## **Índice**

1. [Introdução](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#1-introdu%C3%A7%C3%A3o)
2. [O que é o Cypress?](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#2-o-que-%C3%A9-o-cypress)
3. [Por que Usar Cypress?](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#3-por-que-usar-cypress)
4. [Instalação e Configuração](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#4-instala%C3%A7%C3%A3o-e-configura%C3%A7%C3%A3o)
5. [Estrutura de Projeto](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#5-estrutura-de-projeto)
6. [Comandos Básicos](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#6-comandos-b%C3%A1sicos)
7. [Escrevendo o Primeiro Teste](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#7-escrevendo-o-primeiro-teste)
8. [Boas Práticas](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#8-boas-pr%C3%A1ticas)
9. [Exemplos Práticos](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#9-exemplos-pr%C3%A1ticos)
10. [Integração com CI/CD](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#10-integra%C3%A7%C3%A3o-com-cicd)
11. [Recursos Adicionais](http://127.0.0.1:55898/c/019a118e-7cbf-7118-8ff4-c7bb59990028#11-recursos-adicionais)

## **1. Introdução**

### **1.1 Objetivo deste Documento**

Este documento tem como objetivo fornecer uma base sólida para iniciar a automação de testes utilizando o Cypress, abordando tanto aspectos técnicos para desenvolvedores quanto conceitos fundamentais para não-desenvolvedores.

### **1.2 Público-Alvo**

* **Desenvolvedores:** Que desejam implementar testes automatizados
* **QAs Não-Técnicos:** Que precisam entender o processo de automação
* **Gestores de Projeto:** Que buscam compreender os benefícios da automação

### **1.3 Pré-requisitos**

* Conhecimento básico de JavaScript
* Node.js instalado (versão 14 ou superior)
* Editor de código (VS Code recomendado)

## **2. O que é o Cypress?**

### **2.1 Definição**

O Cypress é um framework de teste **end-to-end** moderno para aplicações web que executa testes diretamente no navegador.

### **2.2 Características Principais**

| **Característica** | **Descrição** |
| --- | --- |
| **Tempo Real** | Visualização dos testes executando em tempo real |
| **Debug Fácil** | Ferramentas integradas de debugging |
| **Configuração Simples** | Setup rápido e intuitivo |
| **Velocidade** | Execução mais rápida que soluções tradicionais |

## **3. Por que usar Cypress?**

### **3.1 Vantagens para Desenvolvedores**

* **API Consistente:** Comandos previsíveis e confiáveis
* **Debug Integrado:** Ferramentas de debug no próprio navegador
* **Wait Automático:** Não precisa de waits explícitos na maioria dos casos
* **Snapshot e Video:** Captura automática de screenshots e vídeos

### **3.2 Vantagens para Não-Desenvolvedores**

* **Interface Visual:** Test Runner com interface amigável
* **Relatórios Claros:** Resultados fáceis de interpretar
* **Feedback Imediato:** Ver testes executando em tempo real
* **Documentação Viva:** Testes como documentação do sistema

### **3.3 Casos de Uso Ideais**

* Testes de interface de usuário (UI)
* Testes de fluxos completos (end-to-end)
* Testes de APIs REST
* Testes de integração

## **4. Instalação e Configuração**

### **4.1 Passo a Passo de Instalação**

bash

# 1. Criar novo projeto (se necessário)

mkdir meus-testes-cypress

cd meus-testes-cypress

# 2. Inicializar projeto Node.js

npm init -y

# 3. Instalar Cypress

npm install cypress --save-dev

# 4. Abrir Cypress (interface gráfica)

npx cypress open

# 5. Executar em modo headless (CI/CD)

npx cypress run

### **4.2 Arquivo de Configuração (cypress.config.js)**

javascript

const { defineConfig } = require('cypress')

module.exports = defineConfig({

e2e: {

baseUrl: 'https://meusite.com',

setupNodeEvents(on, config) {

// plugins aqui

},

},

})

## **5. Estrutura de Projeto**

### **5.1 Organização de Pastas**

**Padrão Cypress puro**

cypress/

├── e2e/ # Testes end-to-end

│ ├── login.cy.js

│ └── cadastro.cy.js

├── fixtures/ # Dados de teste

│ └── usuarios.json

├── support/ # Configurações e comandos customizados

│ ├── commands.js

│ └── e2e.js

└── downloads/ # Arquivos baixados durante testes

[**Padrão Monorepo**](?tab=t.o9ijbb7leyb) **(Cypress + Cucumber)**

### **5.2 Descrição das Pastas**

* **e2e:** Contém os arquivos de teste (.cy.js / .feature)
* **fixtures:** Dados estáticos para mock (ex: usuários, produtos)
* **support:** Configurações globais e comandos personalizados
* **downloads:** Arquivos baixados durante a execução dos testes

## **6. Comandos Básicos**

### **6.1 Comandos de Navegação**

javascript

| cy.visit('/pagina') // Navega para uma URL  cy.go('back') // Volta para página anterior  cy.reload() // Recarrega a página atual  cy.url().should('include', '/login') // Verifica URL |
| --- |

### **6.2 Comandos de Interação**

javascript

| cy.get('#email').type('usuario@teste.com') // Digita texto  cy.get('#botao').click() // Clica em elemento  cy.get('.lista').select('Opção 1') // Seleciona opção  cy.get('#arquivo').attachFile('doc.pdf') // Anexa arquivo |
| --- |

### **6.3 Comandos de Assertação**

javascript

| .should('be.visible') // Elemento visível  .should('contain', 'Texto') // Contém texto  .should('have.value', '123') // Tem valor específico  .should('be.disabled') // Está desabilitado |
| --- |

### **6.4 Localização de Elementos**

javascript

| // Por ID  cy.get('#submit-btn')  // Por Classe  cy.get('.botao-primario')  // Por Atributo  cy.get('[data-testid="user-name"]')  // Por Texto  cy.contains('Cadastrar') |
| --- |

## **7. Escrevendo o Primeiro Teste**

### **7.1 Estrutura Básica de um Teste**

javascript

| describe('Suite de Teste - Login', () => {  beforeEach(() => {  // Executa antes de cada teste  cy.visit('/login')  })  it('Deve fazer login com credenciais válidas', () => {  // Arrange (Preparação)  cy.get('#email').type('usuario@teste.com')  cy.get('#senha').type('senha123')    // Act (Ação)  cy.get('#login-btn').click()    // Assert (Verificação)  cy.url().should('include', '/dashboard')  cy.get('.welcome-message').should('contain', 'Bem-vindo')  })  it('Deve mostrar erro com credenciais inválidas', () => {  // Teste de cenário negativo  cy.get('#email').type('invalido@teste.com')  cy.get('#senha').type('errada')  cy.get('#login-btn').click()  cy.get('.error-message').should('be.visible')  })  }) |
| --- |

### **7.2 Hooks (Ganchos) de Teste**

javascript

| before(() => {}) // Executa uma vez antes de todos os testes  beforeEach(() => {}) // Executa antes de cada teste  afterEach(() => {}) // Executa após cada teste  after(() => {}) // Executa uma vez após todos os testes |
| --- |

## **8. Boas Práticas**

### **8.1 Para Desenvolvedores**

#### **8.1.1 Page Object Pattern**

javascript

| // cypress/support/pages/LoginPage.js  class LoginPage {  elements = {  emailInput: () => cy.get('#email'),  passwordInput: () => cy.get('#senha'),  loginButton: () => cy.get('#login-btn'),  errorMessage: () => cy.get('.error')  }  visit() {  cy.visit('/login')  }  fillEmail(email) {  this.elements.emailInput().type(email)  }  fillPassword(password) {  this.elements.passwordInput().type(password)  }  submit() {  this.elements.loginButton().click()  }  }  export default new LoginPage() |
| --- |

#### **8.1.2 Comandos Customizados**

javascript

// cypress/support/commands.js

Cypress.Commands.add('login', (email, password) => {

cy.visit('/login')

cy.get('#email').type(email)

cy.get('#senha').type(password)

cy.get('#login-btn').click()

})

// Uso no teste

cy.login('usuario@teste.com', 'senha123')

### **8.2 Para Não-Desenvolvedores**

#### **8.2.1 Nomenclatura Clara**

javascript

// Errado

it('teste1', () => {})

// Correto

it('deve exibir mensagem de erro ao tentar login com senha incorreta', () => {})

#### **8.2.2 Organização Lógica**

* Agrupar testes relacionados em mesma suite (describe)
* Ordem lógica: setup → ação → verificação
* Comentários explicativos para steps complexos

## **9. Exemplos Práticos**

### **9.1 Teste de Cadastro de Usuário**

javascript

describe('Cadastro de Usuário', () => {

it('Deve cadastrar novo usuário com sucesso', () => {

cy.visit('/cadastro')

// Preenche formulário

cy.get('#nome').type('João Silva')

cy.get('#email').type('joao@teste.com')

cy.get('#senha').type('Senha123!')

cy.get('#confirmar-senha').type('Senha123!')

// Submete formulário

cy.get('#cadastrar-btn').click()

// Verificações

cy.url().should('include', '/cadastro-sucesso')

cy.get('.success-message').should('contain', 'Cadastro realizado')

})

})

### **9.2 Teste de API**

javascript

describe('API Tests', () => {

it('Deve retornar lista de usuários', () => {

cy.request('GET', '/api/users')

.then((response) => {

expect(response.status).to.eq(200)

expect(response.body).to.have.property('users')

expect(response.body.users).to.have.length.greaterThan(0)

})

})

})

## **10. Integração com CI/CD**

### **10.1 Exemplo GitHub Actions**

| yaml  name: Cypress Tests  on: [push]  jobs:  cypress-run:  runs-on: ubuntu-latest  steps:  - name: Checkout code  uses: actions/checkout@v3    - name: Setup Node.js  uses: actions/setup-node@v3  with:  node-version: '18'    - name: Install dependencies  run: npm install    - name: Run Cypress tests  run: npx cypress run --headless |
| --- |

### **10.2 Comandos para CI/CD**

bash

| # Executar todos os testes  npx cypress run  # Executar testes específicos  npx cypress run --spec "cypress/e2e/login.cy.js"  npx cypress run --spec "cypress/e2e/login.feature"  # Executar em navegador específico  npx cypress run --browser chrome  # Gerar report HTML  npx cypress run --reporter junit |
| --- |

## **11. Recursos Adicionais**

### **11.1 Documentação Oficial**

* [Documentação Cypress](https://docs.cypress.io/)
* [Guia de Introdução](https://docs.cypress.io/guides/getting-started/writing-your-first-test)
* [API Reference](https://docs.cypress.io/api/table-of-contents)

### **11.2 Ferramentas Recomendadas**

* **Cypress Testing Library:** Para testes mais acessíveis
* **Cypress Dashboard:** Para relatórios e analytics
* **Cypress Studio:** Para gravação de testes visuais

### **11.3 Próximos Passos**

1. Implementar testes para funcionalidades críticas
2. Integrar com pipeline de CI/CD
3. Configurar relatórios automatizados
4. Implementar Page Objects pattern
5. Adicionar testes de API

## **Anexo A: Glossário de Termos**

| **Termo** | **Definição** |
| --- | --- |
| **E2E Test** | Teste que simula fluxo completo de usuário |
| **Assertion** | Verificação de que algo é verdadeiro |
| **Selector** | Modo de localizar elementos na página |
| **Fixture** | Dados estáticos para testes |
| **Hook** | Função que executa em momentos específicos |

## **Anexo B: Checklist de Implementação**

* Cypress instalado e configurado
* Estrutura de pastas criada
* Primeiro teste escrito e funcionando
* Page Objects implementados (opcional)
* Integração com CI/CD configurada
* Relatórios de teste implementados
* Timeout e configurações ajustadas
* Testes críticos da aplicação cobertos