará se você deseja testar componentes ou end-to-end (E2E). Escolha a opção que se adequa ao seu caso, geralmente "E2E" para testar a aplicação como um todo.

### 6. Selecionando o Navegador:

- Escolha o navegador que deseja utilizar para os testes. Vamos selecionar, por exemplo, o Edge.

### 7. Criando um Teste Simples:

- O Cypress cria automaticamente uma estrutura de pastas e arquivos de exemplo. Vamos criar um novo arquivo de teste.
- Dentro da pasta de testes (`cypress/e2e`), crie um arquivo de teste. Vamos chamar de `testando.cy.js` e adicionar o seguinte conteúdo:

```
javascript
```

```
Copiar código
```

```
describe('Teste do Frontend', () => { it('Deve acessar o frontend da aplicação', () => {
```

```
cy.visit('http://localhost:5173'); }); });
```

## Executando o Teste

### 1. Certifique-se de que o Frontend está Rodando:

- Antes de rodar o teste, certifique-se de que o frontend está rodando na porta **5173**, como configurado no exemplo acima.

### 2. Executando o Teste:

- Volte para a interface do Cypress, clique no arquivo de teste que criamos e observe o Cypress executar o teste.
- Se houver algum erro, como "HTTP sem resposta", isso pode indicar que o servidor do frontend não está rodando. Certifique-se de iniciar o servidor antes de rodar o teste novamente.

## Conclusão

Nessa aula, vimos como configurar o Cypress em uma aplicação [React.js](#), instalando-o, configurando scripts e criando um teste simples para acessar o frontend da aplicação.

No próximo passo, continuaremos expandindo os testes e integrando-os com o backend.

Até a próxima aula, pessoal! Continuem praticando e explorando o Cypress. Valeu!

## 1.1. Retomando o Teste

Tudo certo, pessoal? Vamos dar continuidade à nossa aula de Cypress.

### Retomando o Teste

Reiniciamos o servidor e o Cypress está pronto para continuar.

Vamos testar o frontend novamente para garantir que ele abre corretamente.

#### 1. Testando o Frontend:

- Execute o Cypress novamente e veja se ele abre a interface do sistema. No nosso caso, a tela de login do blog abriu com sucesso.

### Criando o Primeiro Teste

Agora, vamos criar um teste simples para verificar se os campos de login e senha estão funcionais.

#### 1. Identificando os Elementos:

- No navegador, utilize a ferramenta de inspeção de elementos para verificar os IDs ou nomes dos campos de login e senha.
- Se os campos não tiverem **ID** ou **name**, pode ser mais difícil identificá-los. Nesse caso, você pode adicionar IDs diretamente no código do frontend para facilitar o teste.

Por exemplo:

html


 Copiar código

```
<input id="login" type="text" name="username" /> <input id="senha" type="password" name="password" />
```

#### 2. Escrevendo o Teste:

- Agora, vamos escrever o teste no Cypress para verificar esses campos:

javascript

 Copiar código

```
describe('Teste de Login', () => { it('Deve preencher o login e a senha e acessar o sistema', () => {  
  cy.visit('http://localhost:5173'); // Acesse a URL do frontend // Preencher o campo de login  
  cy.get('#login').type('meuUsuario'); // Preencher o campo de senha cy.get('#senha').type('123456'); // Clicar no botão  
  de acessar cy.contains('Acessar').click(); // Verificar se o login foi bem-sucedido cy.url().should('include',  
  '/dashboard'); // Verifica se redirecionou para a página do dashboard }); });
```

### Executando o Teste

#### 1. Executando o Teste no Cypress:

- Salve o arquivo de teste e execute-o no Cypress. O teste preencherá os campos de login e senha e tentará acessar o sistema.


#### 2. Verificando o Resultado:

- Se o login for bem-sucedido, o Cypress verificará se a URL mudou para a página do dashboard (ou qualquer outra página que o login redirecione).

### Tornando o Teste Mais Visual

Se o teste estiver muito rápido e você quiser ver o que está acontecendo com mais clareza, você pode adicionar pausas entre as ações:

javascript

 Copiar código

```
cy.get('#login').type('meuUsuario'); cy.wait(1000); // Aguarda 1 segundo cy.get('#senha').type('123456'); cy.wait(1000); //  
Aguarda 1 segundo cy.contains('Acessar').click(); cy.wait(2000); // Aguarda 2 segundos
```

## Próximos Passos

Agora que o teste de login está funcionando, você pode avançar para outros testes, como:

- **Criar um novo post:** Testar a funcionalidade de criação de um post.
- **Acessar a tela de posts:** Verificar se a tela de listagem de posts está acessível e funcionando corretamente.

Continue praticando e configurando os testes para garantir que todas as funcionalidades do sistema estejam bem testadas.

Até a próxima, pessoal! Configurem seus projetos e pratiquem os testes com Cypress. Valeu!

## 1.2. Vídeo Aula

dia 48 cypress 01 converted



## 1.3. Teste de Autenticação e Acesso à Tela de Posts

### Teste de Autenticação e Acesso à Tela de Posts

#### 1. Autenticação:

- Primeiro, configuramos o Cypress para realizar o teste de autenticação. Após o login bem-sucedido, somos redirecionados para a página inicial ([home](#)).
- Agora, queremos automatizar o processo de navegação até a tela de posts.

#### 2. Navegação para a Tela de Posts:

- Após a autenticação, vamos configurar o Cypress para navegar até a página de posts:
- Identificamos o link ou botão que leva à tela de posts. Se o elemento não tiver um `id`, podemos adicioná-lo no código do frontend para facilitar a identificação durante os testes.

Exemplo:

html

 Copiar código

```
<a id="menu-posts" href="/posts">Blog</a>
```

No teste, podemos acessar esse link usando o `id`:

javascript

 Copiar código

```
cy.get('#menu-posts').click();
```

#### 3. Criando um Novo Post:

- Após navegar para a página de posts, o próximo passo é abrir o modal de criação de um novo post e preencher os campos necessários:

javascript

 Copiar código

```
cy.contains('Novo Post').click(); cy.get('input[type="text"]').type('Título do Post');  
cy.get('textarea').type('Conteúdo do Post'); cy.contains('Salvar').click();
```

### Executando e Verificando os Testes

#### 1. Executando os Testes:

- Ao executar o Cypress, ele deve passar por todos esses passos automaticamente: autenticar, navegar até a página de posts, abrir o modal e criar um novo post.

#### 2. Resolução de Erros:

- Se o Cypress encontrar algum problema, como um elemento não encontrado, você pode ajustar o código do frontend (por exemplo, adicionando um `id`) ou modificar o seletor usado no teste.
- No exemplo dado, se houver elementos com o mesmo texto ou `id`, pode ser necessário usar seletores mais específicos para garantir que o Cypress interaja com o elemento correto.

### Testando Componentes

#### 1. Configuração de Testes de Componentes:


- Além dos testes end-to-end (E2E), o Cypress também pode ser usado para testar componentes individuais. Isso é útil para verificar se cada componente está funcionando corretamente em isolamento.
- Vamos configurar o Cypress para testes de componentes. Ao iniciar essa configuração, o Cypress perguntará qual framework você está usando (neste caso, [React](#)).

#### 2. Criando um Teste de Componente:



- Após configurar o Cypress para testes de componentes, você pode criar um teste para verificar se um componente específico, como `App`, está renderizando corretamente:

javascript

 Copiar código

```
describe('Teste do Componente App', () => { it('Deve renderizar o App corretamente', () => { cy.mount(<App />); cy.get('h1').should('contain', 'Bem-vindo ao App'); }); });
```

- O comando `cy.mount(<App />);` renderiza o componente `App` para que ele possa ser testado isoladamente.

### 3. Verificando o Resultado:

- Execute o teste de componente e verifique se o Cypress consegue renderizar o componente e realizar as verificações necessárias.

## Conclusão

Com o Cypress, você pode realizar tanto testes end-to-end quanto testes de componentes, garantindo que sua aplicação [React](#) funcione como esperado, tanto no nível global quanto no nível de cada componente individual.

Continuem praticando e explorando o Cypress para fortalecer suas habilidades em testes automatizados.

Configurem seus projetos e continuem praticando.

## 1.4. Vídeo Aula

dia 48 cypress 02 converted

