

# **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.

The screenshot shows a Jira Kanban board for the project "QA - Fake Store API". The board is organized into three columns: "ITENS PENDENTES" (1 item), "EM ANDAMENTO" (2 items), and "ITENS CONCLUÍDOS" (3 items). Each column contains stories with their descriptions, IDs, and status indicators (green checkmark for completed, red X for pending or failed).

ITENS PENDENTES	EM ANDAMENTO	ITENS CONCLUÍDOS
Inconsistência entre documentação e retorno do endpoint POST /auth/login QFSA-29	Criar pipeline CI/CD (QFSA-34) Testes de autenticação via endpoint POST /auth/login – FakeStore API (QFSA-3)	Testes de listagem, consulta e criação de produtos via endpoints GET e POST /products – FakeStore API (QFSA-1) API permite criação de produto com dados inválidos (QFSA-28) Atualizar documentação do endpoint POST /auth/login para refletir status codes reais retornados pela Fake Store API (QFSA-31)

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

< Voltar    Adicionar epic / QFSA-1   

**Testes de listagem, consulta e criação de produtos via endpoints GET e POST /products – FakeStore API**

✓ **Descrição**

Objetivo:  
Validar o funcionamento das operações GET e POST do endpoint /products da Fake Store API.

Escopo de Teste:

- Verificar status code da resposta
- Validar estrutura do JSON
- Conferir campos obrigatórios
- Validar tipos de dados
- Medir tempo de resposta
- Validar retorno com ID válido
- Validar comportamento da API ao consultar ID inexistente
- Validar criação de produto com dados válidos
- Validar criação de produto com dados inválidos

Ferramentas:

- Cypress para automação de testes de API

Critérios de Aceite:

- Endpoint deve responder conforme documentação oficial
- Campos obrigatórios devem ser validados
- Dados inválidos devem ser tratados corretamente

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

✓ **Subtarefas**

... +  
100% concluído

Ticket	Prioridade	Responsável	Status
QFSA-5 Validação de status code da resposta	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-6 Validação da estrutura do JSON	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-7 Validação dos campos obrigatórios do produto	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-8 Validação dos tipos de dados dos campos	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-10 Validação do tempo de resposta do endpoint /products	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-11 Implementar automação com Cypress	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-18 Validação do retorno com ID válido	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-19 Validação do comportamento da API ao consultar ID inexistente	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-25 Validação de criação de produto com dados válidos	= Medium	AE Andressa Evilim	CONCLUÍDO
QFSA-26 Validação de criação de produto com dados inválidos	= Medium	AE Andressa Evilim	CONCLUÍDO

## Validar retorno com ID válido



### ▼ Descrição

Teste manual - Validar retorno com ID válido

Endpoint: GET `https://fakestoreapi.com/products/1`

Passos:

1. Enviar requisição GET para `/products/1`
2. Observar status code
3. Verificar se o corpo da resposta contém um produto

Resultado esperado:

- Status code 200
- Resposta em JSON
- Retorna objeto de produto

Resultado obtido:

- Status 200 retornado
- Produto exibido corretamente

Status:

PASS

---

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

- Story de Autenticação e Login

< Voltar    Adicionar epic / QFSA-3   

### Testes de autenticação via endpoint POST /auth/login – FakeStore API

▼ Descrição

Objetivo:

Validar o funcionamento do endpoint POST /auth/login da Fake Store API, garantindo autenticação correta para credenciais válidas e tratamento adequado para credenciais inválidas.

Escopo de Teste:

- Validar login com credenciais válidas
- Validar login com senha incorreta
- Validar login com corpo de requisição vazio
- Validar status code das respostas
- Validar estrutura da resposta
- Validar geração e retorno do token

Ferramentas:

- Cypress

Resultado Esperado:

O sistema deve autenticar usuários com credenciais válidas, retornando token de acesso conforme especificação da API, e deve rejeitar credenciais inválidas.

▼ Subtarefas

... 100% concluído

Ticket	Prio...	Res...	Status
QFSA-20 Validação de login com credenciais válidas	M..	AE	A. CONCLUIDO
QFSA-21 Validação de login com senha incorreta	M..	AE	A. CONCLUIDO
QFSA-23 Validação de login com corpo de requisição vazio	M..	AE	A. CONCLUIDO
QFSA-24 Implementar automação do endpoint POST /auth/login com Cypress	M..	AE	A. CONCLUIDO

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://fakesstoreapi.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

< Voltar    Adicionar epic / QFSA-31    ^ ^

**Atualizar documentação do endpoint POST /auth/login para refletir status codes reais retornados pela Fake Store API**

+ ...

▼ Descrição

O endpoint **POST /auth/login** da Fake Store API apresenta divergência entre os status codes documentados e o comportamento real observado durante os testes.

Atualmente, a documentação informa:

- 200 – Login successful
- 400 – Bad request

Entretanto, durante os testes realizados, o comportamento identificado foi:

- 201 – Login válido
- 401 – Credenciais inválidas
- 400 – Body inválido ou campos ausentes

Considerando que se trata de uma API pública e limitada (Fake Store API), essa inconsistência pode estar relacionada à atualização não refletida na documentação oficial.

Solicita-se a atualização da documentação para refletir corretamente os status codes retornados atualmente pelo endpoint, garantindo alinhamento entre contrato documentado e comportamento real da API.

▼ Anexos | 3

... +



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

Adicionar epic / QFSA-29

## Inconsistência entre documentação e retorno do endpoint POST /auth/login

[+](#) [...](#)

### ▼ Descrição • Alterações não salvas

Durante testes do endpoint POST /auth/login foi identificado que a documentação oficial informa retorno 200 para login válido. Entretanto, ao executar a requisição com credencias válidas, o endpoint retorna status 201 (Created).

Endpoint:

POST <https://fakestoreapi.com/auth/login>

Resultado Esperado (conforme documentação):

Status code 200

Resultado Obtido:

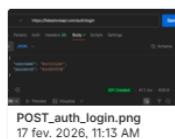
Status code 201

Observação:

Trata-se de inconsistência entre documentação e comportamento real da API.

### ▼ Anexos [1](#)

[...](#) [+](#)



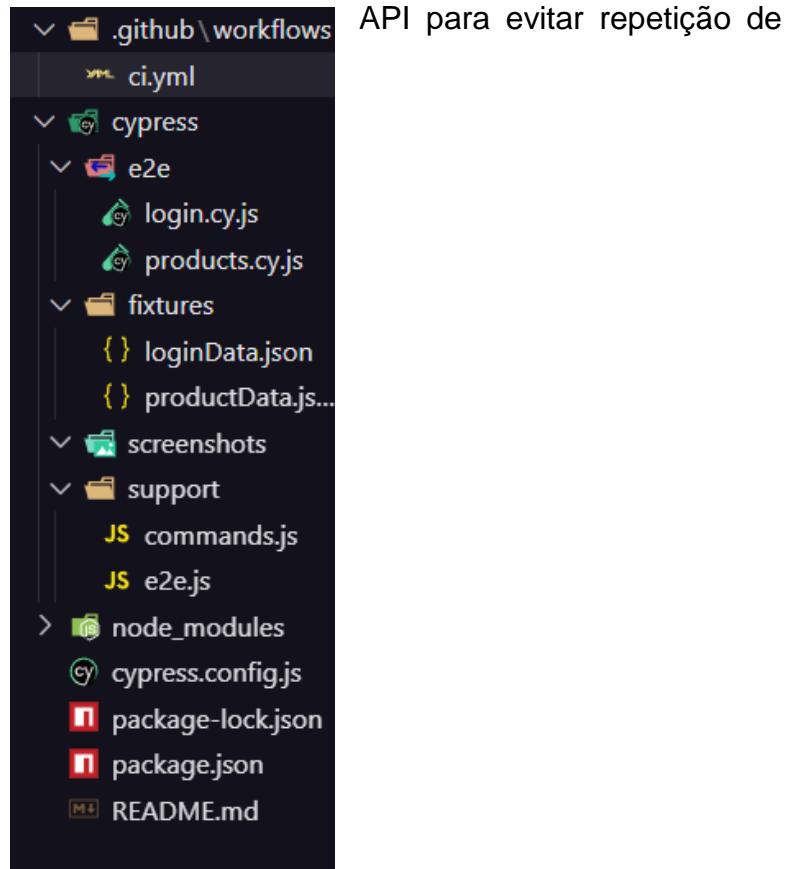
Registro de Bug: Divergência de Status Code no endpoint /auth/login identificada via Postman e documentada no Jira.

## 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



## 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).

The screenshot shows a GitHub Actions job summary for a commit pushed by Andressa Vilin. The job is named "Adicionando README" and is associated with the "main" branch. It was run today at 2:39 PM and completed in 56 seconds. The summary includes a "cypress-run summary" section titled "Cypress Results". The results table shows the following data:

Result	Passed	Failed	Pending	Skipped	Duration
Passing	12	0	0	0	2.274s

Job summary generated at run-time

## 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.