Documentação para Configuração e Execução de Testes com Cypress

1. Introdução:

Este documento descreve os passos necessários para configurar e executar testes automatizados na aplicação de demonstração Sauce Demo utilizando o framework Cypress. Os testes incluem funcionalidades como login, adição e remoção de produtos do carrinho e finalização de compras.

2. Requisitos:

Node.js (versão 14 ou superior)

npm (gerenciador de pacotes do Node)

Cypress (versão 13.15.1 ou superior)

3. Configuração do Ambiente

Passo 1: Instalação do Node.js e npm

Certifique-se de que o Node.js e o npm estão instalados em seu sistema. Você pode verificar isso executando os seguintes comandos no terminal:

```bash
node -v
npm -v
```

Passo 2: Criar um Novo Projeto

Crie uma pasta para o seu projeto e navegue até ela:

```bash
mkdir sauce-demo-cypress
cd sauce-demo-cypress
```

Passo 3: Inicializar o Projeto

Inicie um novo projeto Node.js:

```bash
npm init -y
```

Passo 4: Instalação do Cypress

Instale o Cypress como uma dependência de desenvolvimento:

```bash
npm install cypress --save-dev
```

Passo 5: Estrutura de Pastas

Após a instalação, crie a estrutura de pastas necessária para os testes:

```bash
mkdir -p cypress/{fixtures,e2e,plugins}
```

- fixtures: Armazena dados de teste (como arquivos JSON).
- e2e: Contém os arquivos de teste.
- plugins: Para scripts adicionais e integração.

Passo 6: Configuração do Cypress

Crie o arquivo de configuração cypress.config.js na raiz do projeto e configure a baseUrl:

```javascript
// cypress.config.js
const { defineConfig } = require('cypress');

module.exports = defineConfig({
  e2e: {
    baseUrl: 'https://www.saucedemo.com/v1', // URL base da aplicação
  },
});
```

4. Execução dos Testes

Passo 1: Abrir o Cypress

Para abrir a interface do Cypress, execute o seguinte comando:

```bash
npx cypress open
```

Passo 8: Para executar os testes em modo headless (sem interface), use:

```bash
npx cypress run
```

5. Estrutura do Código dos Testes

Os testes estão organizados na pasta cypress/e2e. Cada arquivo de teste representa uma funcionalidade da aplicação. Aqui está um exemplo de como um teste pode foi estruturado:

```javascript
import LoginPage from '../../pages/LoginPage';
import ProductsPage from '../../pages/ProductsPage';
import CartPage from '../../pages/CartPage';
import CheckoutPage from '../../pages/CheckoutPage';

describe('Adicionar e Remover do Carrinho', () => {
    const loginPage = new LoginPage();
    const productsPage = new ProductsPage();
    const cartPage = new CartPage();
    const checkoutPage = new CheckoutPage();
```

6. Durante os testes, foi identificado um cenário considerado um Bug, detalhado a seguir:

Descrição do Problema: O sistema permite que o usuário, ao acessar um carrinho de compras vazio, prossiga para o checkout e finalize a compra sem nenhum produto.

Passos para Reproduzir:

1. Estar logado no sistema.
2. Acessar o carrinho de compras.
3. Clicar em "Checkout."
4. Preencher as informações de contato e clicar em "Continue."
5. Ser direcionado para a página de validação das informações de compra.
6. Clicar em "Finish."

Comportamento Esperado: Ao acessar o carrinho de compras e este estiver vazio, o sistema deve impedir o usuário de prosseguir para as

etapas de checkout e exibir uma mensagem informando que o carrinho está vazio.

Evidências: Esse comportamento pode ser validado no teste automatizado "Tenta finalizar a compra sem produtos no carrinho" na suíte de testes do carrinho (`cart`).

Autor: Thiago Freitas