10 **Localizando elementos**

Transcrição

Já entendemos que, nos testes *end-to-end*, simulamos o comportamento de uma pessoa us
através da interface.

Queremos começar a testar as funcionalidades no site Adopet. Para isso, como mencionad
precisamos seguir um roteiro de testes. Já temos um pronto, desenvolvido pela equipe.

Testando o Cadastro

Abrindo o explorador lateral do VS Code, temos um arquivo chamado `roteiro.md` no final
Selecionaremos esse arquivo e veremos em seu interior o que precisamos testar.

O primeiro texto é a funcionalidade de cadastro no site Adopet. O cenário é realizar um ca
sucesso. É importante entender isso, porque no teste não deverá ocorrer falha no cadastro

`roteiro.md`

```
Funcionalidade: Cadastro no site Adopet

Cenário: Realizar um cadastro no site com sucesso
```

Logo abaixo, temos uma sequência de passos e o resultado esperado: o sistema processar
fornecidas.

```
Passos:

O usuário acessa a página de cadastro.
O usuário preenche o campo "Nome".
O usuário preenche o campo "E-mail" com um endereço de e-mail válido.
O usuário preenche o campo "Senha" com uma senha válida.
O usuário preenche o campo "Confirmação de Senha" com a mesma senha ins
O usuário clica no botão "Cadastrar".

Resultados Esperados:
O sistema processa as informações fornecidas.
```

Logo abaixo, temos a regra de negócio.

```
Regra de Negócio:
E-mail e senha são campos obrigatórios para o cadastro.
```

É essa sequência de passos que precisamos simular com o Cypress.

Como traduzir esse cenário para o código que o Cypress vai executar? Vamos analisar a pá (https://adopet-frontend-cypress.vercel.app), acessando-a pelo navegador.

Em seu interior, a pessoa usuária vai precisar clicar no link "Cadastrar" que aparece na pág site redirecionar para a página de cadastro, a pessoa deve inserir as informações dos camp

Portanto, o primeiro passo é acessar a página do Adopet, que já realizamos, e o segundo p "Cadastrar". Como vamos dizer ao Cypress para clicar nesse link?

Vamos voltar para a página inicial e entender o que está acontecendo nessa página HTML. botão direito, vamos selecionar a opção "Inspecionar" do navegador para abrir a aba de fe desenvolvimento (*DevTools*).

Em seu interior, na guia "Elements", notaremos que temos vários elementos — como tags H CSS — que compõem a página web.

Para identificar o link de cadastrar, precisamos saber **qual é o seu elemento**. Selecionarem denominado "*Select an element in the page to inspect it*", localizado à esquerda do título da barra de ferramentas. Também é possível acessar essa funcionalidade com o atalho "Ctrl+S

Após o clique, passaremos o mouse por cima do texto do link "Cadastrar". Com isso, verem desenvolvimento que a classe `a` está destacada em azul, e nela temos um texto chamado

```
<a class="initial_link" href="/cadastro" data-test="register-button">Ca
```

Como vamos traduzir isso para o Cypress? Vamos voltar ao VS Code.

Já temos o arquivo da página de testes `spec.cy.js` da pasta "e2e" aberto, no qual vamos tr código que fizemos, de visitar a página do Adopet.

Na linha 3, já temos o `cy.visit` com a URL da página. Vamos inserir um ponto e vírgula n para finalizar a instrução e pressionar "Enter" para criar uma linha vazia abaixo dela.

Na nova linha, vamos inserir a informação de clicar no botão "Cadastrar". Faremos isso ad `cy.contains` e entre seus parênteses, vamos inserir dois parâmetros entre aspas simples: entre aspas simples, e o texto que está naquele elemento, ou seja, `Cadastrar` .

Assim, dissemos para o Cypress procurar na página web se há um elemento com o texto c encontrar esse elemento, ele precisa clicar. Para enviar a instrução de clique, adicionarem direita.

spec.cy.js

```
describe('template spec', () => {
    it('passes', () => {
        cy.visit('https://adopet-frontend-cypress.vercel.app/');
        cy.contains('a', 'Cadastrar').click();
     })
})
```

Vamos ver se está funcionando. Salvaremos o arquivo com "Ctrl+S" e voltaremos à aba do C navegador.

O Cypress já está rodando. Se ele não estiver, devemos acessar o terminal e digitar `npx cyp`

Vamos rodar de novo o teste para conseguirmos visualizar na interface. Rodaremos com o equivale a clicar no botão "Run All Tests", no canto superior direito da aba lateral.

Após rodar, veremos que o teste ocorreu com sucesso. No bloco "TEST BODY" (corpo do te conseguiu acessar a página, em seguida procurou algum elemento com "Cadastrar", clicou redirecionou para a página de cadastro.

TEST BODY

visit https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/

-contains a, Cadastrar

-click

(page load) --page loaded--

(new url) https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/cadastro

Conseguimos simular esse comportamento. Em seguida, precisamos pegar os outros elem como "Nome", e digitar.

Vamos voltar para a página do Adopet pelo navegador. Clicaremos novamente na ferramer de ferramentas de desenvolvimento e clicaremos no "Cadastrar" para acessar a página de c vamos capturar os campos.

Qual é o campo do nome? Ao posicionar o cursor sobre o campo de nome, veremos que es atributo chamado `name` que recebe um `nome` .

```
<input id="name" type="text" name="nome" placeholder="Digite seu nome c
```

Voltando para o VS Code, vamos inserir essa informação no Cypress. Vamos pressionar "E `cy.contains('a', 'Cadastrar').click()` para gerar outra linha vazia.

No lugar de ver se contém algo, ele precisa capturar. Para coletar um elemento, vamos adi comando, `cy.get` . Entre seus parênteses, digitaremos o elemento entre aspas simples: o colchetes, e entre estes passaremos `name="nome"` para informar o atributo que ele deve pe

Com isso, ele vai capturar o elemento. Em seguida, precisamos que ele digite um nome na isso, vamos utilizar outro comando no final dessa linha, o `.type` e um par de parênteses. insira o texto, vamos colocar entre os parênteses um nome qualquer entre aspas simples - Jesus.

Finalizaremos a linha com ponto e vírgula e salvaremos.

`spec.cy.js`

```
describe('template spec', () => {
    it('passes', () => {
        cy.visit('https://adopet-frontend-cypress.vercel.app/');
        cy.contains('a', 'Cadastrar').click();
        cy.get('input[name="nome"]').type('Ana de Jesus');
     })
})
```

Vamos voltar para o Cypress e ver que ele adicionou rapidamente o texto "Ana de Jesus" no

Vamos rodar o teste novamente com a letra "R" pela interface e notaremos no "TEST BODY escrever o `Ana de Jesus` na página web.

TEST BODY

visit https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/

-contains a, Cadastrar

-click

(page load) --page loaded--

(new url) https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/cadastro

get input[name="nome"]

-type Ana de Jesus

Precisamos realizar esse mesmo processo para os outros campos.

Voltando para o VS Code, vamos continuar o código com um trecho que já deixamos pront tempo.

```
cy.get('input[name="email"]').type('ana@email.com');
cy.get('input[name="password"]').type('Senha123');
cy.get('input[name="confirm_password"]').type('Senha123');
cy.contains('button', 'Cadastrar').click();
```

Esse código possui os outros campos. Veremos o `cy.get` no `email`, no `password`, na `senl` `confirm_password`, assim como o clique no elemento `button` que contém o texto de `Cada`

Após colar esse trecho abaixo do último `cy.get` que tínhamos, o resultado pode ser visto a

spec.cy.js

```
describe('template spec', () => {
    it('passes', () => {
        cy.visit('https://adopet-frontend-cypress.vercel.app/');
        cy.contains('a', 'Cadastrar').click();
        cy.get('input[name="nome"]').type('Ana de Jesus');
```

```
        cy.get('input[name="email"]').type('ana@email.com');
        cy.get('input[name="password"]').type('Senha123');
        cy.get('input[name="confirm_password"]').type('Senha123');
        cy.contains('button', 'Cadastrar').click();
    })
})
```

Salvaremos o código e abriremos o terminal integrado do VS Code com o atalho "Ctrl+'". Va terminal com o comando `clear` e realizar o `npx cypress open` para abrir o Cypress.

Após a aba do Cypress ser aberta, vamos acessar novamente a tela de início, escolher o br "Start E2E Testing in Chrome".

Vamos fazer a leitura desse arquivo. Na tela "Specs", clicaremos no "spec.cy.js". Com isso, e e rodará o teste.

Após a execução, notaremos que ele conseguirá preencher os campos do formulário e clic efetuando o cadastro da Ana de Jesus com sucesso.

O que aconteceu no "TEST BODY"? Ele visitou a página inicial, depois encontrou o elemen "Cadastrar", clicou nesse elemento, carregou a página de cadastro e capturou os elementos

Em seguida, digitou os textos que passamos por parâmetro no código do VS Code. E ele fez de `email`, para o campo `password` com a senha `Senha123` e para o `confirm_password`, co

Por fim, ele procurou novamente um elemento botão, o `button`, com o texto "Cadastrar", elemento.

TEST BODY

visit https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/

-contains a, Cadastrar

-click

(page load) --page loaded--

(new url) https://adopet-frontend- (https://adopet-frontend-) cypress.vercel.app/cadastrc

get input[name="nome"]

-type Ana de Jesus

get input[name="email"]

-type ana@email.com

get input[name="password"]

-type Senha123

get input[name="confirm_password"]

-type Senha123

-contains button, Cadastrar

-click

(xhr) POST 201 https://adopet-api-i8qu.onrender.com/adotante/register (https://adopet-api-i8qu.onrender.com/adotante/register)

(new url) https://adopet-frontend-cypress.vercel.app/login (https://adopet-frontend-cypress.vercel.app/login)

É muito interessante que ele também trouxe uma resposta para nós, um `POST 201` na URI informando que ele conseguiu fazer esse registro ou cadastro.

Depois, ele redirecionou para a página de login automaticamente, pois esse é o comportar cadastro.

Nosso teste funcionou!

Vamos voltar para o VS Code para interromper o Cypress no terminal com "Ctrl+C". Vamos que queremos finalizar.

Por fim, vamos fechar o terminal integrado.

O que conseguimos entender desse código? O Cypress, que utiliza o JavaScript internamer **DOM (Document Object Model)** da página. Ele captura os elementos dessa árvore de DOM comandos que solicitamos — nesse caso, solicitamos que ele escrevesse textos.

Conseguimos criar o teste que simula o comportamento de uma pessoa usuária na página Adopet. No próximo vídeo, vamos entender como podemos melhorar esse teste que escre lá!