08

Para saber mais: outras formas de buscar elementos

No decorrer do curso, nós focamos nas boas práticas da React Testing Library e utilizamos `roles` para buscar os elementos HTML. Porém, essa não é a única forma de realizar buscas no DOM.

Vou organizar aqui algumas que você pode precisar, mas não deixe de conferir a documentação (https://testing-library.com/docs/queries/about) se precisa de algo mais específico.

Esse são os tipos de consultas (em inglês, *queries*) disponíveis, de acordo com a quantidade de elementos que você espera:

Buscando elementos únicos:

- getBy
- queryBy
- findBy

Buscando múltiplos elementos:

- getAllBy
- queryAllBy
- findAllBy

E, aqui, você consegue ver as consultas específicas:

ByRole

getByRole, queryByRole, getAllByRole, queryAllByRole, findByRole, findAllByRole

Essa query você já conhece bem, pois foi a que mais utilizamos durante o curso. Quando conseguimos encontrar elementos por sua role, significa que leitores de tela também serão capazes, e a aplicação estará acessível.

ByLabelText

getByLabelText, queryByLabelText, getAllByLabelText, queryAllByLabelText, findByLabelText, findAllByLabelText

É uma alternativa quando, por exemplo, temos múltiplos inputs no mesmo componente. Nesse caso, podemos buscar o input pelo **label** associado a ela. Confira a seguir boas práticas para esse cenário:

```
<label for="email-input">E-mail</label>
<input id="email-input" />

<label id="email-label">E-mail</label>
<input aria-labelledby="email-label" />

<label>E-mail <input /></label>

<label>
  <span>E-mail</span>
  <input />
</label>

<input aria-label="E-mail" />
```

COPIAR CÓDIGO

Em qualquer um dos exemplos acima, podemos obter o input da seguinte forma:

```
const inputEmail = screen.getByLabelText('E-mail')
```

COPIAR CÓDIGO

ByPlaceholderText

getByPlaceholderText, queryByPlaceholderText, getAllByPlaceholderText, queryAllByPlaceholderText, findByPlaceholderText, findAllByPlaceholderText

Aqui, estamos falando do atributo do html, que também podemos utilizar para buscar os elementos:

```
<input placeholder="E-mail" />
```

COPIAR CÓDIGO

E para buscar:

```
const inputEmail = screen.getByPlaceholderText('E-mail')
```

COPIAR CÓDIGO

ByText

getByText, queryByText, getAllByText, queryAllByText, findByText, findAllByText

Agora, estamos falando do `textContent` dos elementos HTML. Então, para buscar uma âncora com essa marcação:

```
<a href="/sobre">Sobre a Alura</a>
```

COPIAR CÓDIGO

Escrevemos:

```
const ancoraSobre = screen.getByText('Sobre a Alura')
```

COPIAR CÓDIGO

É comum também o uso de Expressões Regulares para fazer buscas por texto. Essas expressões são tão poderosas e úteis que existe até um curso (https://www.alura.com.br/curso-online-expressoes-regulares) sobre elas.

ByDisplayValue

getByDisplayValue, queryByDisplayValue, getAllByDisplayValue, queryAllByDisplayValue, findByDisplayValue, findAllByDisplayValue

Podemos utilizar essa query para encontrar elementos do tipo input, textarea, ou select.

O "display value" é o valor em si do elemento.

Se liga aqui no exemplo:

```
<input type="text" id="nome" value=''Agarikov'' />
```

COPIAR CÓDIGO

Agora com o valor preenchido, podemos realizar a query:

```
const nomeInput = screen.getByDisplayValue('Agarikov’)
```

COPIAR CÓDIGO

ByTestId

getByTestId, queryByTestId, getAllByTestId, queryAllByTestId, findByTestId, findAllByTestId

Quando tudo mais falhar, podemos definir um atributo `data-testid` no elemento e utilizar esse tipo de query para encontrá-lo:

```
<div data-testid="id-customizado" />
```

COPIAR CÓDIGO

```
const elemento = screen.getByTestId(‘id-customizado')
```

COPIAR CÓDIGO

Mas fique atento a isto, pois aqui estamos sem valor semântico nenhum. Isso prejudica leitores de tela e demais softwares de acessibilidade.