

ACADEMIA DE TESTES 2025 ACCENTURE

MARCOS FELIPE TENORIO CAVALCANTE

DESAFIO CYPRESS

Atividade de teste automatizado em cypress e cucumber

Recife
2025

MARCOS FELIPE TENORIO CAVALCANTE

DESAFIO CYPRESS

Desafio Cypress de criação de cenários de teste automatizado, utilizando metodologia BDD, ou Desenvolvimento Orientado pelo Comportamento (Behavior-Driven Development), linguagem Gherkin e padrão de projeto Page Object Model (POM), para o site <<https://demoqa.com/webtables>>. Fluxo de testes seguido foi de inserir cadastro no formulário, editar/atualizar o cadastro e por fim deletar o cadastro. Desafio será apresentado aos tutores Jonicler e Manoela

RESUMO

Neste documento é apresentado as configurações iniciais que devem ser aplicadas ao VS Code para que seja possível executar plenamente as features do projeto. O projeto encontra-se hospedado no GitHub no link <https://github.com/marcosftcavalcante/Automacao_Web-Academia_Accenture_2025> e com vídeo explicativo hospedado no YouTube no link <https://www.youtube.com/watch?v=i7mGxOev_Zg>

Lista de ilustrações

| | |
|--|----|
| Figura 1 – Terminal VS Code | 6 |
| Figura 2 – Versão Node.js | 6 |
| Figura 3 – Extensions | 7 |
| Figura 4 – Cucumber (Gherkin) Full Support | 8 |
| Figura 5 – Material Icon Theme | 8 |
| Figura 6 – comando para criar package.json | 9 |
| Figura 7 – comando para instalar cypress | 10 |
| Figura 8 – Cypress config | 11 |
| Figura 9 – package.json | 12 |

Sumário

| | |
|---|-----------|
| Lista de ilustrações | 4 |
| Sumário | 5 |
| 1 Configurações iniciais | 6 |
| 2 Apresentação técnica do Projeto de Automação de Testes de Web Tables | 13 |
| 2.1 Tecnologias Utilizadas | 13 |
| 2.2 Estrutura do Projeto | 13 |
| 2.3 Resumo dos Cenários de Teste | 13 |
| 2.3.1 Cadastro de Novo Funcionário | 13 |
| 2.3.2 Atualização de Cadastro de Funcionário | 14 |
| 2.3.3 Exclusão de Cadastro de Funcionário | 14 |
| 2.4 Arquivos de Implementação | 14 |
| 2.4.1 Comandos e Funções de Página | 14 |
| 2.4.2 Definições de Passo | 15 |
| 2.5 Como Executar os Testes | 15 |
| 2.5.1 Via Cypress Test Runner (UI): | 15 |
| 3 Apresentação acadêmica do Projeto de Automação de Testes de Web Tables | 15 |
| 3.1 Introdução | 15 |
| 3.2 Objetivos | 15 |
| 3.3 Metodologia | 16 |
| 3.3.1 Ferramenta Cypress | 16 |
| 3.3.2 BDD e Linguagem Gherkin | 16 |
| 3.3.3 Padrão de Projeto Page Object | 16 |
| 3.4 Implementação | 16 |
| 3.5 Resultados Obtidos | 17 |
| 3.6 Conclusão | 17 |

1 Configurações iniciais

Siga às configurações iniciais do ambiente para que seja possível replicar o repositório do GitHub em sua máquina e executar a feature sem problemas.

- Instale o Node.js acessando o link <<https://nodejs.org/en/download/>>.
- Após a instalação, no VS Code clicar em Terminal (ou nos três pontinhos horizontais e depois em Terminal) e em New Terminal. Ver figura 1.
- com o terminal aberto digite o comando "node -v" para verificar a versão do Node.js instalada. Ver figura 2

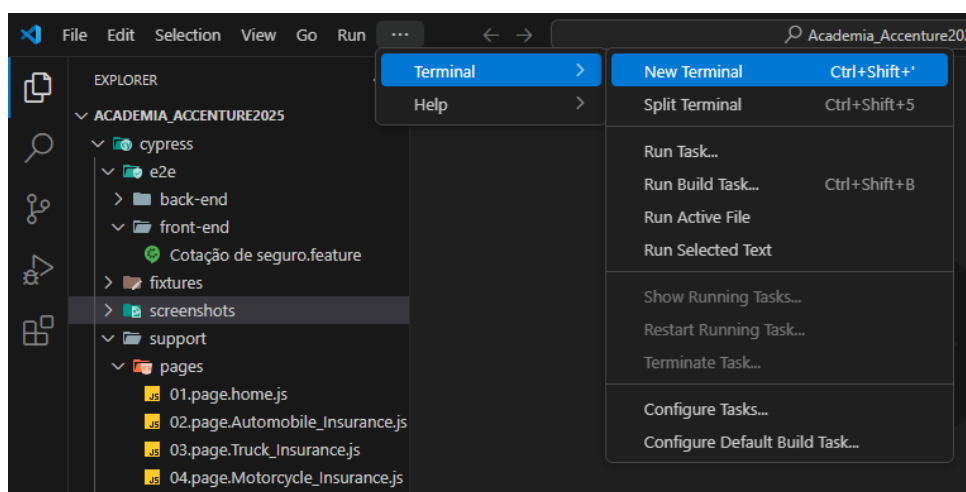


Figura 1 – Terminal VS Code

Fonte: Produzido pelo autor

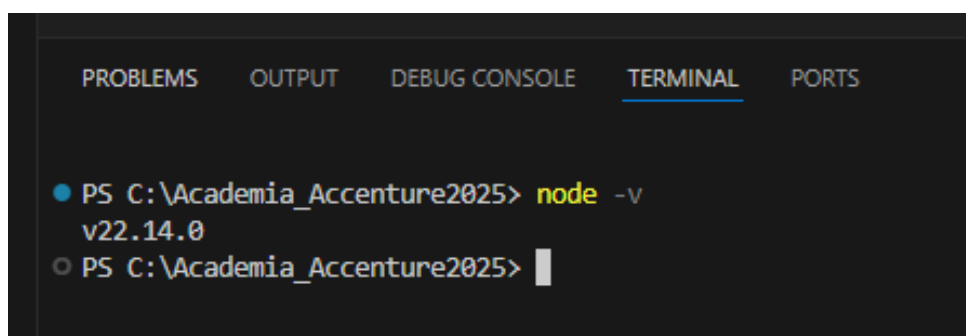


Figura 2 – Versão Node.js

Fonte: Produzido pelo autor

- Após a instalação do Node.js, caso não tenha o VS Code (Visual Studio Code) instalado em sua máquina, é possível realizar o download para instalação através do link <<https://code.visualstudio.com/>>.

- No VS Code, clicar no menu lateral esquerdo na opções Extensões, confirme mostra a figura 3.
- Na janela de Extensões, em Search buscar a extensão Cucumber (Gherkin) Full Support (figura 4) e Material Icon Theme (figura 5), instalar e ativar ambos.

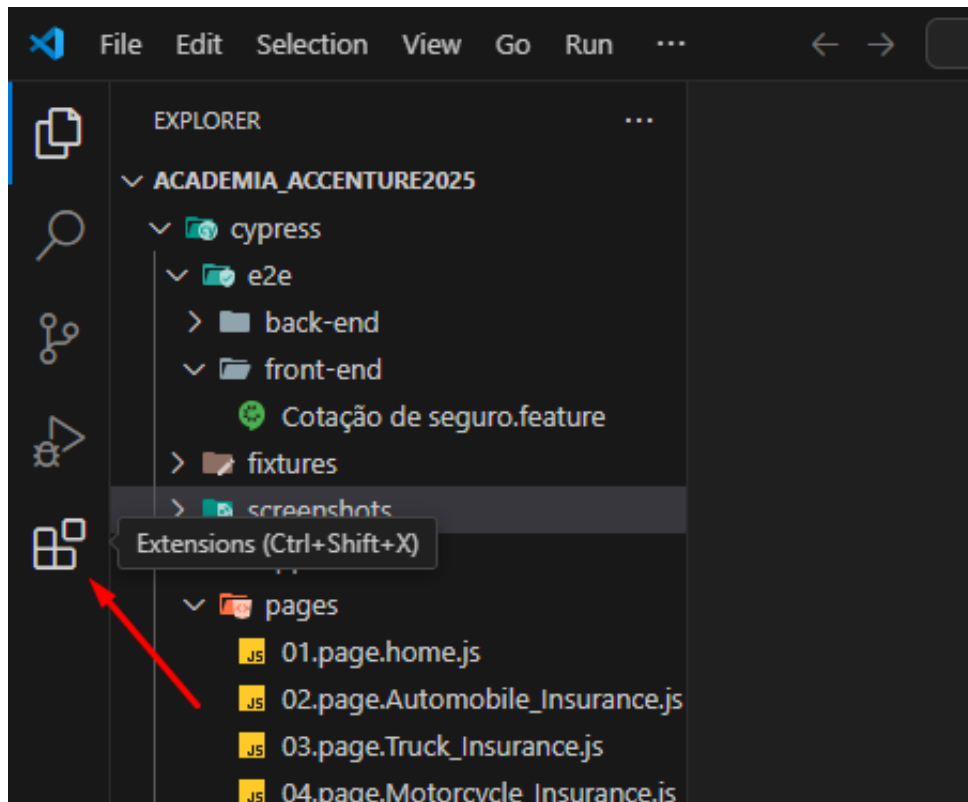


Figura 3 – Extensions

Fonte: Produzido pelo autor

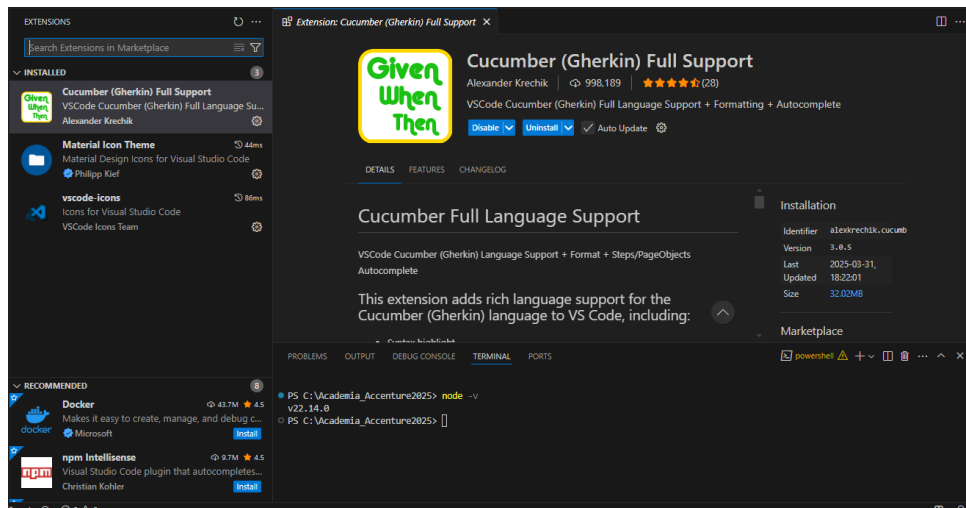


Figura 4 – Cucumber (Gherkin) Full Support

Fonte: Produzido pelo autor

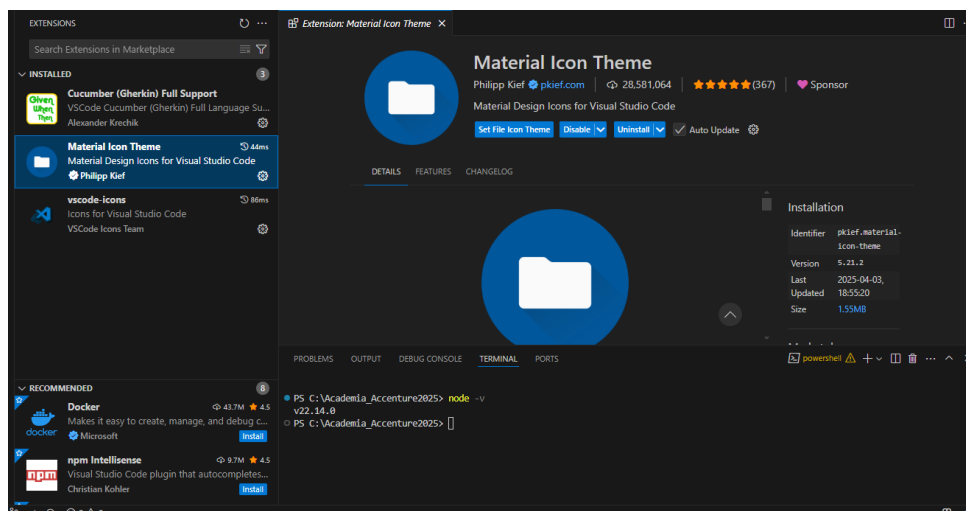


Figura 5 – Material Icon Theme

Fonte: Produzido pelo autor

- Na sequência vamos criar uma pasta na raiz do disco local C:\Automacao_Web-Academia_Accenture_2025
- No terminal do VS Code digitar `cd C:\Automacao_Web-Academia_Accenture_2025` para ser redirecionado até a pasta.
- No terminal, executar o comando `"npm init"` para criação do package.json e pressionar a tecla enter para cada linha nova que for impressa, até que apareça "Is This OK?(yes)", onde será necessário pressionar y e enter. A figura 6 ilustra essas etapas.
- No terminal executar o comando `"npm install cypress --save-dev"` para instalação do Cypress. A figura 7 ilustra esse passo.
- No Vs Code clicar em File > Open Folder e selecionar a pasta criada na raiz.


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\treinamento_cypress> cd C:\Academia_Accenture2025
● PS C:\Academia_Accenture2025> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (academia_accenture2025)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Academia_Accenture2025\package.json:

{
  "name": "academia_accenture2025",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}

Is this OK? (yes) y

○ PS C:\Academia_Accenture2025> █
```

Figura 6 – comando para criar package.json

Fonte: Produzido pelo autor

```
PS C:\Academia_Accenture2025> npm install cypress --save-dev

added 175 packages, and audited 176 packages in 21s

40 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Academia_Accenture2025> |
```

Figura 7 – comando para instalar cypress

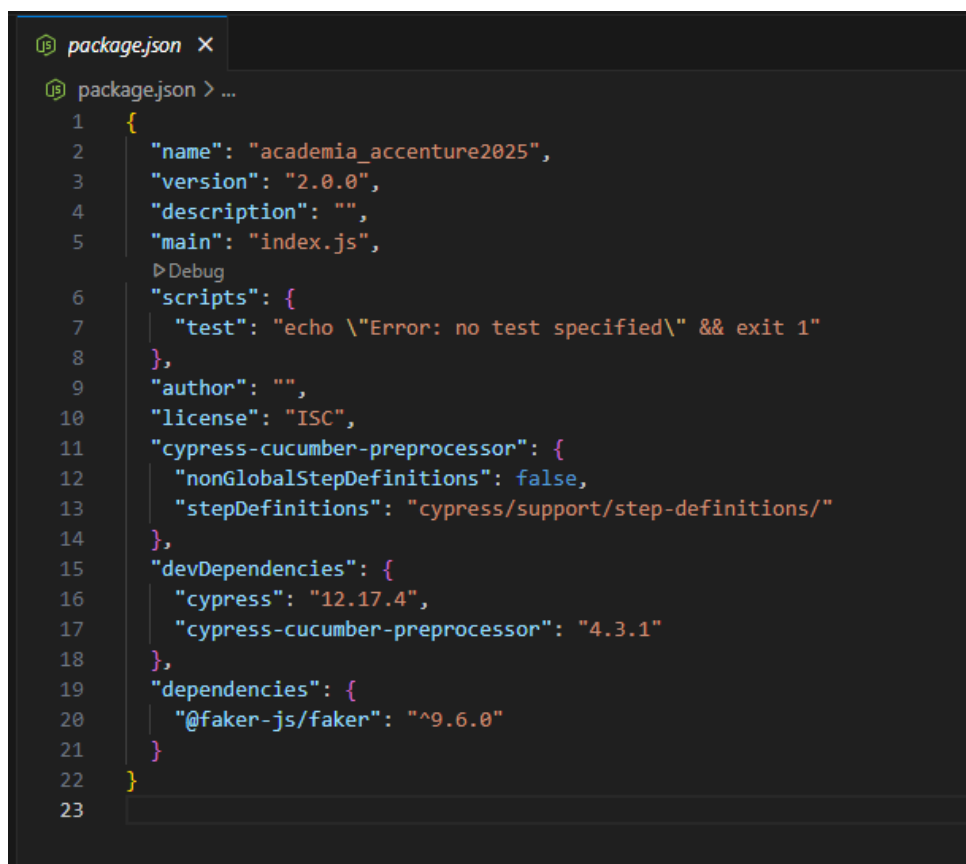
Fonte: Produzido pelo autor

- Executar o comando “npx cypress open” para abrir a interface do Cypress
- Clicar na opção “E2E Testing (Not Configured)” e confirmar a criação dos arquivos de configuração no VS Code.
- Criar uma pasta e2e dentro da pasta cypress. É onde iremos armazenar as features com o steps do Gherkin
- Criar uma pasta pages na pasta cypress/support. É onde guardaremos o arquivo de pages do Page Objects
- Editar o cypress.config.js para que fique como a figura 8. Estamos definindo o tamanho do display do navegador na interface do Cypress e garantindo que o teste do Cypress não sera reiniciado caso realizarmos mudanças nos arquivos. Ademais o comando "blockHosts" não necessário para todos os sites, ele foi desenvolvido especificamente para o site <https://demoqa.com/> para interceptar as Request de propaganda.
- Editar o package.json para que fique igual a figura 9
- Executar o comando “npm install -D cypress-cucumber-preprocessor” no terminal para instalação da biblioteca do cucumber preprocessor
- **atenção** Para este desafio, não se fez necessário o uso do Faker, pois o <https://demoqa.com/> não salva os dados após atualização, logo, todos os dados puderem se repetir n vezes, mas é de bom tom instalar a biblioteca faker através do comando "npm install @faker-js/faker" no terminal

```
cypress.config.js X
cypress.config.js > [e] <unknown> > e2e > blockHosts
1  const { defineConfig } = require("cypress");
2
3  module.exports = defineConfig({
4    e2e: {
5      video: true,
6      viewportWidth: 1920,
7      viewportHeight: 1080,
8      watchForFileChanges: false,
9      specPattern: 'cypress/e2e/**/*.feature',
10     baseUrl: 'https://demoqa.com/',
11
12     // Lista de hosts visualizados no log e bloqueados
13     blockHosts: [
14       '*google-analytics.com',
15       '*googlesyndication.com',
16       '*doubleclick.net',
17       '*googleadservices.com',
18       '*google.com',
19       '*analytics.google.com',
20       '*id5-sync.com',
21       '*crwdcntrl.net',
22       '*openx.net',
23       '*criteo.com',
24       '*rubiconproject.com',
25       '*stat-rock.com',
26       '*flashtalking.com',
27       '*ad-score.com',
28     ],
29
30     setupNodeEvents(on, config) {
31       const cucumber = require('cypress-cucumber-preprocessor').default;
32       on('file:preprocessor', cucumber());
33     },
34   },
35 });
```

Figura 8 – Cypress config

Fonte: Produzido pelo autor



```
1  {
2    "name": "academia_accenture2025",
3    "version": "2.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC",
11   "cypress-cucumber-preprocessor": {
12     "nonGlobalStepDefinitions": false,
13     "stepDefinitions": "cypress/support/step-definitions/"
14   },
15   "devDependencies": {
16     "cypress": "12.17.4",
17     "cypress-cucumber-preprocessor": "4.3.1"
18   },
19   "dependencies": {
20     "@faker-js/faker": "^9.6.0"
21   }
22 }
23
```

Figura 9 – package.json

Fonte: Produzido pelo autor

2 Apresentação técnica do Projeto de Automação de Testes de Web Tables

Este projeto de automação utiliza Cypress com BDD (Behavior-Driven Development) para testar o comportamento de cadastro, atualização e exclusão de funcionários em uma tabela web. A metodologia BDD é implementada com a combinação de Gherkin (linguagem de escrita dos cenários) e Cypress Cucumber Preprocessor (executor dos testes).

2.1 Tecnologias Utilizadas

- Cypress: Framework de automação de testes end-to-end.
- Cypress-Cucumber-Preprocessor: Plugin que permite a execução de arquivos .feature do Gherkin.
- Gherkin: Linguagem de escrita para definir o comportamento dos testes de forma clara e legível (Given, When, Then).

2.2 Estrutura do Projeto

A organização do projeto segue a convenção do Cypress Cucumber, com uma separação clara entre as definições de comportamento e a implementação do código.

- cypress/e2e/features/: Contém os arquivos .feature escritos em Gherkin, que descrevem os cenários de teste.
- cypress/support/step-definitions/: Contém as definições de passo (.cy.js) que conectam cada linha do Gherkin ao código Cypress.
- cypress/support/pages/: Contém arquivos de comandos personalizados que encapsulam a lógica de teste para cada página (Home.page.js, Elements.page.js), seguindo o padrão Page Object.

2.3 Resumo dos Cenários de Teste

O projeto de automação cobre os seguintes cenários de teste:

2.3.1 Cadastro de Novo Funcionário

Cenário: 01-Novo-cadastro-funcionario.feature

Este cenário verifica se um novo funcionário pode ser cadastrado com sucesso na base de dados do RH.

Dado que o usuário está na página Web Tables.
Quando ele preenche o formulário de cadastro.
Então os dados são exibidos corretamente na tabela.

2.3.2 Atualização de Cadastro de Funcionário

Cenário: 02-Atualizacao-de-cadastro.feature

Este cenário testa a funcionalidade de atualização de dados de um funcionário já existente.

Dado que existe um cadastro válido.
Quando o usuário edita o cadastro e atualiza os dados.
Então os dados atualizados são refletidos na tabela.

2.3.3 Exclusão de Cadastro de Funcionário

Cenário: 03-Deleta-cadastro-de-funcionario.feature

Este cenário garante que um cadastro pode ser excluído da tabela com sucesso.

Dado que existe um cadastro válido.
Quando o usuário deleta o cadastro.
Então o cadastro é removido da tabela.

2.4 Arquivos de Implementação

2.4.1 Comandos e Funções de Página

Os arquivos de página (Home.page.js, Elements.page.js) são a base da arquitetura Page Object. Eles mapeiam seletores e encapsulam a lógica de interação com a interface de usuário em comandos reutilizáveis do Cypress.

- `acessaCardsElements()`: Navega para a página de Elements.
- `acessaWebTables()`: Acessa a seção Web Tables.
- `preencheFormulario()`: Preenche os campos do formulário de cadastro.
- `acessaCadastroFuncionario()`: Encontra e clica no botão de edição de um cadastro específico.
- `deletaCadastro()`: Encontra e clica no botão de exclusão de um cadastro específico.

2.4.2 Definições de Passo

O arquivo 01-Cadastro-positivo.cy.js conecta os cenários do Gherkin aos comandos do Cypress. Ele traduz cada passo (Given, When, Then) para as funções de automação correspondentes, garantindo que o comportamento descrito seja executado.

- Before() hook: Utilizado para interceptar e bloquear requisições de anúncios antes de cada teste, melhorando a performance e a estabilidade.
- Cada passo do Gherkin (Given, When, And, Then) é mapeado para um comando Cypress. Por exemplo, Given('que o usuario esteja na pagina Web Tables', () => ...) chama os comandos de navegação.

2.5 Como Executar os Testes

Para executar os testes, você pode usar a interface gráfica do Cypress.

2.5.1 Via Cypress Test Runner (UI):

- 1 Abra a interface do Cypress com o comando `npx cypress open`.
- 2 Selecione um navegador e posteriormente a feature desejada.

3 Apresentação acadêmica do Projeto de Automação de Testes de Web Tables

3.1 Introdução

O presente documento tem como objetivo apresentar o desenvolvimento de um projeto de automação de testes voltado para aplicações web, utilizando a ferramenta Cypress como principal recurso tecnológico. O projeto foi estruturado seguindo as melhores práticas de automação, destacando-se a adoção da metodologia Behavior Driven Development (BDD), a utilização da linguagem Gherkin para definição dos cenários de teste e a implementação do padrão de projeto Page Object.

A motivação para a realização deste projeto está relacionada à necessidade de obter uma abordagem clara, eficiente e de fácil manutenção para os testes automatizados, garantindo maior confiabilidade nos resultados e promovendo uma comunicação efetiva entre a equipe técnica e os stakeholders.

3.2 Objetivos

O projeto teve como principais objetivos:

- Implementar testes automatizados de interface utilizando o Cypress;
- Adotar a metodologia BDD, permitindo que os cenários de teste sejam compreendidos também por pessoas não técnicas;
- Estruturar o código com base no padrão de projeto Page Object, visando reutilização, clareza e organização;
- Validar fluxos funcionais relevantes da aplicação, incluindo cadastro, atualização, manipulação de dados em tabelas e exclusão de registros;
- Disponibilizar o projeto em repositório público no GitHub, com controle de versão realizado via Git.

3.3 Metodologia

A metodologia aplicada no projeto contemplou as seguintes etapas:

3.3.1 Ferramenta Cypress

O Cypress foi a ferramenta escolhida para automação devido à sua arquitetura moderna, que possibilita testes end-to-end (E2E) com execução rápida e integração simplificada com diferentes ambientes de desenvolvimento.

3.3.2 BDD e Linguagem Gherkin

O projeto foi desenvolvido utilizando a abordagem Behavior Driven Development (BDD), na qual os cenários de teste foram escritos em Gherkin, garantindo legibilidade e entendimento comum entre desenvolvedores, testadores e gestores.

A linguagem Gherkin adota uma estrutura composta por palavras-chave como Given, When, Then, And, But (ou Dado, Quando, Então, E e Mas, respectivamente em português), descrevendo os requisitos de maneira clara e acessível.

3.3.3 Padrão de Projeto Page Object

Com o intuito de melhorar a manutenibilidade e escalabilidade do código, foi implementado o padrão de projeto Page Object, que separa a lógica de interação com a interface gráfica da escrita dos cenários de teste. Isso permitiu a criação de seletores e métodos reutilizáveis, reduzindo redundâncias e facilitando futuras alterações na aplicação.

3.4 Implementação

Durante a implementação, foram realizados os seguintes passos:

- Criação da estrutura de diretórios do projeto em Cypress;
- Configuração dos arquivos `cypress.env.json`, `cypress.config.js`, `package.json`;
- Desenvolvimento de arquivos de step definitions vinculados aos cenários descritos em Gherkin;
- Criação de arquivos de Page Object para mapear elementos da interface e métodos de interação;
- Escrita de testes cobrindo funcionalidades essenciais, como:
 - Cadastro de novos usuários;
 - Busca e validação de informações em tabelas dinâmicas;
 - Exclusão de registros com verificação da remoção.

3.5 Resultados Obtidos

O projeto alcançou resultados positivos, entre os quais destacam-se:

- Execução automatizada de fluxos críticos, garantindo cobertura consistente e sem dependência de testes manuais;
- Melhoria na comunicação entre equipe técnica e não técnica, proporcionada pelo uso do Gherkin;
- Redução do tempo de manutenção graças à aplicação do padrão Page Object;
- Registro versionado no GitHub, assegurando rastreabilidade e controle sobre o histórico do desenvolvimento.

3.6 Conclusão

O desenvolvimento deste projeto demonstrou a relevância da automação de testes como parte fundamental do ciclo de desenvolvimento de software. A utilização do Cypress, aliada à abordagem BDD e ao padrão Page Object, contribuiu para a criação de uma solução robusta, escalável e alinhada às boas práticas da indústria.

Além de melhorar a eficiência e a confiabilidade dos testes, o projeto também reforçou a importância de práticas que facilitam a colaboração entre diferentes áreas, promovendo qualidade e sustentabilidade no processo de desenvolvimento.