

Cypress: testes de ponta a ponta



DIM0512 – TESTE DE SOFTWARE II

Baixar o Cypress

Entrar em
www.cypress.io



Test. Automate. Accelerate.

With Cypress, you can easily create tests for your modern web applications, debug them visually, and automatically run them in your continuous integration builds.

`> npm install cypress`

[Compare plans](#)



Direct Download

Download the Cypress application directly to give it a try:

[Download Cypress.zip](#)



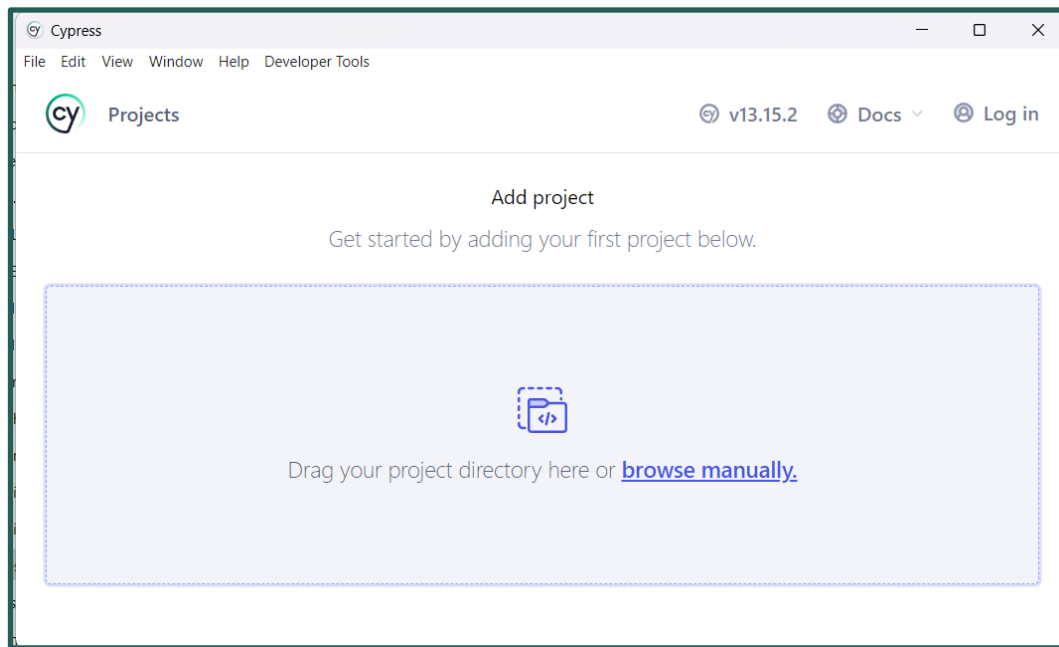
`npx cypress`

Cypress x Selenium

<i>Parâmetro</i>	<i>Cypress</i>	<i>Selenium WebDriver</i>
<i>Linguagem dos Scripts</i>	Javascript e linguagens que transpilam para Javascript	Javascript, Typescript, Java, Python, etc
<i>Paralelização</i>	Somente em múltiplas máquinas, não é possível na mesma máquina	É possível paralelizar tanto na mesma máquina, quanto em máquinas diferentes
<i>Eventos de DOM</i>	É capaz de interagir com eventos de DOM	Não é capaz de escutar automaticamente eventos do DOM

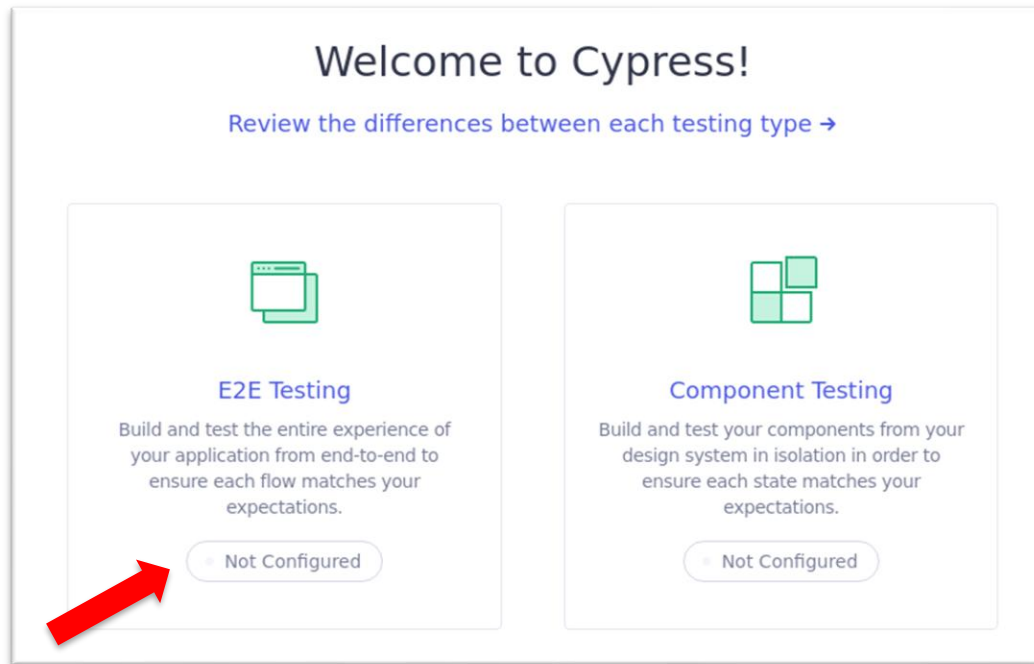
Abrindo o Cypress

- Criar uma pasta para o projeto
- Abrir pelo Cypress



Configurando Projeto

Ao abrir a pasta, se a pasta não tiver os arquivos de configuração, o primeiro passo é configurá-los clicando em “E2E Testing”



Configurando Projeto

Configuration files

We added the following files to your project:



[cypress.config.js](#)

The Cypress config file for E2E testing.



[cypress/support/e2e.js](#)

The support file that is bundled and loaded before each E2E spec.



[cypress/support/commands.js](#)

A support file that is useful for creating custom Cypress commands and overwriting existing ones.



[cypress/fixtures/example.json](#)

Added an example fixtures file/folder



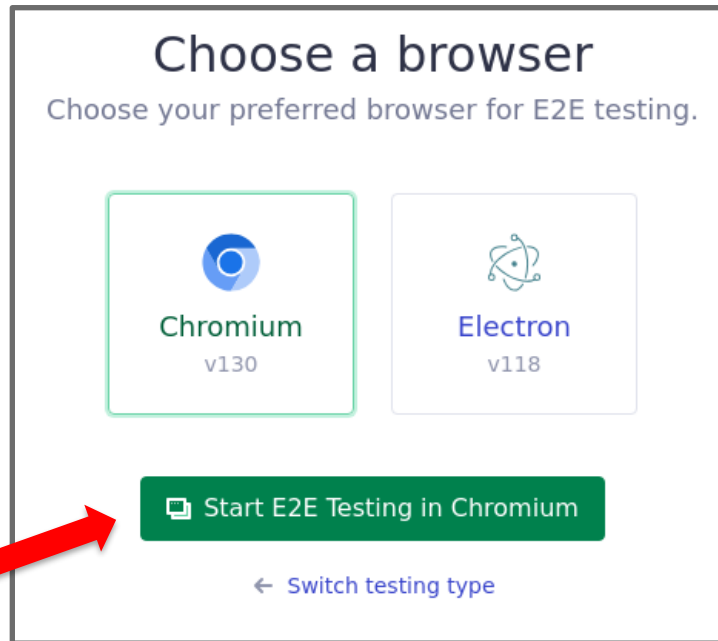
Continue



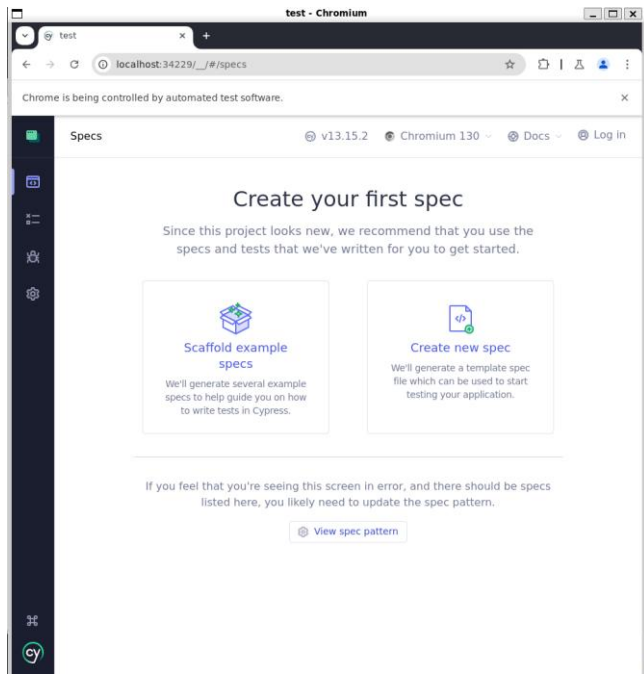
Configurando Projeto

O próximo passo é escolher qual o motor do navegador que você quer utilizar para realizar os testes de ponta a ponta.

Após a escolha, basta clicar em “Start E2E Testing in Chromium”



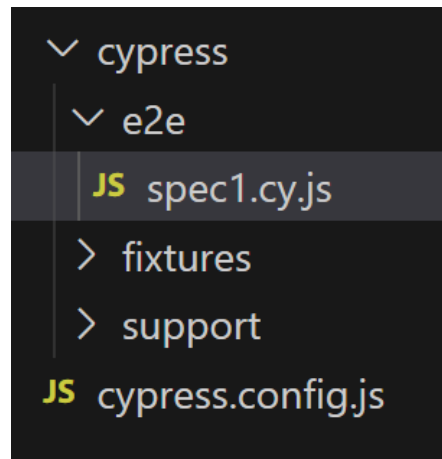
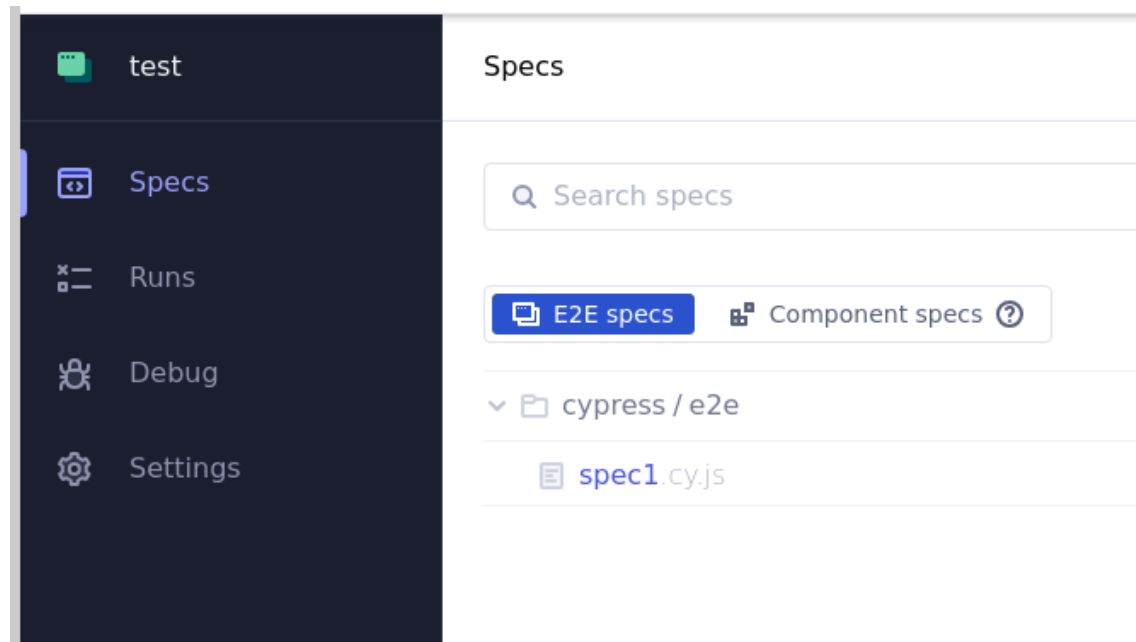
Criar novo script de teste



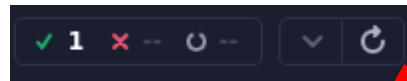
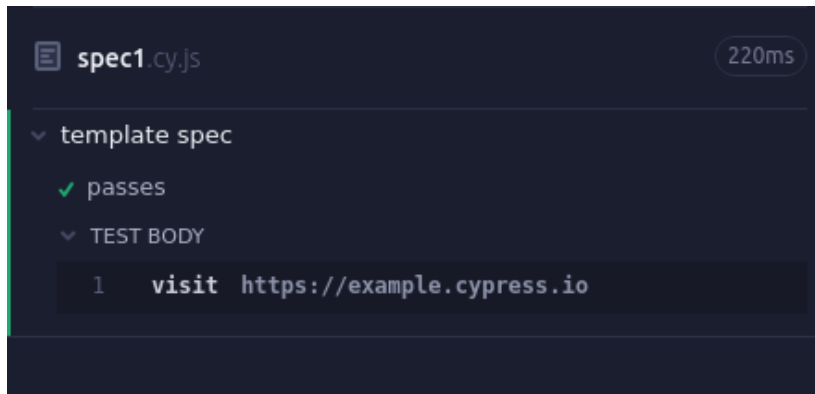
Create new spec

We'll generate a template spec file which can be used to start testing your application.

Criar novo script de teste

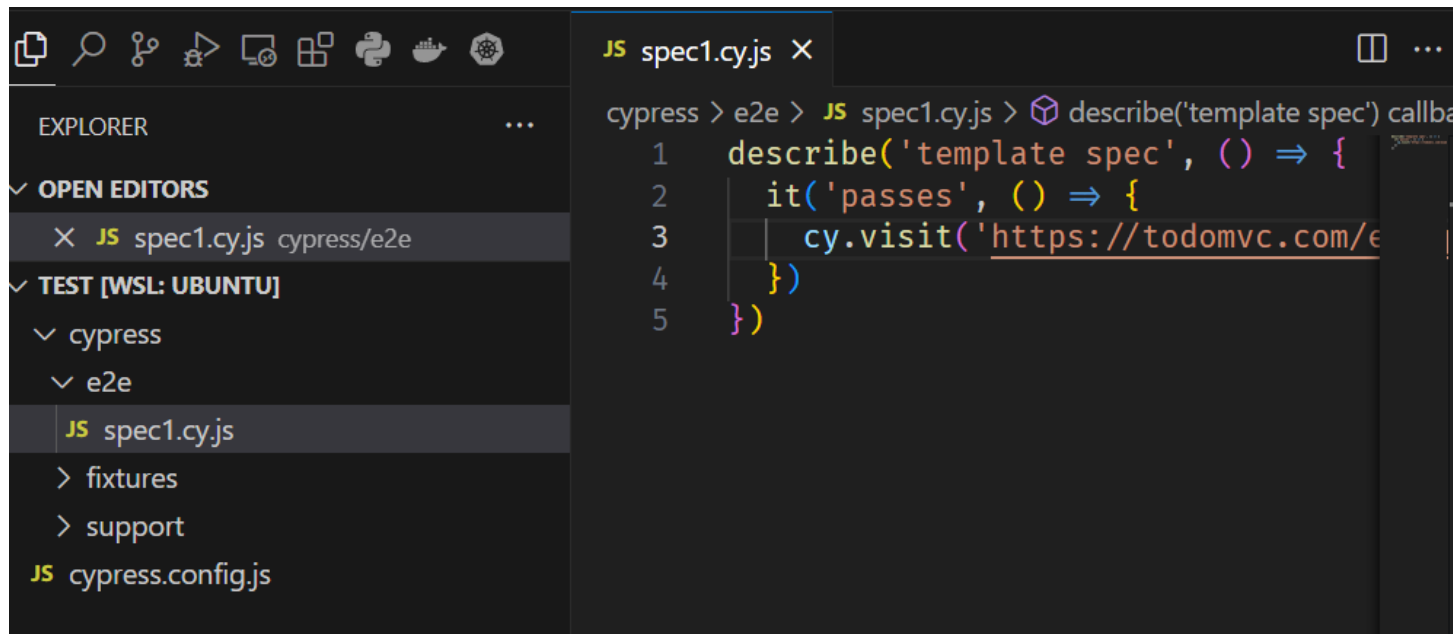


O script



```
cypress > e2e > JS spec1.cy.js > ...  
1 describe('template spec', () => {  
2   it('passes', () => {  
3     cy.visit('https://example.cypress.io')  
4   })  
5 })
```

Abrir pasta na IDE favorita





Conceitos Fundamentais

Anatomia dos Specs



1. describe, context, it, specify, Only
2. Hooks: before, beforeEach, after, afterEach

```
describe('Unit test our math functions', () => {  
  context('math', () => {  
    it('can add numbers', () => {  
      expect(add(1, 2)).to.eq(3)  
    })  
  
    it('can subtract numbers', () => {  
      expect(subtract(5, 12)).to.eq(-7)  
    })  
  
    specify('can divide numbers', () => {  
      expect(divide(27, 9)).to.eq(3)  
    })  
  
    specify('can multiply numbers', () => {  
      expect(multiply(5, 4)).to.eq(20)  
    })  
  })  
})
```

Query e Assertions

Queries e Implicit Assertions



```
cy.visit('/home')  
  
cy.get('.main-menu').contains('New Project').click()  
  
cy.get('.title').type('My Awesome Project')  
  
cy.get('form').submit()
```

Query e Assertions

Explicit Assertions


[Assertions | Cypress Documentation](#)

[Table of Contents | Cypress Documentation](#)

```
describe('Unit test our math functions', () => {  
  context('math', () => {  
    it('can add numbers', () => {  
      expect(add(1, 2)).to.eq(3)  
    })  
  
    it('can subtract numbers', () => {  
      expect(subtract(5, 12)).to.eq(-7)  
    })  
  
    specify('can divide numbers', () => {  
      expect(divide(27, 9)).to.eq(3)  
    })  
  
    specify('can multiply numbers', () => {  
      expect(multiply(5, 4)).to.eq(20)  
    })  
  })  
})
```

Referências para explorar mais

Consulta de elementos, cadeia de comandos, assertions

- 
1. <https://docs.cypress.io/app/core-concepts/introduction-to-cypress>
 2. <https://docs.cypress.io/app/core-concepts/interacting-with-elements>

Prática usando o Cypress

O exercício consiste em fazer um arquivo de spec que realize os seguintes testes na aplicação TodoMVC
(<https://todomvc.com/examples/angular/dist/browser>):

1. Verificar se a página está apresentando os elementos de título h1 (com valor "todos") e input (com placeholder "what needs to be done?")
2. Verificar se o TodoMVC está inserindo uma tarefa
3. Verificar se uma tarefa recém criada aparece no filtro "Active"
4. Verificar se o botão "Clear completed" está funcionando

Exercício valendo ponto

<https://todomvc.com/examples/angular/dist/browser>

- Criar um conjunto de testes que procurem exercitar a aplicação de maneira eficiente, utilizando as técnicas aprendidas nas disciplinas de testes. O conjunto de testes vai ser avaliado através de testes de mutação
- Entregar em arquivo zip, no seguinte formato: “<matricula>.zip”.
- Deve conter somente arquivos “*.cy.js”, e todos eles devem ser criados a partir do **template**: [link](#).
- **Entrega:** 19/01