Instituto Metropolitano de Ensino

Faculdade Metropolitana de Manaus – Fametro

Engenharia de Software

Teste de Software

Manaus/AM

Autor: João Vitor de Sousa Pereira

A seguir, apresento uma documentação baseada no código de testes, que utiliza a biblioteca `requests` e `pytest` para testar a API do JSONPlaceholder (https://jsonplaceholder.typicode.com). Esses testes cobrem operações básicas de API REST: GET para obter posts, POST para criar, PUT para atualizar e DELETE para deletar. A documentação em Gherkin foi adaptada para descrever cenários comportamentais (BDD) alinhados a esses testes, utilizando a estrutura padrão com Features, Scenarios, Given, When e Then. Os cenários estão em português para consistência com o contexto fornecido.

Para executar esses cenários, integre ferramentas como Cucumber (para Python) ou Behave (para Python) ao projeto. Os testes assumem que a API está funcionando corretamente e retornando respostas esperadas.

Feature: Obter Todos os Posts

Como um usuário da API,

Eu quero obter uma lista de todos os posts disponíveis,

Para visualizar o conteúdo geral.

Scenario: Acesso bem-sucedido a todos os posts

Given que a API está acessível

When eu faço uma requisição GET para “/posts”

Then o status da resposta deve ser 200

And a resposta deve conter uma lista de posts

And cada post deve ter um campo “id”

Feature: Obter um Post Específico

Como um usuário da API,

Eu quero obter detalhes de um post específico,

Para visualizar informações individuais.

Scenario: Acesso bem-sucedido a um post específico

Given que a API está acessível

When eu faço uma requisição GET para “/posts/1”

Then o status da resposta deve ser 200

And a resposta deve conter o post com id 1

And o post deve ter um campo “title”

Feature: Criar um Novo Post

Como um usuário da API,

Eu quero criar um novo post,

Para adicionar conteúdo à plataforma.

Scenario: Criação bem-sucedida de um novo post

Given que a API está acessível

And eu preparo um payload com título “Novo Post”, corpo “Corpo do post” e userId 1

When eu faço uma requisição POST para “/posts” com o payload

Then o status da resposta deve ser 201

And a resposta deve conter o novo post com título “Novo Post”

And o post deve ter um campo “id” atribuído

Feature: Atualizar um Post

Como um usuário da API,

Eu quero atualizar um post existente,

Para modificar seu conteúdo.

Scenario: Atualização bem-sucedida de um post

Given que a API está acessível

And eu preparo um payload com título “Título Atualizado”

When eu faço uma requisição PUT para “/posts/1” com o payload

Then o status da resposta deve ser 200

And a resposta deve conter o post atualizado com título “Título Atualizado”

Feature: Deletar um Post

Como um usuário da API,

Eu quero deletar um post existente,

Para remover conteúdo da plataforma.

Scenario: Exclusão bem-sucedida de um post

Given que a API está acessível

When eu faço uma requisição DELETE para “/posts/1”

Then o status da resposta deve ser 200

Esses cenários em Gherkin mapeiam diretamente os testes unitários fornecidos no código (ex.: `test_get_all_posts`, `test_get_single_post`, etc.). Eles podem ser implementados em um arquivo `.feature` para integração com frameworks BDD. Para mais detalhes sobre a implementação dos testes, consulte o código Python fornecido.