2. TESTES AUTOMATIZADOS DE API

Objetivo: Avaliar a capacidade de automatizar testes de API e interpretar os resultados.

Tarefas Práticas:

- Tarefa 1 : Usando ferramentas como Rest Assured, Cypress ou Playwright, crie um teste para validar endpoints de uma API de exemplo. O teste deve incluir verificações de resposta (status codes, headers e corpo).
    - Avaliação: Uso correto de ferramentas, clareza dos testes e cobertura de cenários importantes (testes positivos, negativos, etc.).
- Tarefa 2: Automatizar testes para múltiplos endpoints da API, validando diferentes métodos HTTP (GET, POST, PUT, DELETE), e realizar validações de status codes, headers e corpo de resposta para cada um dos métodos.
    - Relatório: Gerar e apresentar um relatório detalhado com os resultados dos testes.

INFORMAÇÕES GERAIS

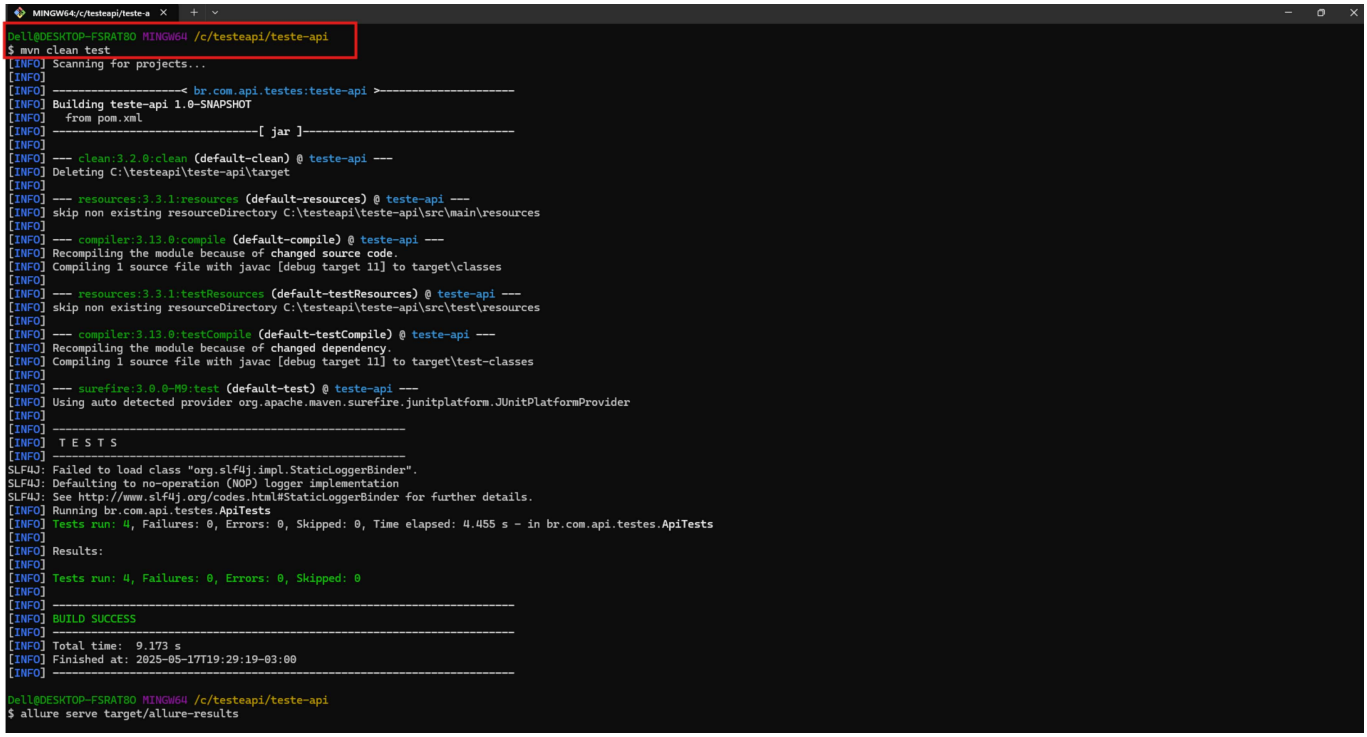
Projeto: Teste API com TestGetPosts, TestPostCreate, TestPutUpdate e TestDeletePost, 400 (BadRequest) 401 (Unauthorized),403 (Forbiden) , 404 (NotFound) e 500 (Server Error)
Ferramenta: Rest Assured 5.4.0, JUnit 5.10.2, Allure 2.13.9, Maven 3.9 e Java 11
Base URL: https://jsonplaceholder.typicode.com
Base URL simulando erros: https://httpstat.us
Sistema Operacional: Windows 10
Data de Execução: 17/05/2025
Responsável: *[Bruno Nascimento]*

EVIDÊNCIAS

Estrutura dos testes:
TestGetPosts: valida GET /posts
TestPostCreate: valida criação via POST
TestPutUpdate: valida atualização via PUT
TestDeletePost: valida exclusão via DELETE

Teste	Método HTTP	Endpoint	Descrição	Resultado esperado
TestGetPosts	GET	/posts	Valida retorno status 200 e lista não vazia	Status 200, JSON com > 0 itens
TestPostCreate	POST	/posts	Cria novo post com título, corpo e userId	Status 201, retorno com dados criados
TestPutUpdate	PUT	/posts/1	Atualiza título e corpo do post 1	Status 200, título atualizado
TestDeletePost	DELETE	/posts/1	Deleta post 1	Status 200 ou 204

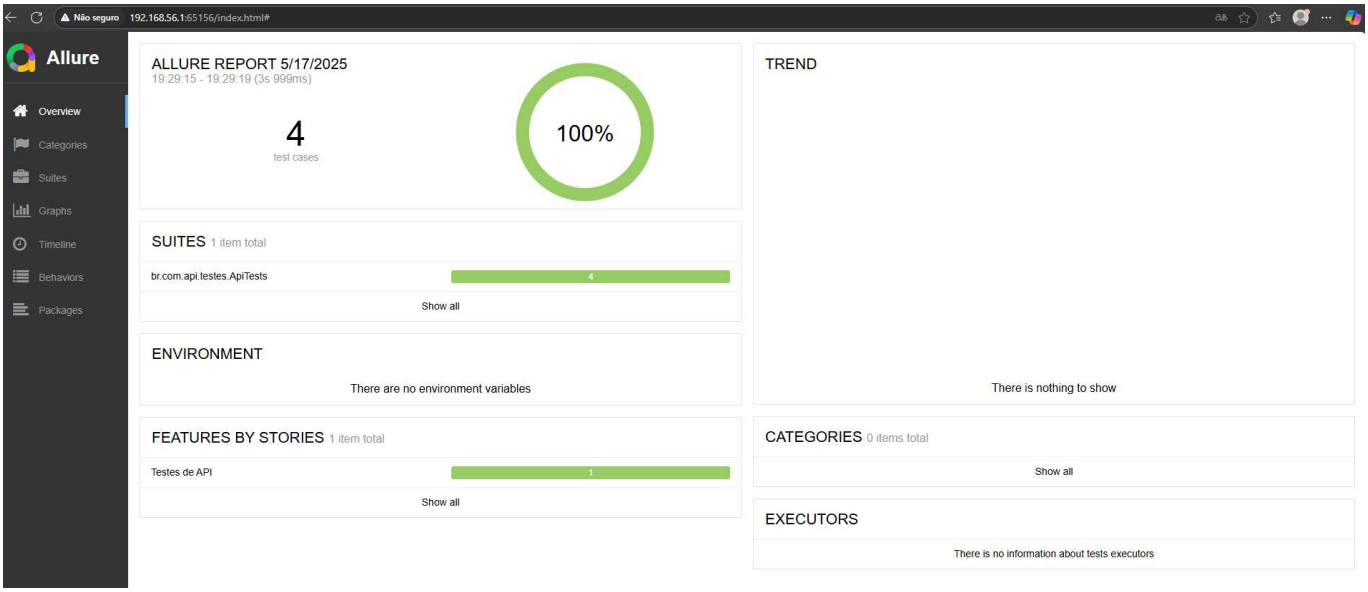
Executando o comando mvn clean test no bash:
Com esse comando geramos os tests results:



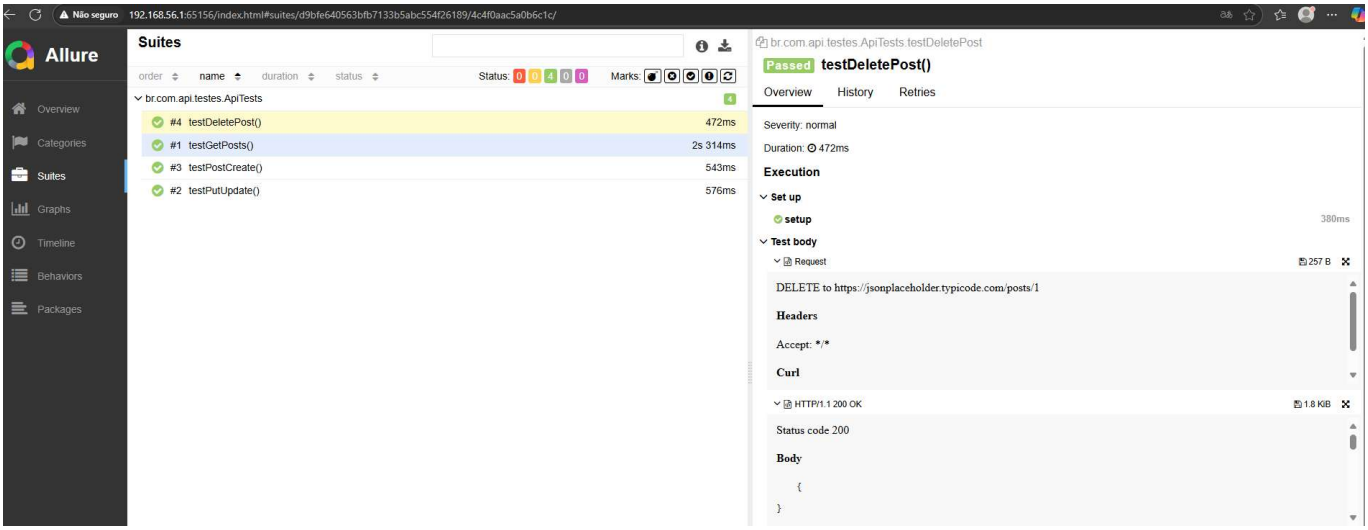
Executando o comando do report allure após gerar o teste corretamente:
allure serve target/allure-results

```
Dell@DESKTOP-FSRAT8O MINGW64 /c/testeapi/teste-api
$ allure serve target/allure-results
Generating report to temp directory...
Report successfully generated to C:\Users\Dell\AppData\Local\Temp\11739275443976481236\allure-report
Starting web server...
2025-05-17 19:33:57.924:INFO::main: Logging initialized @3008ms to org.eclipse.jetty.util.log.StdErrLog
Server started at <http://192.168.56.1:65156/>. Press <Ctrl+C> to exit
```

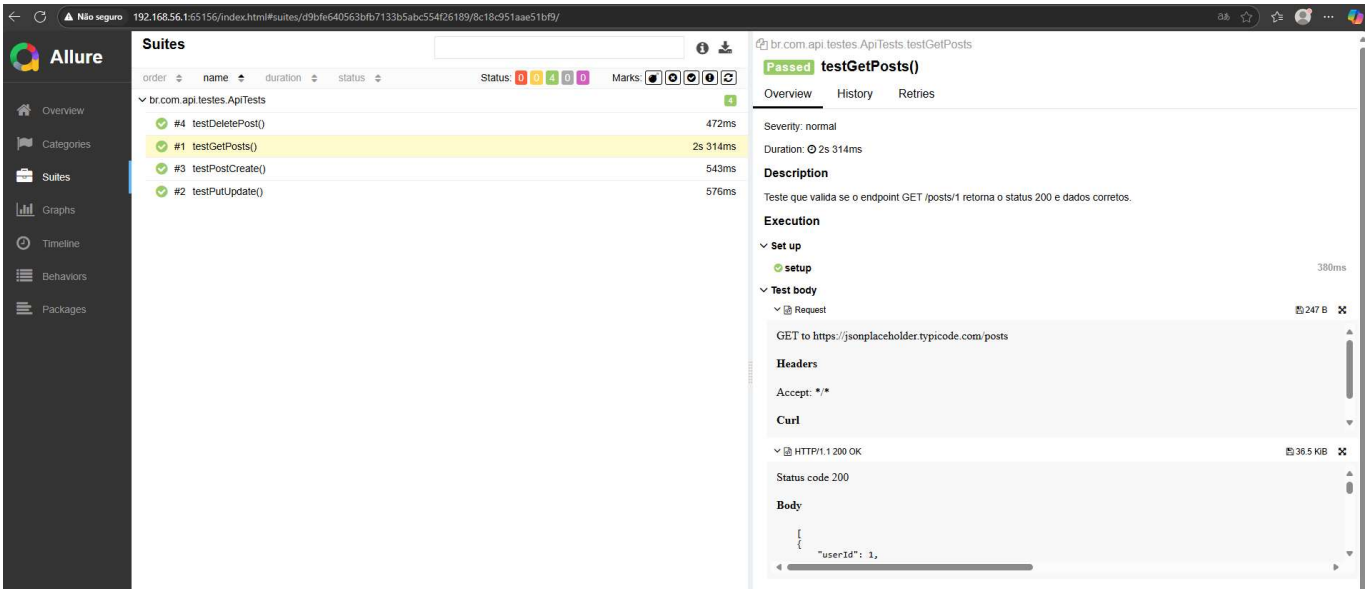
Após executar o comando acima, automaticamente abre o Allure Report:



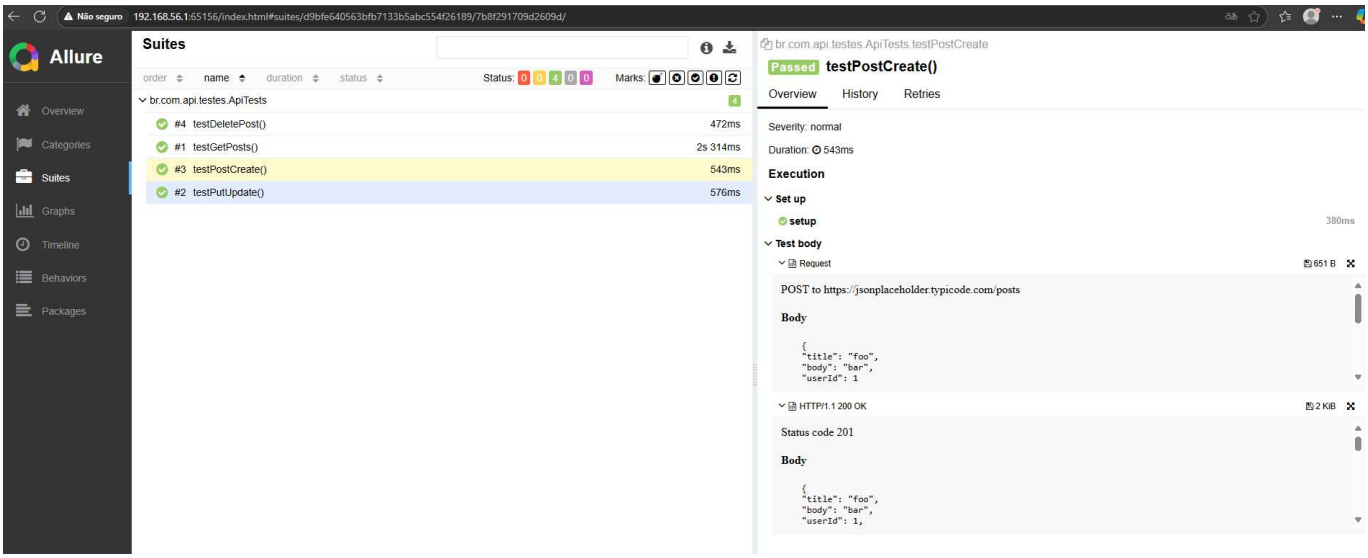
testDeletPost()



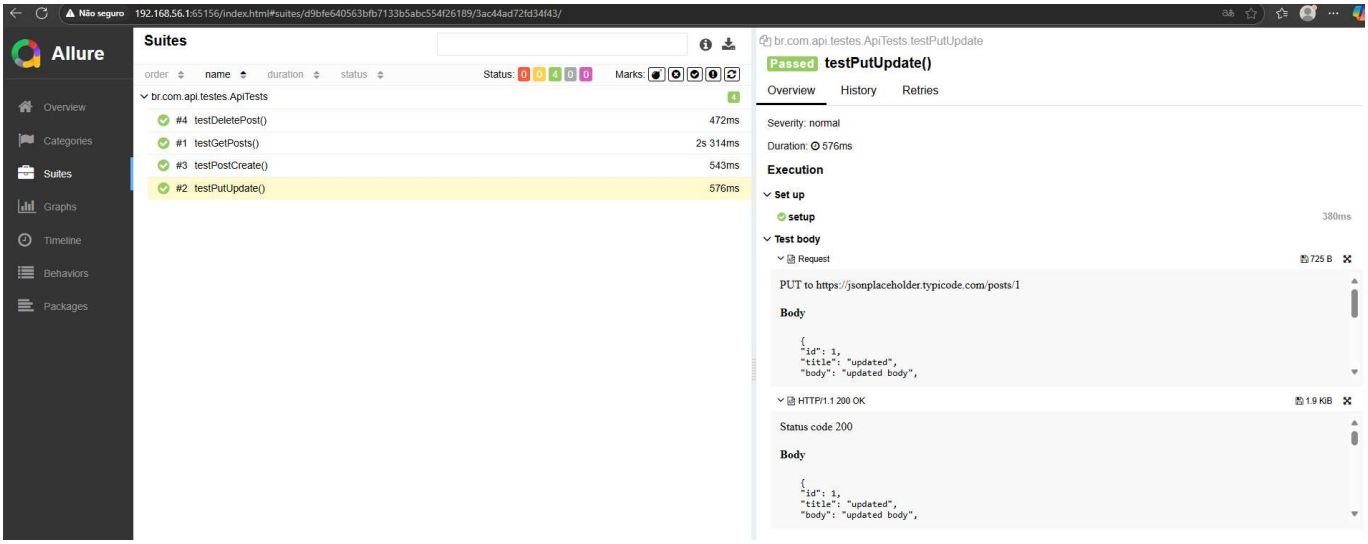
TestGetPost()



TestPostCreat()

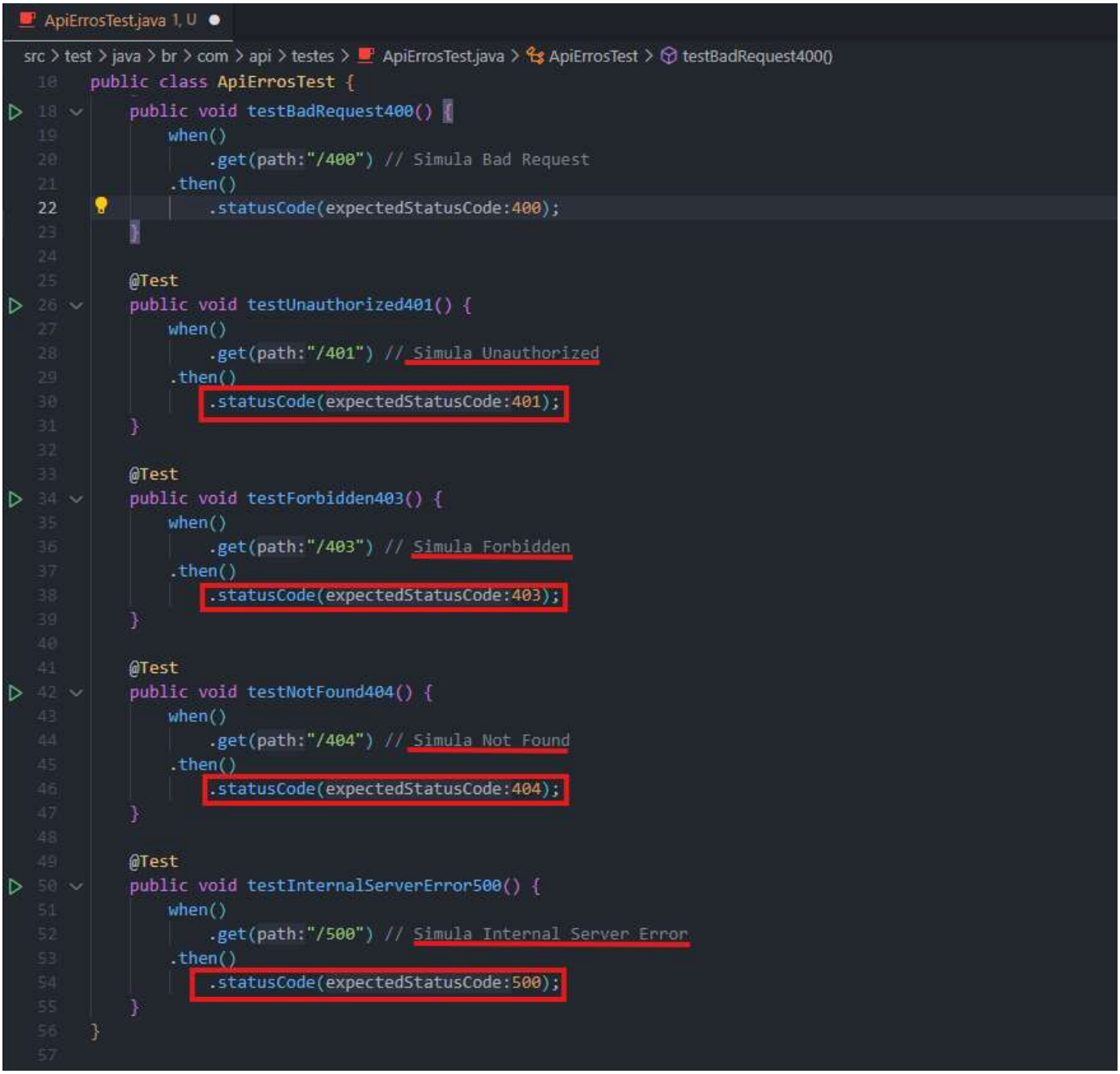


TestPostUpdate()



SIMULANDO API COM ERROS

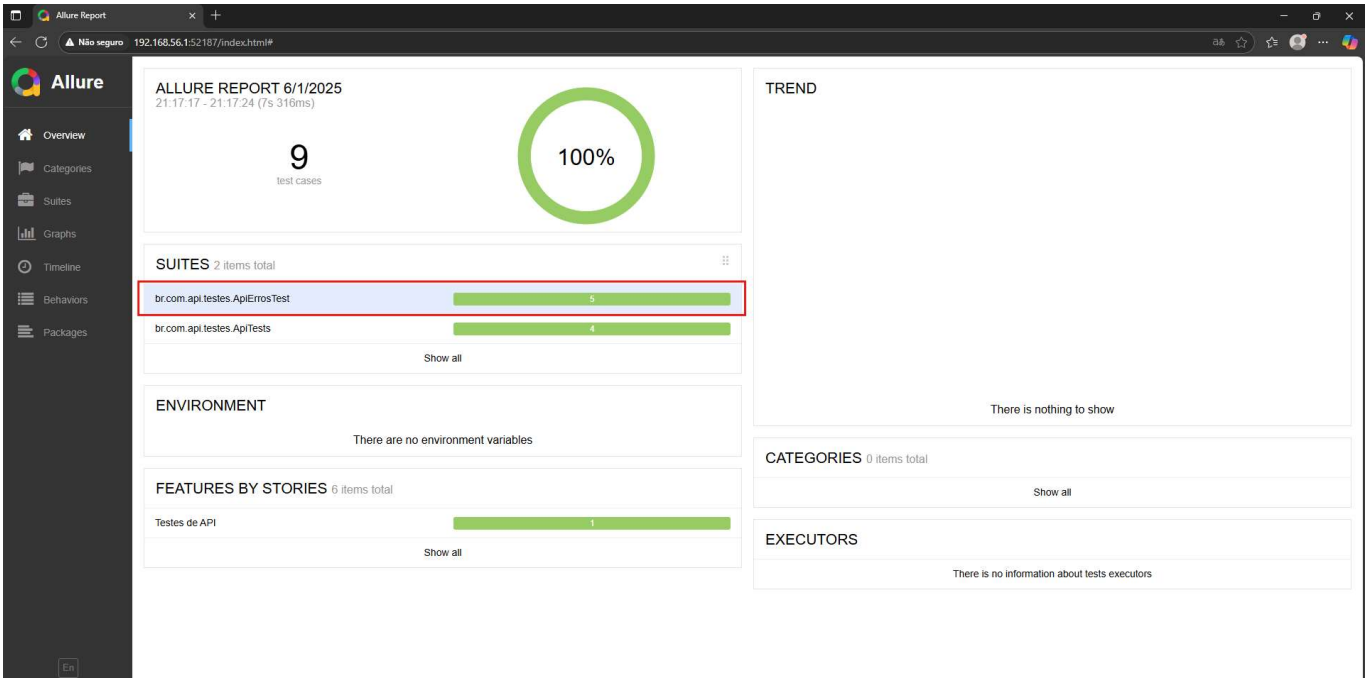
400 (BadRequest)
401 (Unauthorized)
403 (Forbiden)
404 (NotFound)
500 (Server Error)

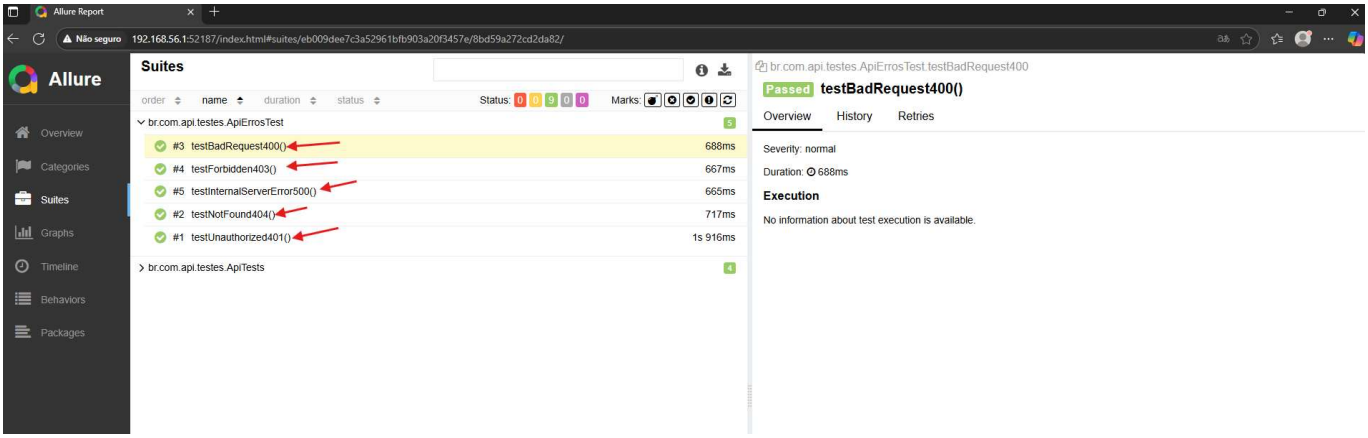


Execução:

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[INFO] Running br.com.api.testes.ApiErrosTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.166 s - in br.com.api.testes.ApiErrosTest
[INFO] Running br.com.api.testes.ApiTests
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.56 s - in br.com.api.testes.ApiTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.925 s
[INFO] Finished at: 2025-06-01T21:17:24-03:00
[INFO] ------------------------------------------------------------------------
PS C:\TesteAPI> allure serve target/allure-results
Generating report to temp directory...
```

Results:


Allure Report
Não seguro 192.168.56.1:52187/index.html#
Allure
Overview
Categories
Suites
Graphs
Timeline
Behaviors
Packages
ALLURE REPORT 6/1/2025
21:17:17 - 21:17:24 (7s 316ms)
9
test cases
100%
TREND
There is nothing to show
SUITES 2 items total
br.com.api.testes.ApiErrosTest 5
br.com.api.testes.ApiTests 4
Show all
ENVIRONMENT
There are no environment variables
FEATURES BY STORIES 6 items total
Testes de API 1
Show all
CATEGORIES 0 items total
Show all
EXECUTORS
There is no information about tests executors


Allure Report
Não seguro 192.168.56.1:52187/index.html#suites/eb009dee7c3a52961bfb903a20f3457e/8bd59a272cd2da82/
Allure
Overview
Categories
Suites
Graphs
Timeline
Behaviors
Packages
Suites
order name duration status
Status: 0 0 9 0 0
Marks:
br.com.api.testes.ApiErrosTest 5
#3 testBadRequest400() 688ms
#4 testForbidden403() 667ms
#5 testInternalServerError500() 665ms
#2 testNotFound404() 717ms
#1 testUnauthorized401() 1s 916ms
br.com.api.testes.ApiTests 4
br.com.api.testes.ApiErrosTest.testBadRequest400
Passed testBadRequest400()
Overview History Retries
Severity: normal
Duration: 688ms
Execution
No information about test execution is available.

CONCLUSÃO

A automação de testes foi conduzida com sucesso, validando os principais métodos HTTP (**GET**, **POST**, **PUT**, **DELETE**) da API pública JSONPlaceholder, além de simular e verificar respostas com **códigos de erro** (**400**, **401**, **403**, **404**, **500**) utilizando a API httpstat.us.

Todos os testes foram desenvolvidos em **Java**, utilizando a biblioteca **Rest Assured** e o framework **JUnit 5**.

A integração com o **Allure Report** permitiu a geração de relatórios detalhados, com logs das requisições, dados das validações, status dos testes e evidências visuais claras, facilitando a rastreabilidade e análise de falhas (caso ocorram).

Todos os testes executaram conforme o esperado, sem apresentar erros ou falhas, demonstrando que:

- Os endpoints estão operacionais e respondem de forma consistente;
- A estrutura de automação está funcional, modular, reutilizável e de fácil manutenção;
- O projeto **já está integrado ao GitHub Actions**, possibilitando a execução automatizada dos testes a cada push ou pull request, fortalecendo a confiabilidade do processo de entrega contínua.

Este cenário comprova que a estratégia de testes automatizados adotada é eficaz tanto para **execução local** quanto para **pipelines de CI/CD**, agregando valor ao processo de validação de APIs REST e contribuindo para uma cultura de qualidade contínua.