Projeto de Automação de Testes de API

Fake Store API – Estudo de Caso

Andressa Evilin Gomes da Silva

Estudante de Quality Assurance (QA)

Tecnologias utilizadas

Cypress • Node.js • GitHub Actions • Jira

Objetivo do projeto

O principal objetivo deste projeto foi desenvolver uma suíte de testes automatizados para a Fake Store API, uma API pública utilizada para fins educacionais e validação de testes, disponibilizando endpoints para autenticação de usuários e gerenciamento de produtos.
Ela permite simular cenários reais de consumo de API, sendo ideal para prática de automação e validação de contratos.

Escopo dos testes automatizados:

1. Products (/products)

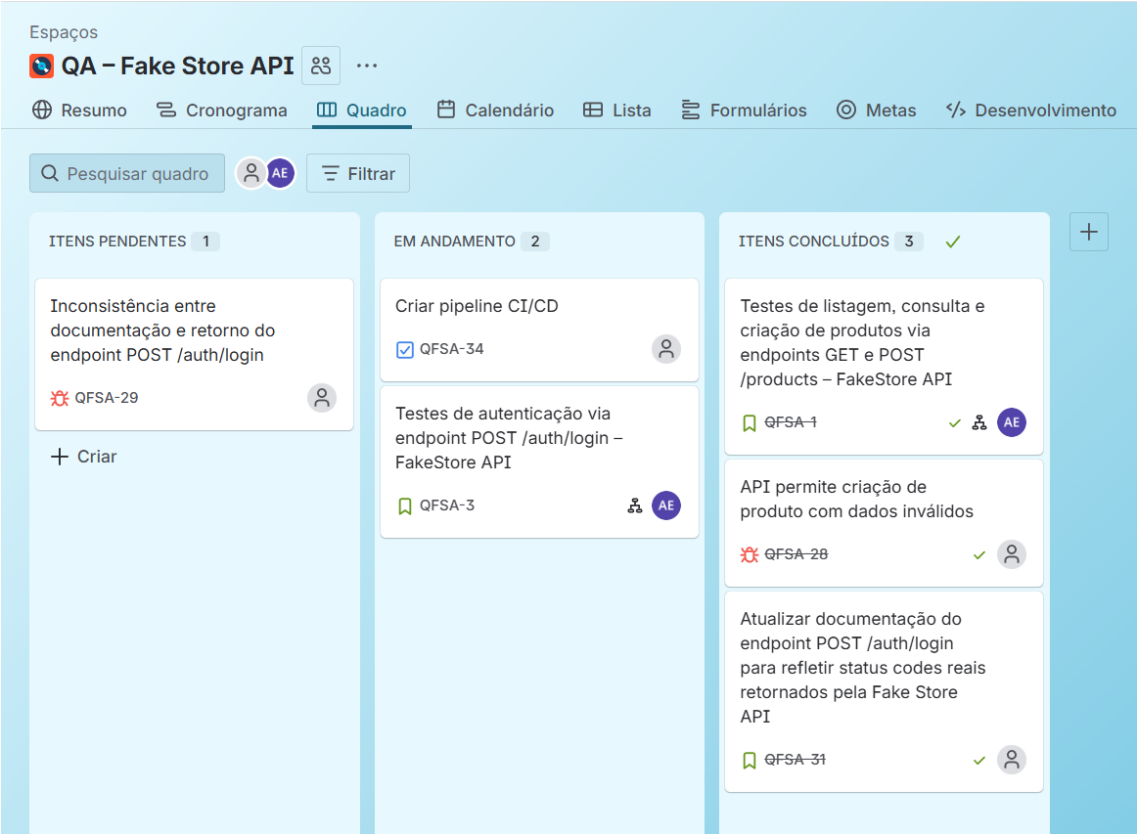
- Listagem: Validação de GET /products e Status Code 200.
- Contrato: Verificação de campos obrigatórios e tipos de dados (ID, Title, Price).
- Criação: Teste de POST /products (Cenários positivos e negativos).

2. Login/Autenticação (/auth/login)

- Teste de login com credenciais válidas: Verificação de retorno de token.
- Status Code: Identificada **inconsistência entre a documentação (que prevê 200) e o comportamento real da API (retorno 201).**
- Validação de erros: Cenários de senha inválida (401) e corpo vazio (400).

Organização e Gestão (Jira)

Para garantir a rastreabilidade e a qualidade das entregas, o projeto foi gerenciado no **Jira** utilizando a metodologia Kanban. A estrutura foi desenhada para cobrir todo o ciclo de vida do teste, desde a análise manual até a automação.



Visualização do fluxo de trabalho no Jira, onde as colunas refletem o status atual das Stories de Login, Produtos e Gestão de Inconsistências.

Estrutura do Backlog

- **Story de Produtos:** Abrange os fluxos de listagem, consulta e criação.
- **Story de Autenticação:** Focada na segurança e geração de tokens.
- **Story de Inconsistências:** Dedicada ao monitoramento de divergências da documentação oficial.

Ciclo de Desenvolvimento (Tasks)

- **Testes Manuais:** Validação exploratória e funcional.
- **Automação (Cypress):** Desenvolvimento dos scripts baseados nos cenários validados.
- **Bugs/Issues:** Registro detalhado de falhas encontradas.

Detalhamento das Tasks e Evidências

Cada funcionalidade passou por uma etapa de **validação manual via Postman** antes da automação, garantindo a integridade dos cenários de teste.

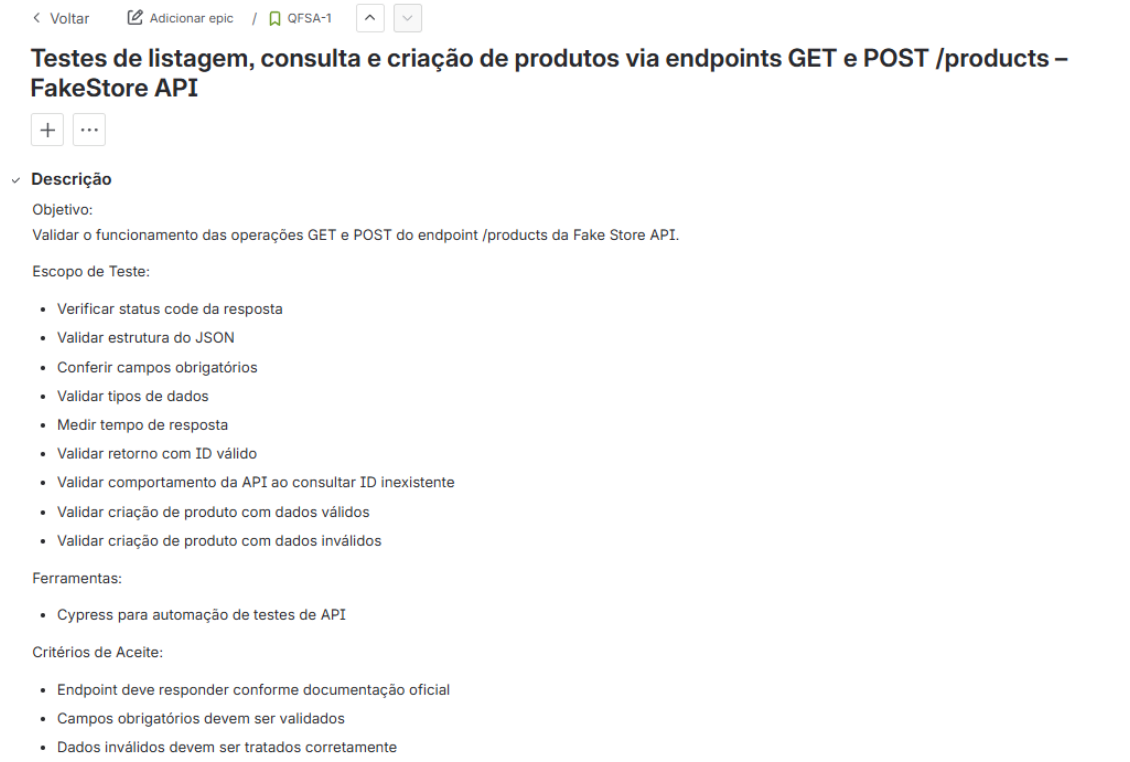
Estrutura do Teste Manual (Jira):

Dentro de cada Task no Jira, os testes foram documentados contendo:

- **Endpoint:** A rota testada (ex: /products).
- **Passos:** O roteiro executado.
- **Resultado Esperado:** O que a documentação previa.
- **Resultado Obtido:** O comportamento real da API.
- **Status:** Passou ou Falhou.

Detalhamento das Stories e Execução de Testes

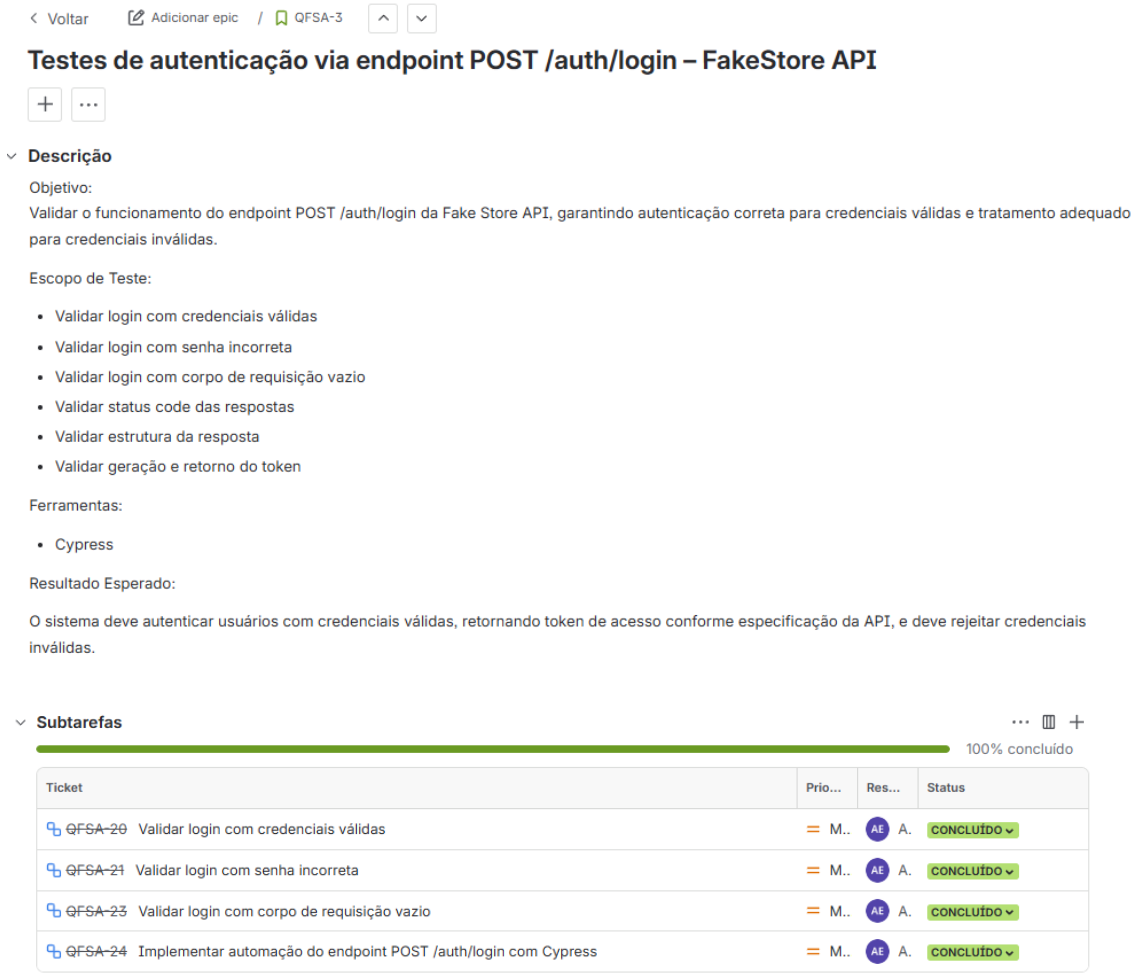
- Story de Produtos



Detalhamento da User Story de Produtos, evidenciando o planejamento de critérios de aceite e a quebra em subtarefas para garantir a cobertura de testes

Exemplo de documentação de teste manual dentro de uma task, servindo como base para a posterior automação no Cypress

- Story de Autenticação e Login



Planejamento dos fluxos de segurança e autenticação.

Voltar QFSA-3 / QFSA-20

Validar login com credenciais válidas

Descrição

Teste Manual - Validar login com credencias válidas

Endpoint:

POST `https://fakestoreapi.com/auth/login`

Pré-condição:

Usuário válido já cadastrado na API.

Dados da Requisição:

```
{
"username": "kevinryan",
"password": "kev02937@"
}
```

Passos:

1. Enviar requisição POST para `/auth/login`
2. Informar credencias validas no corpo da requisição
3. Verificar status code retornado
4. Verificar estrutura da resposta

Resultado esperado:

- Status code = 201
- A resposta deve conter o campo "token"
- O campo "token" não deve estar vazio
- O campo "token" deve ser do tipo string

Resultado obtido:

- Status code = 201
- Campo "token" presente na resposta
- Campo "token" não está vazio
- Tipo do campo "token" = string

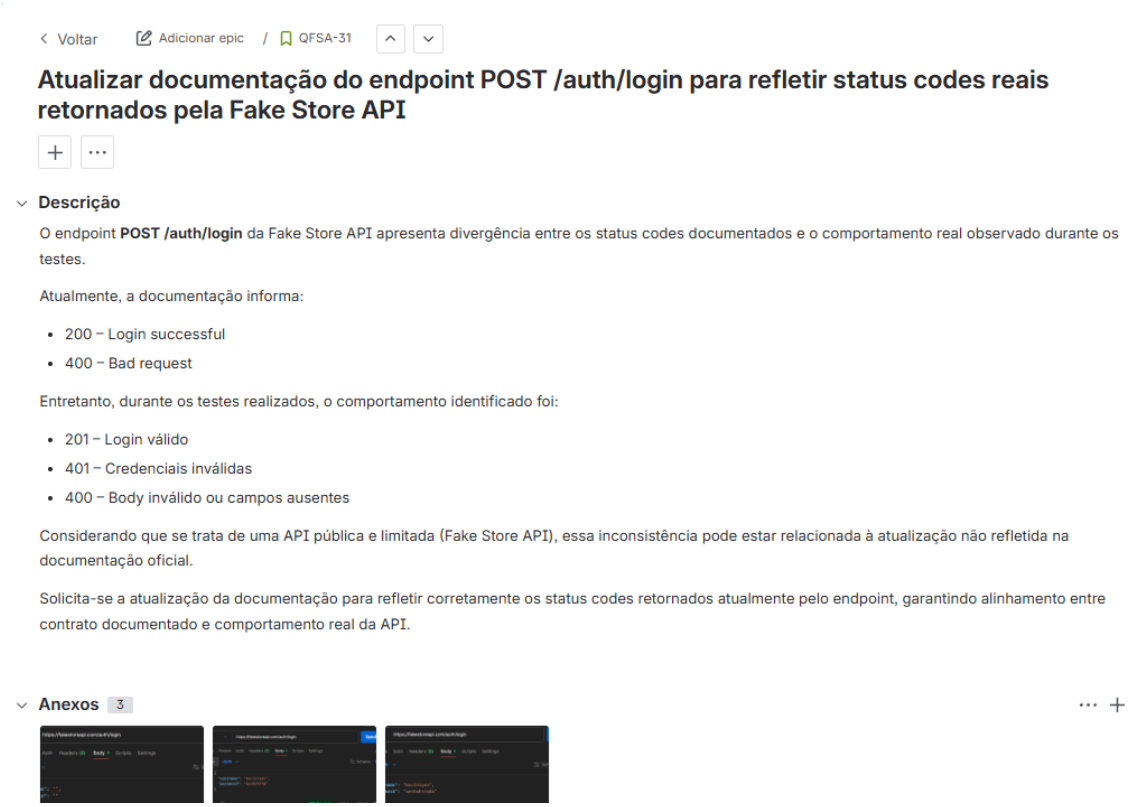
Status

- PASS

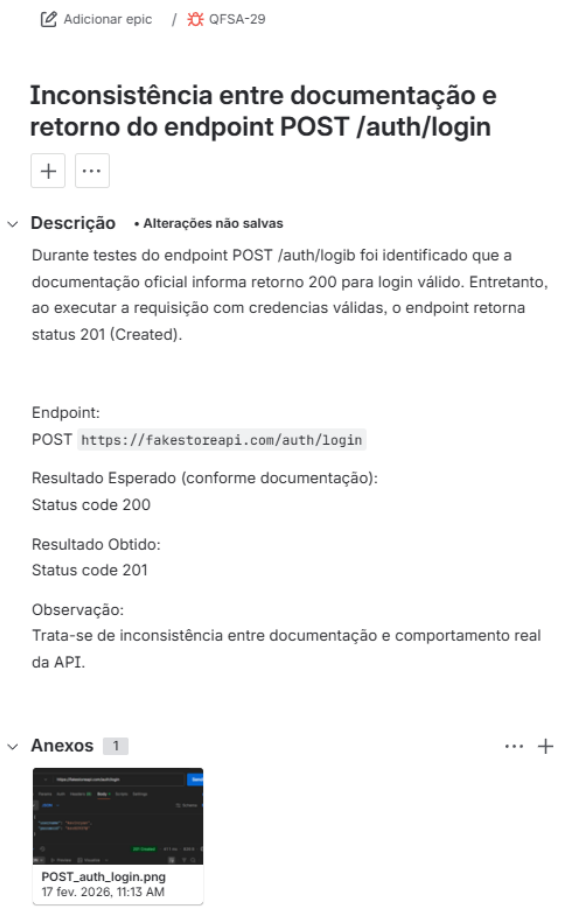
Observação:
Embora a documentação da API informe retorno 200 para login válido, o endpoint retorna 201 (Created). Inconsistência registrada no bug QFSA-29

Detalhamento de teste manual no Jira exemplificando a inconsistência de Status Code encontrada no endpoint de autenticação.

- Story Gestão de Inconsistências



Utilizada para documentar e monitorar divergências técnicas entre a especificação e o comportamento real da API, garantindo a integridade do contrato



Registro de Bug: Divergência de Status Code no endpoint /auth/login identificada via Postman e documentada no Jira.

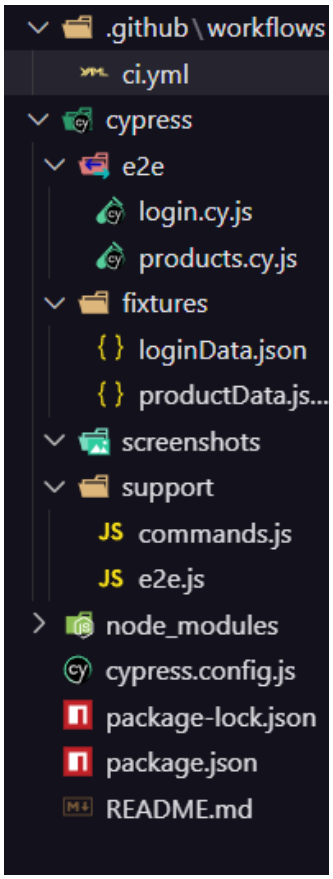
Arquitetura do Projeto (Cypress)

A arquitetura do projeto foi estruturada para ser escalável e organizada, utilizando o **Cypress** como framework principal para a automação de testes de API. O foco foi garantir a separação de responsabilidades entre scripts de teste, massa de dados e configurações de ambiente.

A organização do diretório foi planejada para facilitar a manutenção e a rastreabilidade, conectando-se diretamente às **Stories** planejadas no Jira:

- **fixtures/**: Armazena as massas de dados em formato JSON para garantir que o código do teste seja dinâmico.

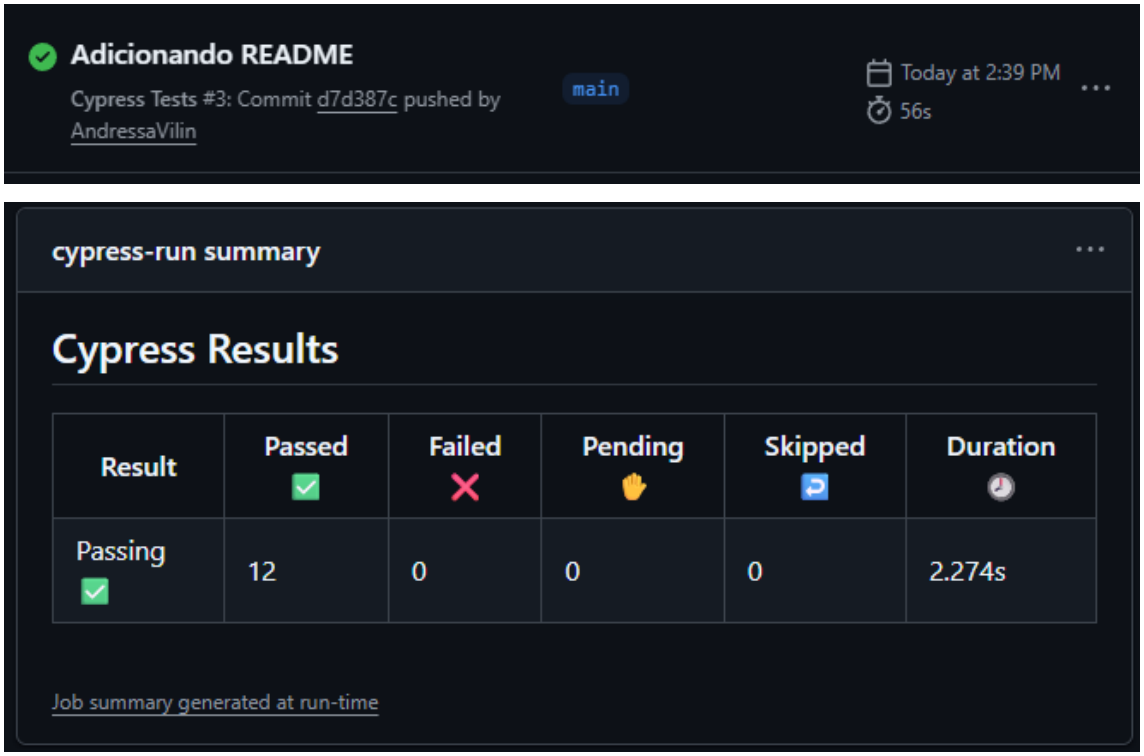
- **e2e/ (ou integration/):** Contém os scripts de teste automatizados. divididos por domínio, refletindo as **Stories** do backlog.
- **support/**: Local onde ficam os comandos customizados (commands.js).
- **cypress.config.js**: Configurações globais do projeto, incluindo a baseUrl da Fake Store API para evitar repetição de código.



Integração Contínua (CI/CD)

Como parte da entrega de qualidade, o projeto inclui automação de fluxo:

- **GitHub Actions**: Configurado para executar a suíte de testes automaticamente a cada *push* ou *pull request*, garantindo que novas alterações não quebrem funcionalidades existentes (testes de regressão).



Repositório do Projeto

O código fonte completo, incluindo a configuração da pipeline de **CI/CD (GitHub Actions)** e a documentação técnica detalhada (**README**), está disponível no meu GitHub.

Acesse o repositório completo no GitHub

Conclusão

Este projeto demonstrou o ciclo completo de **Quality Assurance**, desde a análise de requisitos e gestão de backlog no **Jira** até à execução de testes manuais e a implementação de uma suíte de regressão automatizada com **Cypress**.

A detecção de divergência no Status Code de autenticação reforça o papel do QA na validação de contratos, garantindo que a comunicação entre sistemas seja precisa e que a documentação técnica esteja em total conformidade com o comportamento real da API.