


## Plano de testes Automation Exercise

Abaixo está um **plano de testes completo e desafiador** para você praticar automação com **Cypress**, abrangendo tanto testes de **interface (UI)** quanto de **API (GET e POST)** usando o site [Automation Exercise](#). Esse plano é dividido em **módulos**, com **casos de teste detalhados** e sugestões para implementar com **Page Object Model (POM)** e **comandos customizados**.

## Visão Geral do Projeto

**Objetivo:** Testar o fluxo completo de um e-commerce fictício: cadastro, login, compra de produto, envio de mensagem e validação via API.



### Configuração Inicial (Cypress)

Antes de mais nada, no seu projeto Cypress:

bash

```
npm install cypress --save-dev  
npx cypress open
```

Copiar

Editar

Adicione o seguinte no `cypress.config.js` (ou `cypress.json` se for versão antiga):

js


```
baseUrl: "https://www.automationexercise.com",
```

Copiar

Editar

## Estrutura Sugerida (Page Object)

bash

 Copiar

 Editar

```
cypress/
|
├─ e2e/
|   └─ userFlow.cy.js
|   └─ apiTests.cy.js
|
├─ support/
|   └─ commands.js
|   └─ e2e.js
|
└─ pageObjects/
    └─ homePage.js
    └─ loginPage.js
    └─ signUpPage.js
    └─ productsPage.js
    └─ cartPage.js
    └─ checkoutPage.js
    └─ contactPage.js
```

## ✓ Plano de Testes: UI + API

### ◆ Módulo 1: Cadastro e Login

#### Caso de Teste 1: Cadastro de novo usuário

- Acessar página inicial
- Clicar em *Signup* / *Login*
- Preencher nome e e-mail
- Preencher o formulário completo de cadastro
- Verificar se está logado como o nome do usuário

#### Caso de Teste 2: Login com usuário válido e inválido

- Login com e-mail/senha válidos → esperar sucesso
- Login com e-mail/senha inválidos → esperar erro visível

---

### ◆ Módulo 2: Produtos

#### Caso de Teste 3: Visualizar lista de produtos

- Clicar em *Products*
- Verificar se todos os produtos são carregados (elementos visíveis)
- Validar que a busca por nome retorna o produto certo

#### Caso de Teste 4: Adicionar múltiplos produtos ao carrinho

- Adicionar 2 produtos
- Verificar que o carrinho contém ambos
- Validar nome, quantidade e valor de cada item

---

### 🔗 Módulo 3: Carrinho e Checkout

#### Caso de Teste 5: Checkout completo

- Login (caso necessário)
- Adicionar produtos ao carrinho
- Clicar em *Cart* → *Proceed to Checkout*
- Preencher dados de pagamento fictícios
- Finalizar pedido
- Verificar mensagem de sucesso

---

### 🔗 Módulo 4: Contato

#### Caso de Teste 6: Enviar mensagem de contato

- Clicar em *Contact Us*
- Preencher nome, e-mail, assunto e mensagem
- Fazer upload de um arquivo (usando `attachFile` do plugin)
- Validar mensagem de sucesso

---

### 🔗 Módulo 5: Logout e Exclusão

#### Caso de Teste 7: Logout e exclusão da conta

- Fazer logout
- Logar novamente
- Ir para configurações e excluir conta
- Validar que a conta foi excluída e o usuário é redirecionado

## Extras (Avançado)

- Teste responsivo com `cy.viewport()`
- Teste de performance com `cy.intercept()` para medir tempo de carregamento
- Validar campos obrigatórios com testes negativos
- Testar upload de arquivo na área de contato
- Criar comandos customizados em `commands.js` para login, cadastro etc.

# Testes de API (GET e POST)

A API de testes do site está em: <https://www.automationexercise.com/api>

## ◆ Teste API GET: Lista de produtos

js

Copiar

Editar

```
describe('GET - Lista de produtos', () => {  
  it('Deve retornar todos os produtos', () => {  
    cy.request('GET', '/api/productsList').then((response) => {  
      expect(response.status).to.eq(200);  
      expect(response.body.products.length).to.be.greaterThan(0);  
    });  
  });  
});
```

## ◆ Teste API POST: Cadastro de usuário

js

📄 Copiar

✎ Editar

```
describe('POST - Cadastro de usuário', () => {
  it('Deve criar novo usuário', () => {
    cy.request('POST', '/api/createAccount', {
      name: 'LucasQA',
      email: `lucas.qa.${Date.now()}@mail.com`,
      password: '123456',
      title: 'Mr',
      birth_date: '1',
      birth_month: 'January',
      birth_year: '1990',
      firstname: 'Lucas',
      lastname: 'QA',
      company: 'TestCorp',
      address1: 'Rua dos Testes',
      address2: 'QA Bloco',
      country: 'Canada',
      zipcode: '12345',
      state: 'SP',
      city: 'TestCity',
      mobile_number: '11999999999'
    }).then((response) => {
      expect(response.status).to.eq(200);
      expect(response.body).to.have.property('responseCode', 201);
    });
  });
});
```

💡 **Nota:** As APIs são limitadas e nem todas funcionam 100% com POST, mas servem bem para treino. Use `.log()` para ver respostas se necessário.