

# Cenários de Teste

**Software:** Fake Store API

**QA responsável:** Elisama dos Santos Gregorio

---

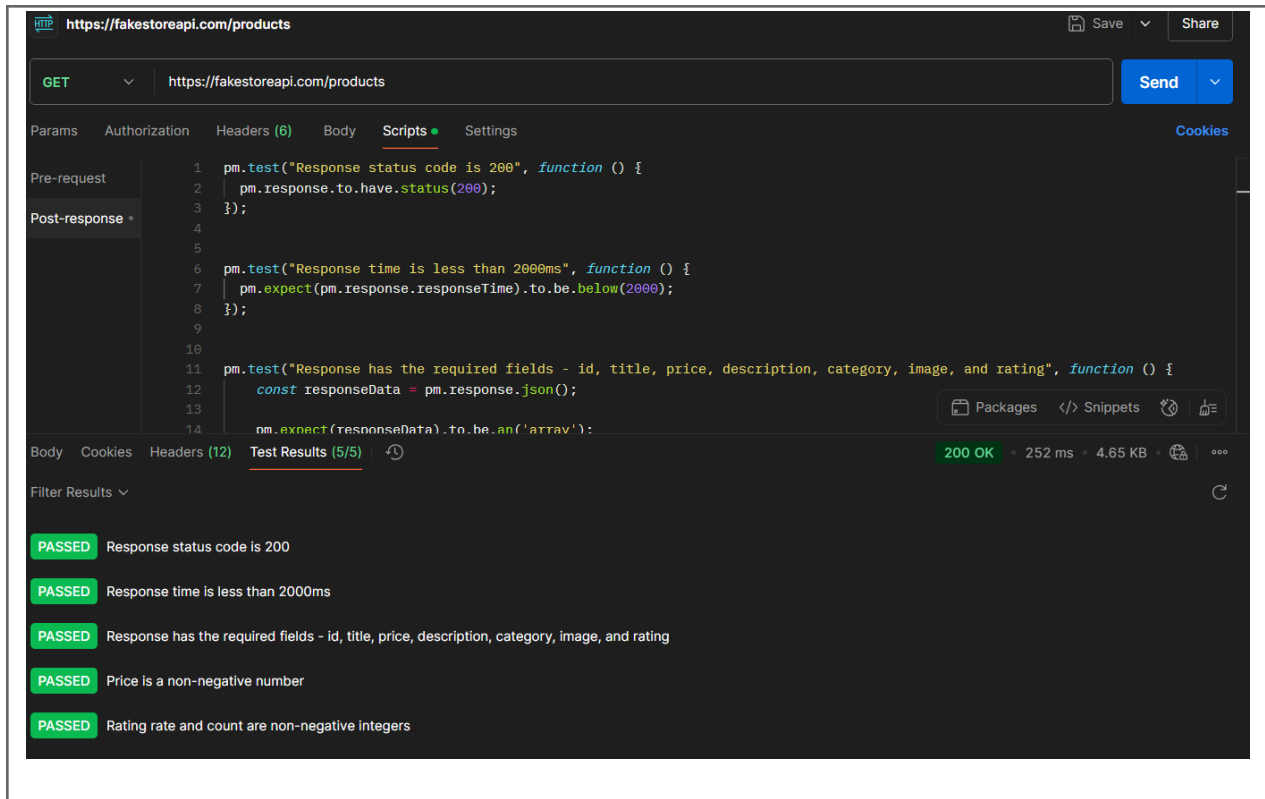
## Produtos (/products)

### Cenário de Teste 01

Cenário de Teste: Listar todos os produtos		
ID	Descrição	
TC001	Verificar se o endpoint <b>GET /products</b> da FakeStoreAPI retorna corretamente uma lista de produtos com os campos esperados e o status apropriado. Esse teste garante que a API está funcional para consumo de dados de produtos.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Funcional</li><li>• API – Teste Positivo</li><li>• Black-box</li></ul>
Pré-condições		
<ul style="list-style-type: none"><li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li><li>• Ferramenta de testes de API deve estar configurada (ex: Postman).</li><li>• Conexão com a internet ativa.</li><li>• O ambiente não exige autenticação para este endpoint.</li></ul>		

Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma requisição com o método <b>GET</b>.</li> <li>3. Informar a URL: <a href="https://fakestoreapi.com/products">https://fakestoreapi.com/products</a>.</li> <li>4. Verificar se nenhum cabeçalho especial é necessário.</li> <li>5. Enviar a requisição.</li> <li>6. Analisar o código de status retornado.</li> <li>7. Validar se o corpo da resposta é um array.</li> <li>8. Verificar se o array contém pelo menos um produto.</li> <li>9. Para cada item do array, validar se existem os seguintes campos: <ol style="list-style-type: none"> <li>a. <code>id</code> (inteiro)</li> <li><code>title</code> (string)</li> <li>b. <code>price</code> (float)</li> <li>c. <code>category</code> (string)</li> </ol> </li> <li>10. Verificar se o tempo de resposta é menor que 2 segundos.</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status HTTP: 200 OK</li> <li>• Corpo: Array com pelo menos um produto</li> <li>• Cada item com os campos válidos (<code>id</code>, <code>title</code>, <code>price</code>, <code>category</code>)</li> <li>• Tempo de resposta: &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200 OK</li> <li>• Corpo: Array com ao menos 1 produto</li> <li>• Todos os produtos com campos válidos</li> <li>• Tempo de resposta: &lt;2 segundos</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A API deve retornar ao status <b>HTTP 200</b>.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração deve ocorrer no estado da API ou do sistema.</li> </ul>

<ul style="list-style-type: none"><li>• O corpo da resposta deve ser um <b>array de objetos JSON</b>.</li><li>• Cada objeto deve conter obrigatoriamente os campos:<ol style="list-style-type: none"><li>1. <b>id</b> (int)</li><li>2. <b>title</b> (string)</li><li>3. <b>price</b> (float)</li><li>4. <b>category</b> (string)</li></ol></li><li>• O tempo de resposta deve ser <b>inferior a 2000 ms</b>.</li></ul>	<ul style="list-style-type: none"><li>• Nenhum dado deve ser criado, alterado ou excluído.</li></ul>
Dados de teste	Status
<ul style="list-style-type: none"><li>• Requisição sem corpo.</li><li>• Método: <b>GET</b></li><li>• Endpoint: <b><a href="https://fakestoreapi.com/products">https://fakestoreapi.com/products</a></b></li></ul>	<b>Sucesso</b>
Evidência(s)	



## Cenário de Teste 02

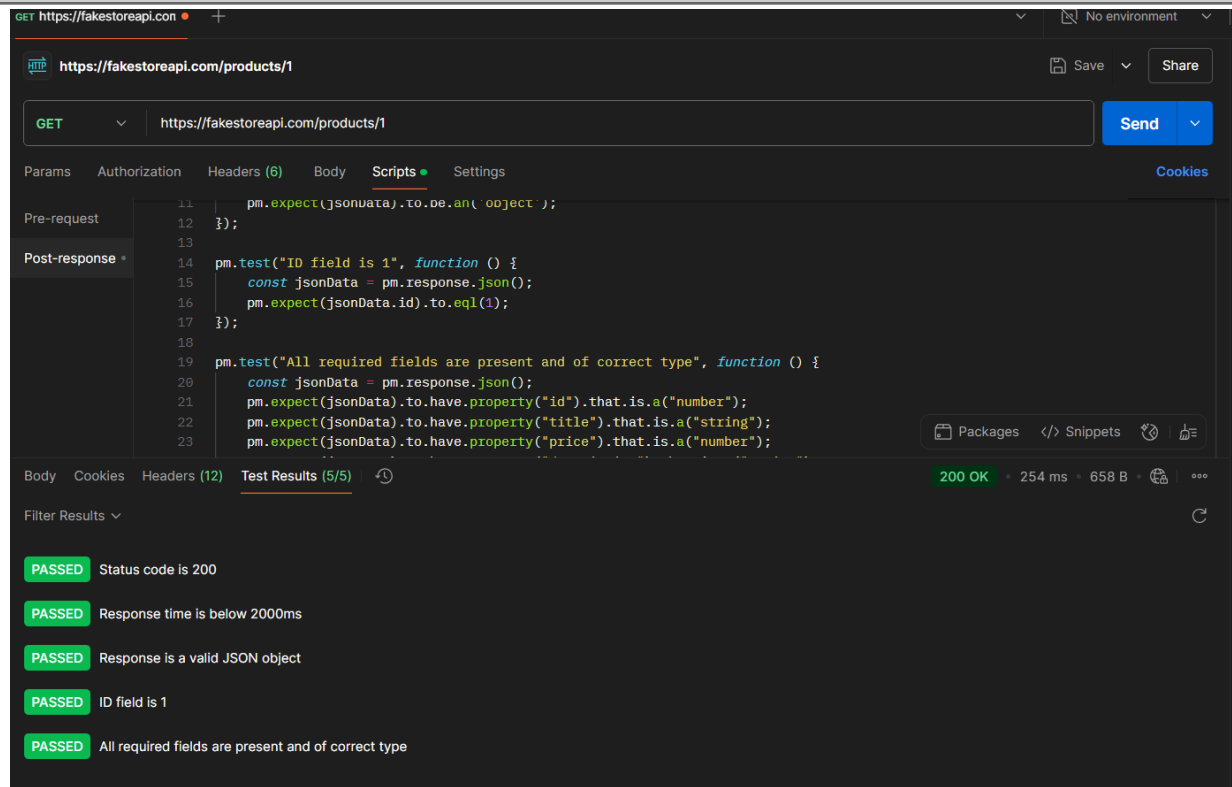
Cenário de Teste: Buscar produto por ID existente		
ID	Descrição	
TC002	Verificar se a API retorna corretamente os dados de um produto existente ao realizar uma requisição para o endpoint <code>GET /products/{id}</code> utilizando um ID válido (1).	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Funcional</li><li>• API – Teste Positivo</li><li>• Black-box</li></ul>

<b>Pré-condições</b>	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• O produto com ID 1 deve existir no sistema.</li> <li>• Ferramenta de testes de API instalada e funcionando ( Postman).</li> <li>• Conexão com a internet ativa.</li> </ul>	
<b>Passos</b>	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma nova requisição com o método <b>GET</b>.</li> <li>3. Informar a URL: <a href="https://fakestoreapi.com/products/1">https://fakestoreapi.com/products/1</a>.</li> <li>4. Enviar a requisição.</li> <li>5. Verificar o status da resposta.</li> <li>6. Validar que o corpo da resposta é um objeto JSON.</li> <li>7. Verificar se o campo <b>id</b> é igual a 1.</li> <li>8. Verificar se os campos obrigatórios estão presentes no corpo da resposta: <ul style="list-style-type: none"> <li>• <b>id</b> (int)</li> <li>• <b>title</b> (string)</li> <li>• <b>price</b> (float)</li> <li>• <b>category</b> (string)</li> <li>• <b>description</b> (string)</li> <li>• <b>image</b> (string - URL) Validar que os campos têm valores coerentes com o tipo de dado esperado.]</li> </ul> </li> <li>9. Validar o tempo de resposta (inferior a 2 segundos).</li> </ol>	
<b>Resultado Esperado</b>	<b>Resultado Obtido</b>

<ul style="list-style-type: none"> <li>• Status HTTP: <b>200 OK</b></li> <li>• Campo <code>id</code> = 1</li> <li>• Todos os campos com tipos corretos</li> <li>• Tempo de resposta &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>• Status: <b>200 OK</b></li> <li>• Tempo de resposta: <b>254 ms</b></li> <li>• Campos validados com sucesso:</li> <li>• <code>id</code> = 1</li> <li>• Tipos corretos para <code>title</code>, <code>price</code>, <code>description</code>, <code>category</code>, <code>image</code></li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A API deve retornar ao status <b>HTTP 200</b>.</li> <li>• O corpo da resposta deve ser um <b>objeto JSON</b>.</li> <li>• O campo <code>id</code> deve ser igual a 1.</li> <li>• Os campos obrigatórios devem estar presentes e preenchidos: <code>id</code>, <code>title</code>, <code>price</code>, <code>category</code>, <code>description</code>, <code>image</code></li> <li>• Tempo de resposta inferior a <b>2000 ms</b>.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração deve ocorrer no estado da API ou no banco de dados.</li> <li>• Nenhum dado novo é criado, modificado ou excluído.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <code>GET</code></li> </ul>	<b>Sucesso</b>

- Endpoint:  
`https://fakestoreapi.com/products/1`
- Headers:        `Content-Type:`  
`application/json`
- Produto esperado: ID 1

### Evidência(s)



## Cenário de Teste 03

### Cenário de Teste: Buscar produto por ID existente

ID	Descrição
----	-----------

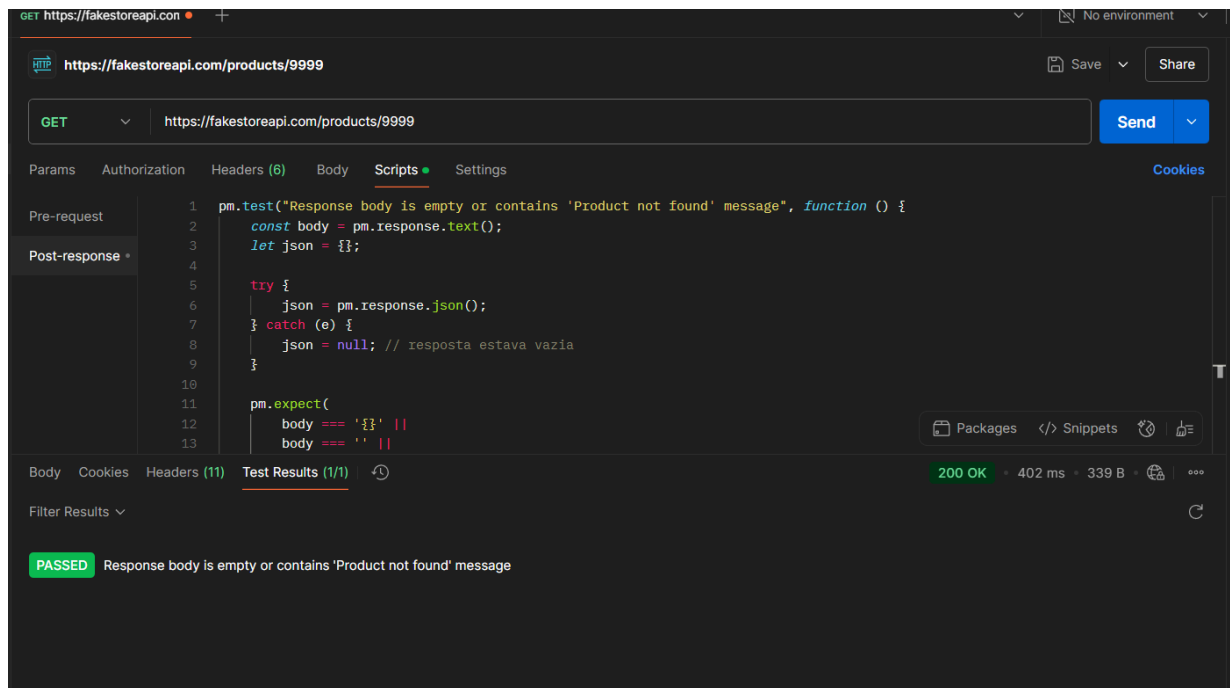
TC003	Verificar o comportamento da API ao realizar uma requisição para o endpoint <b>GET /products/{id}</b> utilizando um ID que <b>não existe</b> no sistema (ex: ID 9999). O objetivo é validar o tratamento de erro e a resposta retornada pela API.
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• API – Teste Negativo</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• O ID 9999 não deve existir na base de dados da API.</li> <li>• Ferramenta de testes de API deve estar disponível e funcional.</li> <li>• Conexão com a internet ativa.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API (ex: Postman).</li> <li>2. Criar uma requisição com o método <b>GET</b>.</li> <li>3. Informar a URL: <a href="https://fakestoreapi.com/products/9999">https://fakestoreapi.com/products/9999</a>.</li> <li>4. Enviar a requisição.</li> <li>5. Verificar o código de status retornado.</li> <li>6. Analisar o corpo da resposta.</li> <li>7. Verificar se o corpo está vazio <b>ou</b> se contém uma mensagem indicando que o produto não foi encontrado.</li> <li>8. Validar o tempo de resposta (inferior a 2 segundos).</li> </ol>	
Resultado Esperado	Resultado Obtido



<ul style="list-style-type: none"> <li>• Status HTTP: <b>404 Not Found</b> ou <b>200 OK</b> com corpo vazio</li> <li>• Mensagem: <b>"Product not found"</b> (se aplicável)</li> <li>• Tempo de resposta &lt; 2 segundos</li> <li>• Nenhuma falha no sistema ou erro inesperado</li> </ul>	<ul style="list-style-type: none"> <li>• Status HTTP: 200 OK</li> <li>• Corpo: vazio</li> <li>• Tempo de resposta: 395ms</li> <li>• A resposta não apresentou erro inesperado, mas retornou 200 em vez de 404, o que é um comportamento que pode ser melhorado na API.</li> </ul>
CrITÉRIOS DE ACEITAÇÃO	Pós-condições
<ul style="list-style-type: none"> <li>• A API deve retornar ao status <b>HTTP 200</b>.</li> <li>• O corpo da resposta deve ser um <b>objeto JSON</b>.</li> <li>• O campo <b>id</b> deve ser igual a <b>1</b>.</li> <li>• Os campos obrigatórios devem estar presentes e preenchidos: <b>id, title, price, category, description, image</b></li> <li>• Tempo de resposta inferior a <b>2000 ms</b>.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração no estado da API ou dos dados da base.</li> <li>• Nenhum novo dado foi criado, atualizado ou deletado.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <b>GET</b></li> </ul>	<b>Sucesso</b>

- Endpoint:  
<https://fakestoreapi.com/products/9999>
- Headers: **Content-Type:**  
**application/json** (opcional)
- ID de produto inexistente: **9999**

### Evidência(s)



## Cenário de Teste 04

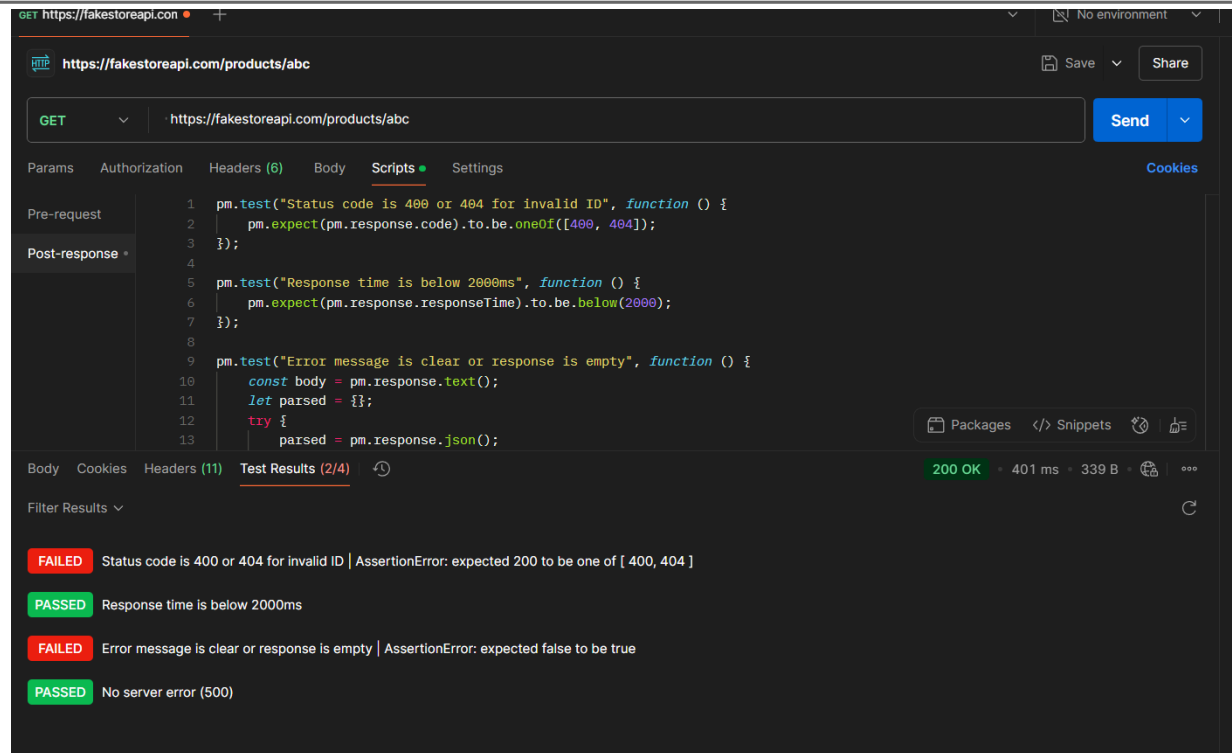
**Cenário de Teste: Buscar produto com ID inválido (string ou especial)**

ID	Descrição	
TC004	Verificar o comportamento da API ao enviar valores inválidos como ID (ex: strings ou caracteres especiais, como "abc" ou "@\$"). O objetivo é garantir que a API realiza a validação adequada da entrada e retorna um erro tratável e compreensível.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"> <li>• Funcional</li> <li>• API – Teste Negativo</li> <li>• Validação de entrada / Robustez</li> <li>• Black-box</li> </ul>
Pré-condições		
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• A ferramenta de testes de API deve estar funcional (Postman).</li> <li>• Conexão com a internet ativa.</li> </ul>		
Passos		
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma requisição com o método <b>GET</b>.</li> <li>3. Inserir a URL com o ID inválido: <ul style="list-style-type: none"> <li>• Ex: <a href="https://fakestoreapi.com/products/abc">https://fakestoreapi.com/products/abc</a></li> </ul> </li> <li>4. Enviar a requisição.</li> <li>5. Verificar o status HTTP da resposta.</li> <li>6. Verificar se a API retorna uma mensagem de erro clara ou estrutura controlada.</li> <li>7. Validar se <b>nenhum erro 500 ou exceção técnica</b> ocorre.</li> <li>8. Verificar o tempo de resposta (inferior a 2 segundos).</li> </ol>		

Resultado Esperado	Resultado Obtido
<p>Para "abc"</p> <ul style="list-style-type: none"> <li>Status HTTP: 400 Bad Request ou 404 Not Found</li> <li>Mensagem de erro clara (ex: "Invalid ID")</li> <li>Corpo pode ser {} ou JSON com message</li> <li>Tempo de resposta &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>Status: Retornou 200</li> <li>Mensagem: Mensagem não clara ou ausente</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>A API deve retornar ao status HTTP 200.</li> <li>O corpo da resposta deve ser um objeto JSON.</li> <li>O campo id deve ser igual a 1.</li> <li>Os campos obrigatórios devem estar presentes e preenchidos: id, title, price, category, description, image</li> <li>Tempo de resposta inferior a 2000 ms.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhuma alteração no estado da API ou dos dados da base.</li> <li>Nenhum novo dado foi criado, atualizado ou deletado.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Valor Testado: "abc"</li> <li>Método: GET</li> </ul>	Bug Encontrado

- Endpoint:  
*https://fakestoreapi.com/products/abc*

### Evidência(s)



## Cenário de Teste 05

### Cenário de Teste: Criar produto com método não permitido

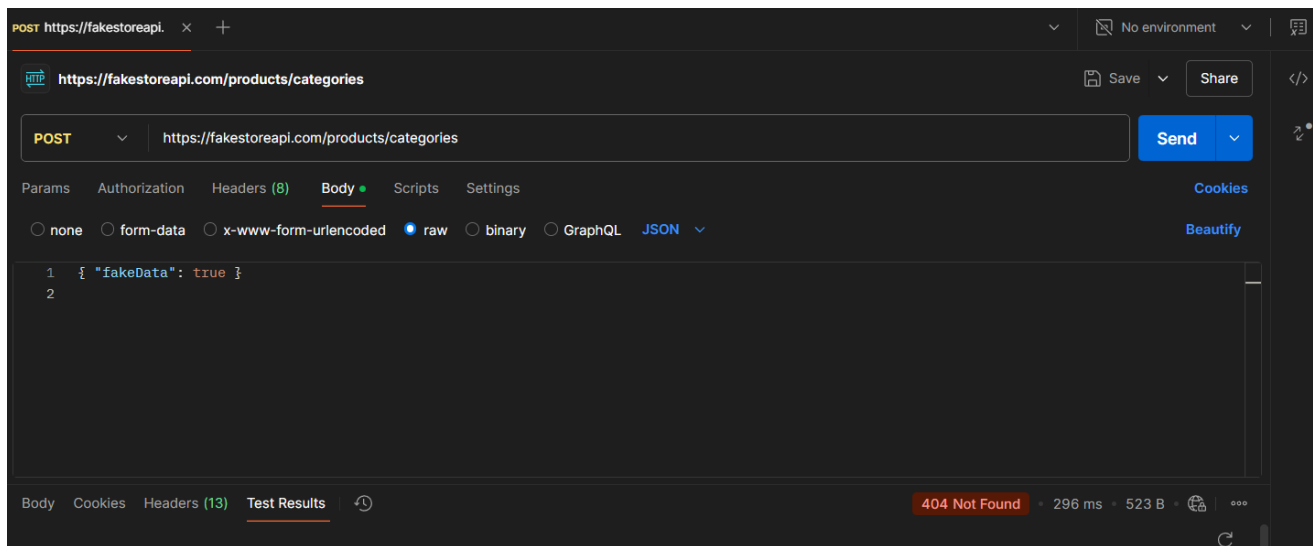
ID	Descrição
<b>TC005</b>	Verificar se a API lida corretamente com o uso indevido do método HTTP POST no endpoint <code>/products/categories</code> , que é destinado apenas à leitura de categorias. O objetivo é garantir que a API retorne um status

	apropriado indicando que o método não é permitido, sem criar ou alterar dados.
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• API – Teste Negativo</li> <li>• Validação de método HTTP</li> <li>• Segurança / Robustez</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• Nenhum dado deve ser criado ou modificado.</li> <li>• Ferramenta de testes de API instalada e funcional (ex: Postman).</li> <li>• Conexão com a internet ativa.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma nova requisição com o método <b>POST</b>.</li> <li>3. Informar a URL: <a href="https://fakestoreapi.com/products/categories">https://fakestoreapi.com/products/categories</a></li> <li>4. (Opcional) Inserir um corpo JSON qualquer (ex: { "test": "invalid" })</li> <li>5. Enviar a requisição.</li> <li>6. Verificar o código de status retornado.</li> <li>7. Verificar se a resposta contém uma <b>mensagem indicando que o método não é permitido</b>.</li> <li>8. Validar se <b>nenhum dado foi criado</b>.</li> <li>9. Verificar o <b>tempo de resposta</b> (deve ser inferior a 2 segundos).</li> <li>10. Garantir que <b>não houve erro 500</b> ou outro comportamento inesperado.</li> </ol>	
Resultado Esperado	Resultado Obtido

<ul style="list-style-type: none"><li>• Status HTTP: <b>404 Not Found</b></li><li>• Mensagem: "Method Not Allowed" ou equivalente.</li><li>• Corpo: { "error": "Method Not Allowed" } ou vazio.</li><li>• Tempo de resposta: &lt; <b>2000 ms</b></li><li>• Nenhuma criação ou alteração de dados.</li></ul>	<ul style="list-style-type: none"><li>• Status: 404</li><li>• Corpo: { "error": "Method Not Allowed" }</li><li>• Tempo de resposta: 296 ms</li></ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"><li>• A API <b>não deve aceitar</b> requisição POST para esse endpoint.</li><li>• Deve retornar <b>404 ou outro erro controlado</b>.</li><li>• Nenhum dado novo deve ser criado.</li><li>• Tempo de resposta inferior a 2 segundos.</li><li>• Nenhum erro 500 ou exceção técnica inesperada.</li></ul>	<ul style="list-style-type: none"><li>• Nenhum dado novo deve ser criado na API.</li><li>• Nenhuma modificação no banco de dados ou estado do sistema.</li></ul>
Dados de teste	Status
<ul style="list-style-type: none"><li>• Método: <i>POST</i></li></ul>	<b>Sucesso</b>

- **Endpoint:**  
`https://fakestoreapi.com/products/categories`
- **Headers:** `Content-Type: application/json`

### Evidência(s)



## Cenário de Teste 06

### Cenário de Teste: Verificar tipo de dados dos campos

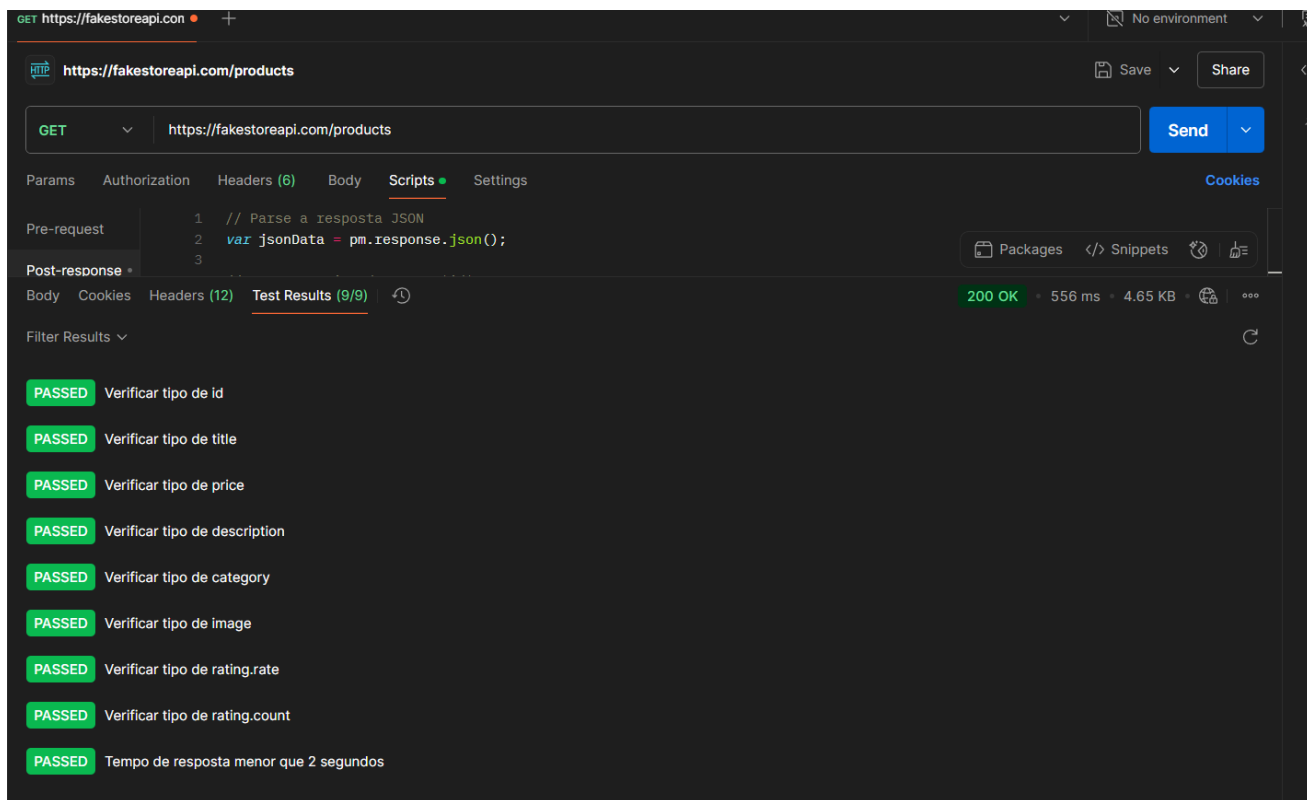
ID	Descrição
TC006	Validar se os campos retornados na resposta do endpoint <code>GET /products</code> estão com os <b>tipos de dados corretos</b> , como <code>title</code> sendo uma <b>string</b> , <code>price</code> sendo <b>numérico</b> , <code>id</code> como <b>inteiro</b> , etc. Esse teste garante a integridade dos dados da API.



Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• API – Validação de dados</li> <li>• Teste de contrato / Tipagem</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• Ferramenta de testes de API funcional (Postman).</li> <li>• Conexão com a internet.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar requisição <b>GET</b> para o endpoint: <a href="https://fakestoreapi.com/products">https://fakestoreapi.com/products</a>.</li> <li>3. Enviar a requisição.</li> <li>4. Receber o array de produtos.</li> <li>5. Selecionar um ou mais objetos do array.</li> <li>6. Validar os tipos dos seguintes campos: <ul style="list-style-type: none"> <li>• <b>id</b>: inteiro</li> <li>• <b>title</b>: string</li> <li>• <b>price</b>: número (float ou inteiro)</li> <li>• <b>description</b>: string</li> <li>• <b>category</b>: string</li> <li>• <b>image</b>: string (URL)</li> </ul> </li> <li>7. Repetir a verificação em diferentes itens da lista para garantir consistência.</li> <li>8. Validar tempo de resposta inferior a 2 segundos.</li> </ol>	

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>A resposta deve ser um array de objetos JSON.</li> <li>Cada objeto deve conter: <pre> {    "id": int,    "title": string,    "price": float,    "description": string,    "category": string,    "image": string,    "rating": {      "rate": float,      "count": int    }  }</pre> </li> <li>Nenhuma exceção ou tipo inesperado.</li> <li>Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li><code>id</code>: inteiro</li> <li><code>title</code>: string</li> <li><code>price</code>: float</li> <li><code>description</code>: string</li> <li><code>category</code>: string</li> <li><code>image</code>: string (URL)</li> <li><code>rating.rate</code>: float</li> <li><code>rating.count</code>: inteiro</li> <li>Tempo de resposta: &lt; 2 segundos.</li> </ul>
Critérios de aceitação	Pós-condições

<ul style="list-style-type: none"> <li>• Todos os campos devem retornar com <b>tipagem correta</b> conforme esperado.</li> <li>• Nenhum campo deve vir com valor <b>null</b> indevido ou tipo inconsistente.</li> <li>• A estrutura dos objetos deve estar padronizada.</li> <li>• Tempo de resposta inferior a <b>2000 ms</b>.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração ou impacto nos dados da API.</li> <li>• Apenas leitura e validação.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• <i>Método:</i> <b>GET</b></li> <li>• <i>Endpoint:</i> <b><i>https://fakestoreapi.com/products</i></b></li> </ul>	<b>Sucesso</b>
<b>Evidência(s)</b>	



## Cenário de Teste 07

### Cenário de Teste: Performance - Tempo de resposta da listagem de produtos

ID	Descrição	
TC007	Avaliar se o endpoint <b>GET /products</b> responde dentro do tempo estabelecido no SLA (Service Level Agreement). O objetivo é garantir que a resposta da listagem completa de produtos seja retornada em <b>menos de 2 segundos</b> , mesmo em condições normais de uso.	
Prioridade		Tipo de Teste

Alta	<ul style="list-style-type: none"> <li>• Performance</li> <li>• API – Tempo de resposta</li> <li>• Teste não funcional</li> </ul>
<b>Pré-condições</b>	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• Ferramenta de testes de API funcional (Postman).</li> <li>• Conexão com a internet.</li> </ul>	
<b>Passos</b>	
<ol style="list-style-type: none"> <li>1. Abrir ferramenta de testes com medição de tempo.</li> <li>2. Criar uma requisição do tipo <b>GET</b> para o endpoint: <a href="https://fakestoreapi.com/products">https://fakestoreapi.com/products</a></li> <li>3. Executar a requisição.</li> <li>4. Observar o tempo total de resposta registrado pela ferramenta.</li> <li>5. Anotar os tempos e verificar se estão abaixo de 2 segundos.</li> <li>6. Validar se os dados estão completos (corpo da resposta é um array de produtos).</li> </ol>	
<b>Resultado Esperado</b>	<b>Resultado Obtido</b>
<ul style="list-style-type: none"> <li>• Tempo de resposta <math>\leq</math> 2000 ms</li> <li>• Status HTTP: <b>200 OK</b></li> <li>• Corpo: array JSON com os produtos</li> <li>• Nenhum erro ou timeout</li> </ul>	<ul style="list-style-type: none"> <li>• Execução 3: 240 ms</li> <li>• Status: 200 OK</li> </ul>
<b>Critérios de aceitação</b>	<b>Pós-condições</b>

- A resposta deve ser retornada em **até 2 segundos**.
- A resposta deve conter um array de produtos completos.
- Nenhum erro de rede ou falha do servidor.
- Consistência nos resultados entre múltiplas execuções.
- Sem degradação perceptível de desempenho.

- Nenhum impacto ou alteração nos dados da API.
- Somente leitura de dados.

#### Dados de teste

#### Status

- *Método: GET*
- *Endpoint:*  
*<https://fakestoreapi.com/products>*
- *Nenhum corpo de requisição necessário*

**Sucesso**

#### Evidência(s)

The screenshot displays a Postman interface for a GET request to `https://fakestoreapi.com/products`. The 'Scripts' tab is active, showing a pre-request script that tests the response time: `pm.test("Tempo de resposta ≤ 2000 ms", function () { pm.expect(pm.response.responseTime).to.be.below(2000); // Verificar se o tempo de resposta é menor que 2000 ms });`. The 'Test Results' tab at the bottom shows a single result: **PASSED** Tempo de resposta ≤ 2000 ms. The status bar indicates a 200 OK response with a 240 ms response time and 4.65 KB of data.

---

## Carrinho (/carts)

### Cenário de Teste 01

Cenário de Teste: Listar todos os carrinhos	
ID	Descrição
TC008	Verificar se o endpoint <b>GET /carts</b> retorna corretamente a lista completa de carrinhos cadastrados. O teste valida a integridade da resposta, incluindo estrutura, status e presença de campos esperados como <b>id</b> , <b>userId</b> , <b>date</b> e <b>products</b> .
Prioridade	
Alta	<ul style="list-style-type: none"><li>• Funcional</li><li>• API – Teste positivo</li><li>• Black-box</li></ul>
Pré-condições	
<ul style="list-style-type: none"><li>• A API deve estar disponível em <b>https://fakestoreapi.com</b>.</li><li>• Ferramenta de testes de API funcional (Postman).</li><li>• Conexão com a internet.</li></ul>	
Passos	
<ol style="list-style-type: none"><li>1. Abrir a ferramenta de testes de API.</li><li>2. Criar uma nova requisição do tipo <b>GET</b>.</li><li>3. Informar a URL: <b>https://fakestoreapi.com/carts</b>.</li></ol>	

4. Enviar a requisição.
5. Verificar se o status de resposta é **200 OK**.
6. Analisar o corpo da resposta:
  - Confirmar que é um **array**.
  - Verificar se cada objeto contém:
    - **id** (inteiro)
    - **userId** (inteiro)
    - **date** (string no formato de data)
    - **products** (array de objetos com **productId** e **quantity**)
7. Validar se o array não está vazio e a estrutura está padronizada.
8. Confirmar que o tempo de resposta é inferior a 2 segundos.

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>Status HTTP: <b>200 OK</b></li> <li>A resposta deve ser um array, mesmo que contenha apenas um item.</li> <li>Nenhum campo sensível presente.</li> <li>Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>Status: 200 OK</li> <li>Estrutura: Array de objetos</li> <li>Campos esperados presentes: sim</li> <li>Dados consistentes: sim</li> <li>Tempo de resposta: &lt; 2 segundos.</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>A resposta deve conter status <b>200 OK</b>. O corpo da resposta deve ser um <b>array de carrinhos</b>.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhuma alteração nos dados da API.</li> <li>Apenas leitura dos carrinhos.</li> </ul>



- Cada carrinho deve conter os campos esperados (`id`, `userId`, `date`, `products`).
- O campo `products` deve conter pelo menos um produto com `productId` e `quantity`.
- Tempo de resposta inferior a **2000 ms**..

#### Dados de teste

#### Status

- Método: `GET`
- Endpoint:  
`https://fakestoreapi.com/carts`
- Nenhum parâmetro ou body necessário

**Sucesso**

#### Evidência(s)

The screenshot shows a REST client interface with the following details:

- Request:** GET `https://fakestoreapi.com/carts`
- Scripts:**

```

1 let jsonData = pm.response.json();
2
3 // Verifica se a resposta é um array
4 pm.test("Resposta é um array", function () {
5     pm.expect(Array.isArray(jsonData)).to.be.true;
6     pm.expect(jsonData.length).to.be.above(0); // Verifica se o array não está vazio
7 });

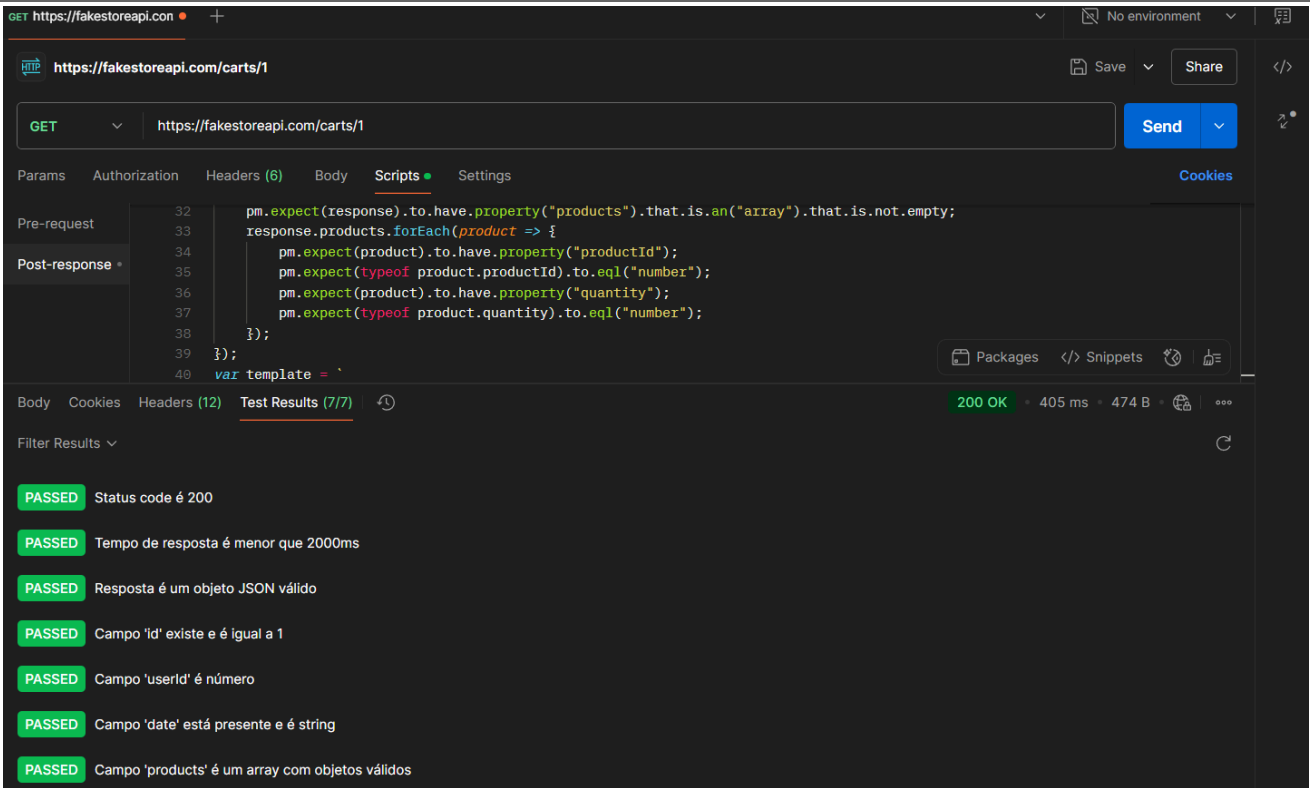
```
- Test Results (4/4):**
  - PASSED** Resposta é um array
  - PASSED** Status HTTP é 200 OK
  - PASSED** Tempo de resposta < 2000 ms
  - PASSED** Objetos do array possuem os campos esperados
- Response Summary:** 200 OK, 524 ms, 559 B

## Cenário de Teste 02

Cenário de Teste: Buscar carrinho por ID existente		
ID	Descrição	
TC009	Verificar se o endpoint <code>GET /carts/{id}</code> retorna corretamente os dados de um carrinho específico ao utilizar um ID válido. O teste assegura que os campos retornados estão completos e corretos, incluindo o <code>id</code> , <code>userId</code> , <code>date</code> e a lista de <code>products</code> .	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Funcional</li><li>• API – Teste positivo</li><li>• Black-box</li></ul>
Pré-condições		
<ul style="list-style-type: none"><li>• A API deve estar acessível em <code>https://fakestoreapi.com</code>.</li><li>• Ferramenta de testes de API configurada (Postman).</li><li>• Conexão com a internet.</li><li>• ID de carrinho válido previamente conhecido (ex: 1).</li></ul>		
Passos		
<ol style="list-style-type: none"><li>1. Abrir a ferramenta de testes de API.</li><li>2. Criar uma requisição do tipo <code>GET</code>.</li><li>3. Utilizar o endpoint: <code>https://fakestoreapi.com/carts/1</code> (ou outro ID conhecido existente).</li><li>4. Enviar a requisição.</li><li>5. Verificar se o status da resposta é <code>200 OK</code>.</li><li>6. Validar se o corpo da resposta contém:</li></ol>		

- **id**: número inteiro e corresponde ao ID requisitado.
  - **userId**: número inteiro.
  - **date**: string no formato ISO (data).
  - **products**: array de produtos com:
    - **productId** (inteiro)
    - **quantity** (inteiro)
7. Confirmar se os dados são consistentes e completos.
8. Validar o tempo de resposta (deve ser < 2 segundos).

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: 200 OK</li> <li>• Todos os campos presentes e corretos.</li> <li>• Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200 OK</li> <li>• ID retornado: 1 Estrutura do carrinho: Completa</li> <li>• Tempo de resposta: &lt; 2 segundos.</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A API deve retornar status <b>200 OK</b>.</li> <li>• O corpo da resposta deve conter todos os campos esperados (<b>id</b>, <b>userId</b>, <b>date</b>, <b>products</b>).</li> <li>• O campo <b>products</b> deve ser um array com objetos contendo <b>productId</b> e <b>quantity</b>.</li> <li>• O <b>id</b> retornado deve ser o mesmo que o enviado na requisição.</li> <li>• O tempo de resposta deve ser inferior a <b>2000 ms</b>.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração nos dados da API.</li> <li>• Apenas leitura de informações.</li> </ul>

Dados de teste	Status
<ul style="list-style-type: none"><li>Método: <i>GET</i></li><li>Endpoint: <i>https://fakestoreapi.com/carts/1</i></li></ul>	<b>Sucesso</b>
Evidência(s)	
	

## Cenário de Teste 04

Cenário de Teste: Criar carrinho com dados válidos	
ID	Descrição

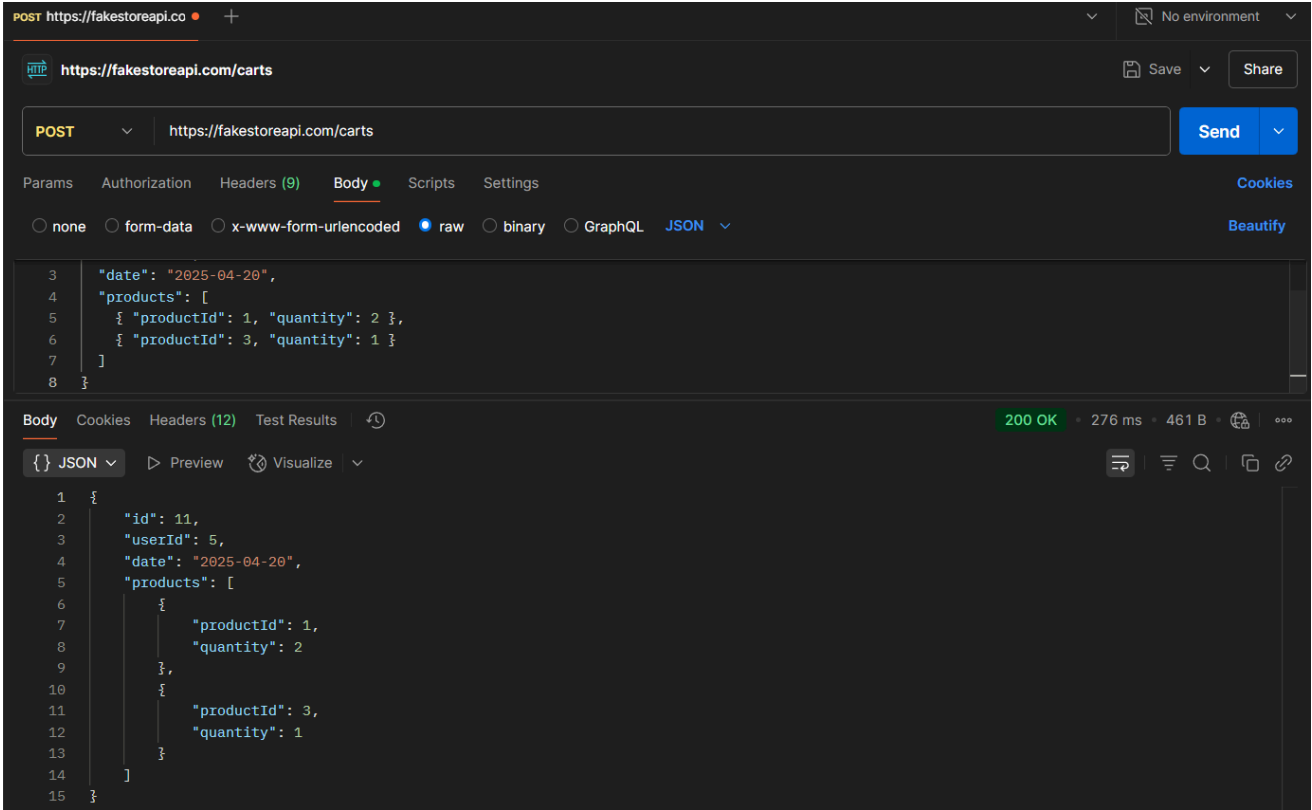
TC010	<p>Verificar se o endpoint <b>POST /carts</b> permite criar um novo carrinho com dados válidos, como <b>userId</b> e lista de produtos com <b>productId</b> e <b>quantity</b>. O teste valida o comportamento da API em operações de criação, confirmando o retorno do status apropriado, a geração de um novo <b>id</b> e o reflexo correto dos dados enviados.</p>
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• API – Teste positivo</li> <li>• Criação de recurso</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar acessível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• Ferramenta de testes de API configurada (Postman).</li> <li>• Conexão com a internet.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma requisição do tipo <b>POST</b>.</li> <li>3. Utilizar o endpoint: <a href="https://fakestoreapi.com/carts">https://fakestoreapi.com/carts</a>.</li> <li>4. No corpo da requisição, enviar um JSON com <b>userId</b>, <b>date</b> e <b>products</b> com <b>productId</b> e <b>quantity</b>.</li> <li>5. Enviar a requisição.</li> <li>6. Verificar se o <b>status de resposta</b> é <b>200 OK</b> ou <b>201 Created</b>.</li> <li>7. Confirmar que o corpo da resposta contém: <ul style="list-style-type: none"> <li>• Um novo <b>id</b> gerado para o carrinho.</li> </ul> </li> </ol>	

- Os dados enviados no corpo, refletidos corretamente.

8. Validar se a estrutura da resposta está adequada.

9. Confirmar que o tempo de resposta está dentro do SLA (< 2s).

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: 200 OK ou 201 Created</li> <li>• Todos os campos refletidos corretamente</li> <li>• Tempo de resposta &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 201 Created</li> <li>• ID gerado: 51</li> <li>• Dados refletidos corretamente: sim</li> <li>• Tempo de resposta:</li> </ul>
Critérios de aceitação	Pós-condições
<p>A API deve retornar 200 OK ou 201 Created.</p> <p>O corpo da resposta deve conter:</p> <ul style="list-style-type: none"> <li>• Um novo id para o carrinho criado.</li> <li>• Os dados enviados, refletidos corretamente.</li> </ul> <p>Nenhum erro ou campo inconsistente deve ser retornado.</p>	<ul style="list-style-type: none"> <li>• Um novo carrinho deve ser criado na base de dados simulada.</li> <li>• O carrinho pode ser consultado por GET /carts/{id} (se persistido).</li> </ul>

O tempo de resposta deve ser inferior a <b>2 segundos</b> .	
Dados de teste	Status
<ul style="list-style-type: none"><li>Método: <b>POST</b></li><li>Endpoint: <b>https://fakestoreapi.com/carts</b></li><li>Headers:</li><li><b>Content-Type: application/json</b></li></ul>	<b>Sucesso</b>
Evidência(s)	
	

Cenário de Teste 05

## Cenário de Teste: Criar carrinho com dados inválidos

ID	Descrição	
TC011	Verifica o comportamento da API ao tentar criar um carrinho utilizando um payload inválido, com campos obrigatórios ausentes, malformados ou com tipos incorretos. O objetivo é garantir que a API realize a <b>validação adequada dos dados de entrada</b> , retornando erro controlado ( <b>400 Bad Request</b> ) e mensagem clara.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Negativo</li><li>• Funcional</li><li>• Validação de entrada</li><li>• Black-box</li></ul>
Pré-condições		
<ul style="list-style-type: none"><li>• A API deve estar acessível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li><li>• Ferramenta de testes de API configurada (Postman).</li><li>• Conexão com a internet.</li></ul>		
Passos		
<ol style="list-style-type: none"><li>1. Abrir a ferramenta de testes de API.</li><li>2. Criar uma requisição do tipo <b>POST</b>.</li><li>3. Utilizar o endpoint: <a href="https://fakestoreapi.com/carts">https://fakestoreapi.com/carts</a>.</li><li>4. Enviar um dos payloads inválidos listados acima.</li><li>5. Enviar a requisição.</li><li>6. Verificar se o <b>status de resposta</b> é <b>400 Bad Request</b> (ou similar como <b>422 Unprocessable Entity</b>).</li><li>7. Validar se a resposta inclui:</li></ol>		



- Mensagem clara sobre o erro.
  - Indicação do(s) campo(s) com problema.
8. Verificar se **nenhum carrinho foi criado**.
9. Confirmar tempo de resposta adequado (< 2s).

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: <b>400 Bad Request</b></li> <li>• Mensagem clara (exemplos): <ul style="list-style-type: none"> <li>{ <ul style="list-style-type: none"> <li>"error": "Campo 'products' é obrigatório"</li> </ul> </li> <li>}</li> <li>ou</li> <li>{ <ul style="list-style-type: none"> <li>"error": "products deve ser do tipo número"</li> </ul> </li> <li>}</li> </ul> </li> <li>• Nenhum ID gerado</li> <li>• Tempo de resposta &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>• A API <b>não retornou o status apropriado</b> (400 ou 4224), respondendo incorretamente com status <b>200 OK</b>.</li> <li>• A <b>mensagem de erro</b> também não foi apresentada</li> <li>• ID gerado</li> <li>• Carrinho criado com valores inválidos</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A API <b>não deve criar um novo carrinho</b>.</li> <li>• Deve retornar status <b>400, 422</b> ou equivalente indicando erro de validação.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhum carrinho deve ser persistido.</li> <li>• A API deve permanecer estável e funcional.</li> </ul>

- A mensagem de erro deve ser clara e específica.
- Nenhum dado incorreto ou parcial deve ser salvo.
- Tempo de resposta inferior a 2 segundos.

### Dados de teste

### Status

- Payload vazio:

```
{}
```

- Ausência de campo **products**

**Bug Encontrado**

### Evidência(s)

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** https://fakestoreapi.com/carts
- Body (raw JSON):**

```
{
  "userId": 2,
  "date": "2025-04-20",
  "products": [
    { "productId": "abc", "quantity": 2 }
  ]
}
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
{
  "id": 11,
  "userId": 2,
  "date": "2025-04-20",
  "products": [
    {
      "productId": "abc",
      "quantity": 2
    }
  ]
}
```

## Cenário de Teste 06

Cenário de Teste: Criar carrinho com dados válidos	
ID	Descrição
TC012	Verificar se o endpoint <code>PUT /carts/{id}</code> permite atualizar os dados de um carrinho existente, modificando <code>userId</code> , <code>date</code> ou a lista de produtos. Este teste avalia a <b>capacidade da API em modificar recursos corretamente</b> e refletir as alterações no retorno da requisição.
Prioridade	
Alta	<ul style="list-style-type: none"><li>• Positivo</li><li>• Funcional</li><li>• API – Atualização de recurso</li><li>• Black-box</li></ul>
Pré-condições	
<ul style="list-style-type: none"><li>• A API deve estar acessível em <code>https://fakestoreapi.com</code>.</li><li>• Um carrinho válido já deve existir (ex: ID 5).</li><li>• Ferramenta de testes de API funcionando.</li><li>• Conexão com a internet.</li></ul>	
Passos	
<ol style="list-style-type: none"><li>1. Abrir a ferramenta de testes de API.</li><li>2. Criar uma requisição <code>PUT</code> com o endpoint <code>https://fakestoreapi.com/carts/5</code>.</li><li>3. Inserir o corpo da requisição com os novos dados.</li><li>4. Enviar a requisição.</li><li>5. Verificar o <b>status de resposta</b>: deve ser <code>200 OK</code>.</li></ol>	

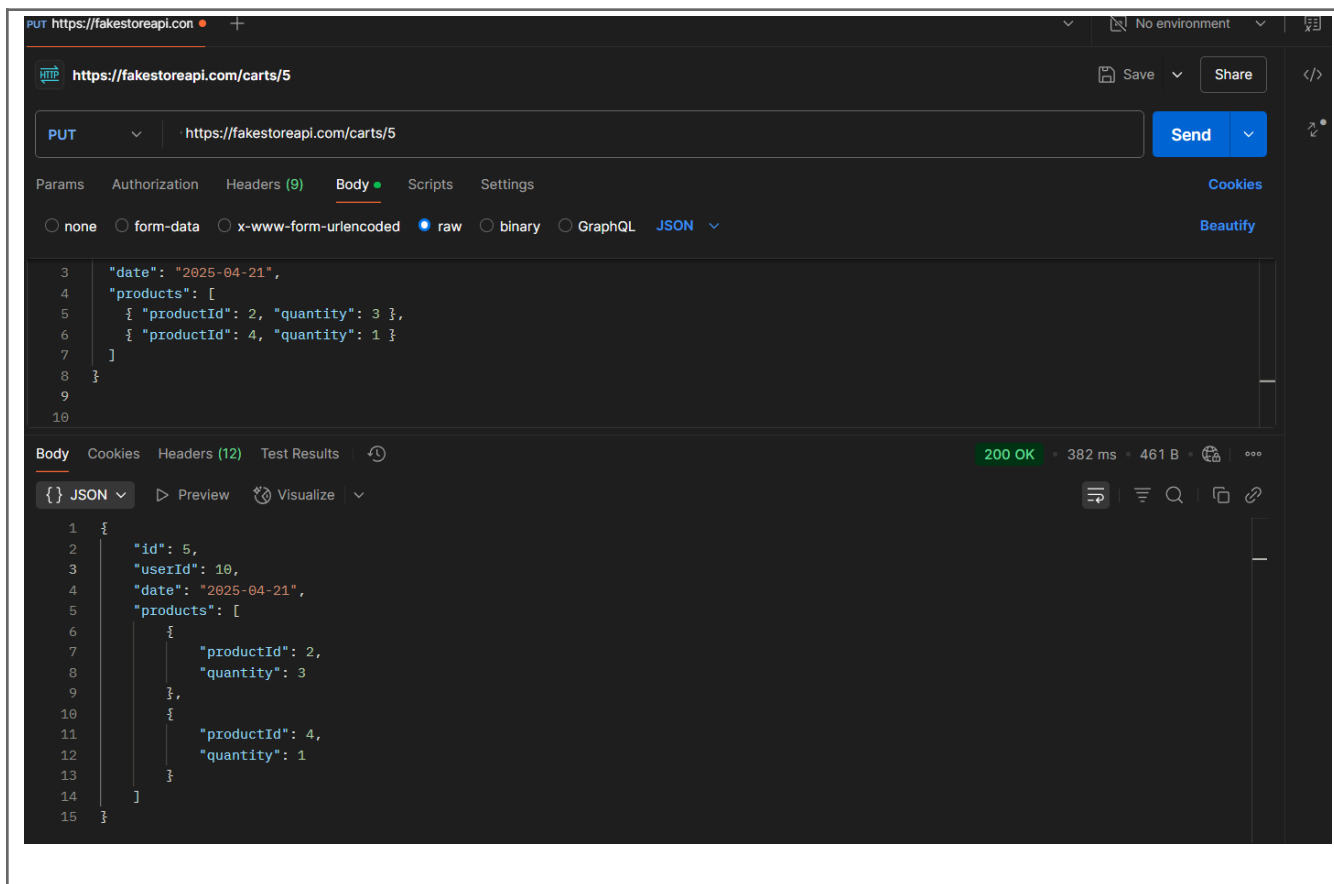
6. Validar se o corpo da resposta reflete as mudanças corretamente.
7. Verificar se o **id** continua o mesmo (5).
8. Confirmar tempo de resposta dentro do SLA (< 2s).

Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>Status: 200</li> <li>Dados atualizados</li> <li>Tempo de resposta &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>Status: 200 OK ID: 5</li> <li>Dados atualizados: Confirmado</li> <li>Tempo de resposta: 382 ms</li> </ul>
CrITÉRIOS de aceitação	Pós-condições
<p>A API deve retornar 200 OK.</p> <p>O corpo da resposta deve conter:</p> <ul style="list-style-type: none"> <li>O mesmo <b>id</b> do carrinho atualizado.</li> <li>Os dados atualizados corretamente.</li> </ul> <p>Nenhuma inconsistência nos tipos de dados.</p> <p>Tempo de resposta inferior a 2 segundos.</p>	<ul style="list-style-type: none"> <li>O carrinho com ID 5 deve conter os novos dados enviados.</li> <li>Pode ser verificado com <b>GET /carts/5</b>.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: PUT</li> <li>Endpoint: <b>https://fakestoreapi.com/carts/5</b></li> </ul>	<b>Sucesso</b>

- Corpo da requisição (exemplo de atualização):

```
{  
  
  "userId": 10,  
  
  "date": "2025-04-21",  
  
  "products": [  
  
    { "productId": 2, "quantity": 3 },  
  
    { "productId": 4, "quantity": 1 }  
  
  ]  
  
}
```

**Evidência(s)**



## Cenário de Teste 07

Cenário de Teste: Atualizar carrinho com ID inexistente		
ID	Descrição	
TC013	Este teste avalia o comportamento da API ao tentar atualizar um carrinho que <b>não existe na base de dados</b> , utilizando o método <b>PUT</b> com um ID inválido (ex: <b>9999</b> ). O objetivo é garantir que a API <b>não crie recursos por engano</b> nem retorne sucesso indevido.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Negativo</li><li>• Funcional</li><li>• Validação de atualização</li></ul>

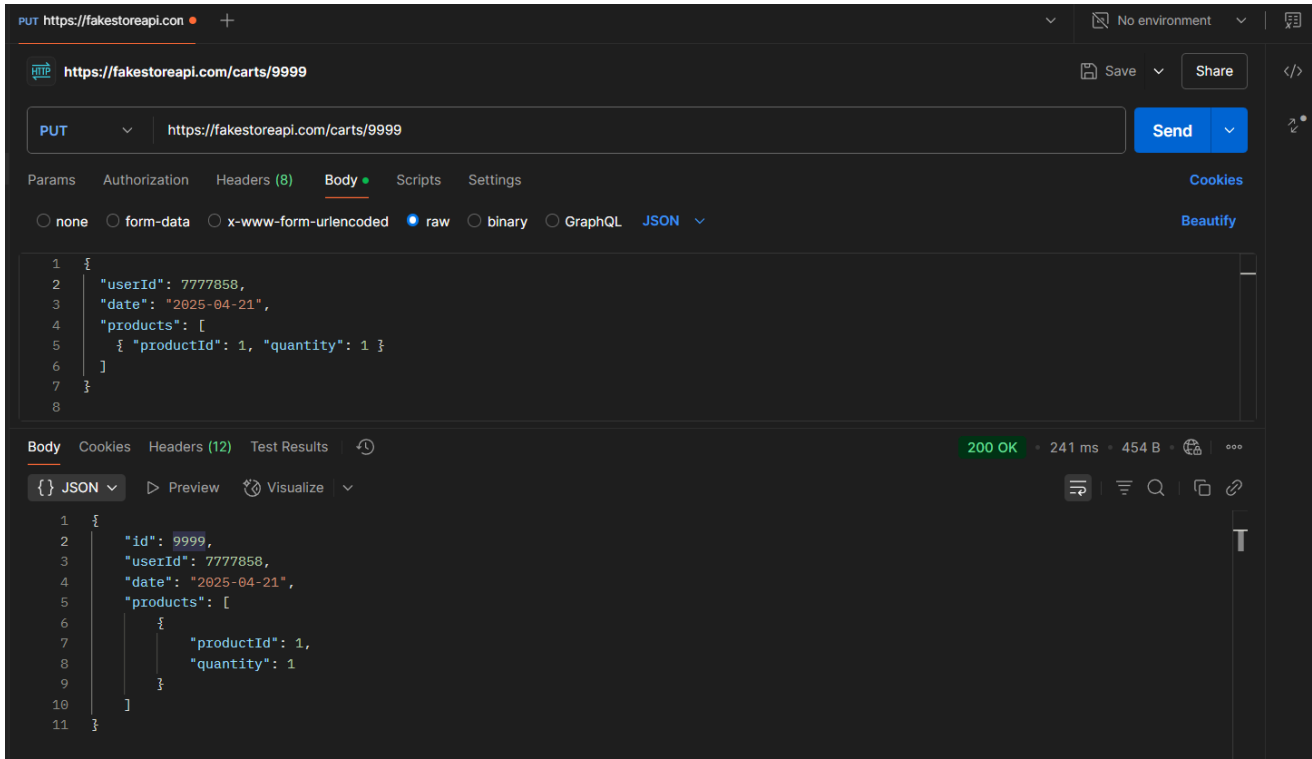
	<ul style="list-style-type: none"> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar acessível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• O ID 9999 não deve existir na base.</li> <li>• Ferramenta de testes de API operante.</li> <li>• Conexão com a internet.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Abrir a ferramenta de testes de API.</li> <li>2. Criar uma requisição PUT para o endpoint <a href="https://fakestoreapi.com/carts/9999">https://fakestoreapi.com/carts/9999</a>.</li> <li>3. Inserir o corpo da requisição com dados válidos.</li> <li>4. Enviar a requisição.</li> <li>5. Observar o <b>status da resposta</b>: espera-se 404 Not Found, 400, ou uma mensagem indicando que o recurso não foi encontrado.</li> <li>6. Verificar se <b>nenhum carrinho foi criado</b>.</li> <li>7. Confirmar o tempo de resposta dentro do esperado (&lt; 2s).</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: 404 Not Found ou similar</li> <li>• Mensagem de erro (exemplo): <pre>{   "error": "Carrinho com ID 9999 não encontrado" }</pre> </li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200</li> <li>• Carrinho criado: Sim</li> </ul>

<ul style="list-style-type: none"> <li>Nenhum ID criado</li> <li>Tempo de resposta &lt; 2 segundos</li> </ul>	
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>A API <b>não deve criar ou atualizar nenhum carrinho.</b></li> <li>O status deve ser <b>404</b>, <b>400</b> ou mensagem de erro coerente.</li> <li>O ID <b>9999</b> não deve passar a existir após a tentativa.</li> <li>Mensagem clara sobre o erro.</li> <li>Tempo de resposta inferior a 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhuma alteração ou criação de recurso deve ocorrer.</li> <li>A integridade dos dados deve ser mantida.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: <b>PUT</b></li> <li>Endpoint: <a href="https://fakestoreapi.com/carts/9999">https://fakestoreapi.com/carts/9999</a></li> <li>Headers: <b>Content-Type: application/json</b></li> <li>Corpo da requisição (dados fictícios):   <pre>{   "userId": 7777858,   "date": "2025-04-22",   "products": [</pre> </li> </ul>	<p><b>Bug Encontrado</b></p>



<pre>{ "productId": 1, "quantity": 1 }  ]</pre>	
---	--

# Evidência(s)



## Cenário de Teste 08

Cenário de Teste: Verificação de tipos de dados	
ID	Descrição
TC014	Este teste tem como objetivo <b>validar os tipos de dados retornados pela API</b> no endpoint de carrinhos. A intenção é garantir que os campos como

	<code>products</code> , <code>productId</code> , <code>quantity</code> , <code>userId</code> , entre outros, estejam com seus <b>tipos corretos</b> , conforme esperado pelo contrato da API.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"> <li>• Validação</li> <li>• Funcional</li> <li>• Contrato de API</li> <li>• Black-box</li> </ul>
Pré-condições		
<ul style="list-style-type: none"> <li>• A API deve estar acessível em <code>https://fakestoreapi.com</code>.</li> <li>• Deve haver carrinhos disponíveis para listagem (GET <code>/carts</code>).</li> <li>• Ferramenta de testes de API funcionando.</li> <li>• Conexão com a internet.</li> </ul>		
Passos		
<ol style="list-style-type: none"> <li>1. Realizar uma requisição <code>GET</code> no endpoint <code>https://fakestoreapi.com/carts</code>.</li> <li>2. Verificar se o status da resposta é <code>200 OK</code>.</li> <li>3. Analisar os campos retornados no corpo da resposta e validar os <b>tipos de dados</b>:               <ul style="list-style-type: none"> <li>• <code>id</code>: número inteiro</li> <li>• <code>userId</code>: número inteiro</li> <li>• <code>date</code>: string (formato ISO)</li> <li>• <code>products</code>: array                   <ul style="list-style-type: none"> <li>○ Dentro de <code>products</code>:                       <ul style="list-style-type: none"> <li>■ <code>productId</code>: número inteiro</li> <li>■ <code>quantity</code>: número inteiro</li> </ul> </li> </ul> </li> </ul> </li> <li>4. Verificar se <b>todos os objetos do array</b> seguem esse padrão.</li> <li>5. Validar que não há campos com tipo inesperado, valores nulos indevidos ou campos malformados.</li> </ol>		

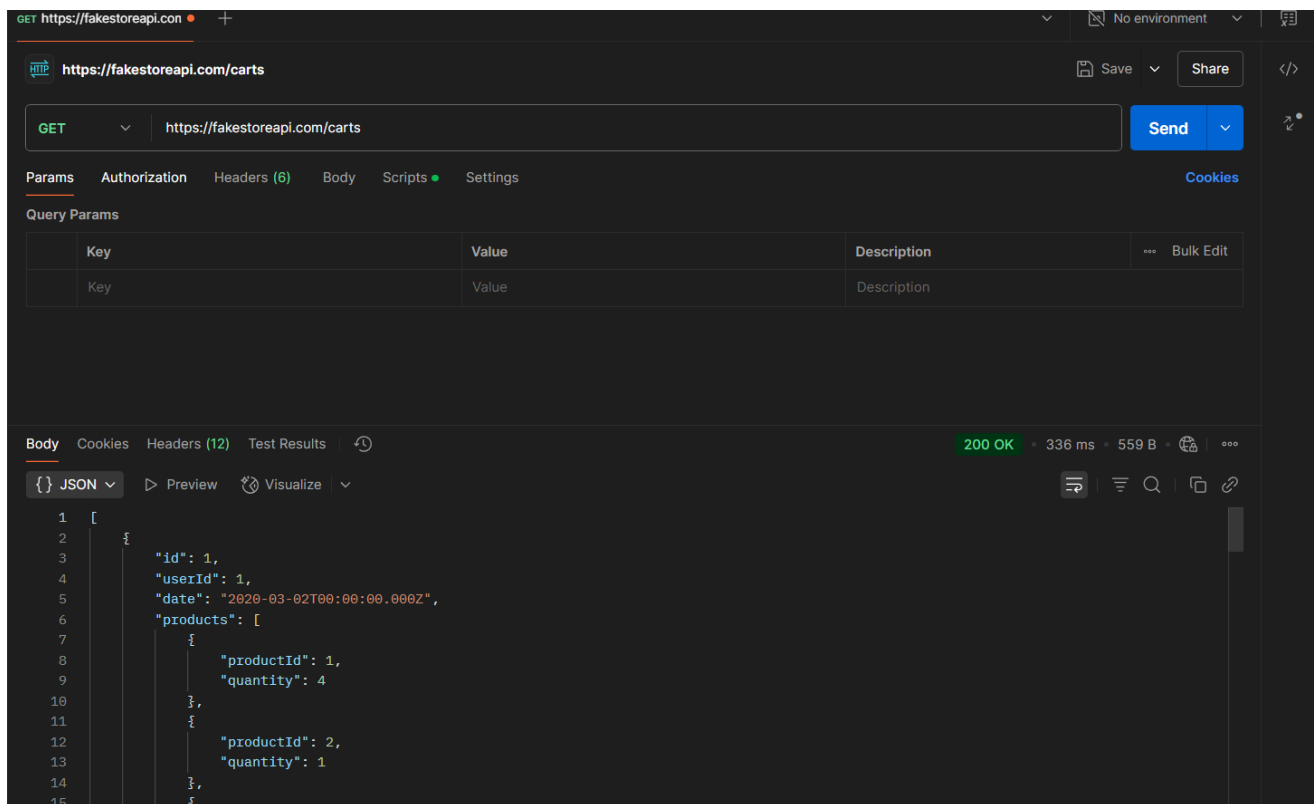
6. Confirmar tempo de resposta inferior a 2 segundos.

Resultado Esperado	Resultado Obtido
<p>Status: 200 OK</p> <p>Tipos de dados confirmados:</p> <ul style="list-style-type: none"><li>• <b>id</b>: integer</li><li>• <b>userId</b>: integer</li><li>• <b>date</b>: string ISO</li><li>• <b>products</b>: array<ul style="list-style-type: none"><li>◦ <b>productId</b>: integer</li><li>◦ <b>quantity</b>: integer</li></ul></li></ul> <p>Estrutura consistente em todos os objetos</p> <ul style="list-style-type: none"><li>• Tempo de resposta &lt; 2 segundos</li></ul>	<ul style="list-style-type: none"><li>• Status: 200 OK</li><li>• Tipos: Corretos</li><li>• Erros encontrados: Nenhum</li><li>• Tempo de resposta: 336 ms</li></ul>
CrITÉRIOS de aceitaÇ���	P��s-condi���es
<ul style="list-style-type: none"><li>• Todos os campos retornados devem conter os <b>tipos corretos conforme o contrato</b>.</li><li>• A estrutura dos objetos deve ser consistente em todos os itens.</li><li>• Nenhum campo pode ter tipo incompat��vel ou valor inesperado.</li><li>• Tempo de resposta inferior a 2 segundos.</li></ul>	<ul style="list-style-type: none"><li>• Nenhuma modifica����� realizada — apenas leitura.</li><li>• N��o h�� impacto nos dados da API.</li></ul>
Dados de teste	Status

- Método: **GET**
- Endpoint:  
**https://fakestoreapi.com/carts**

**Sucesso**

## Evidência(s)



## Cenário de Teste 9

### Cenário de Teste: Verificação de tipos de dados

ID	Descrição
----	-----------

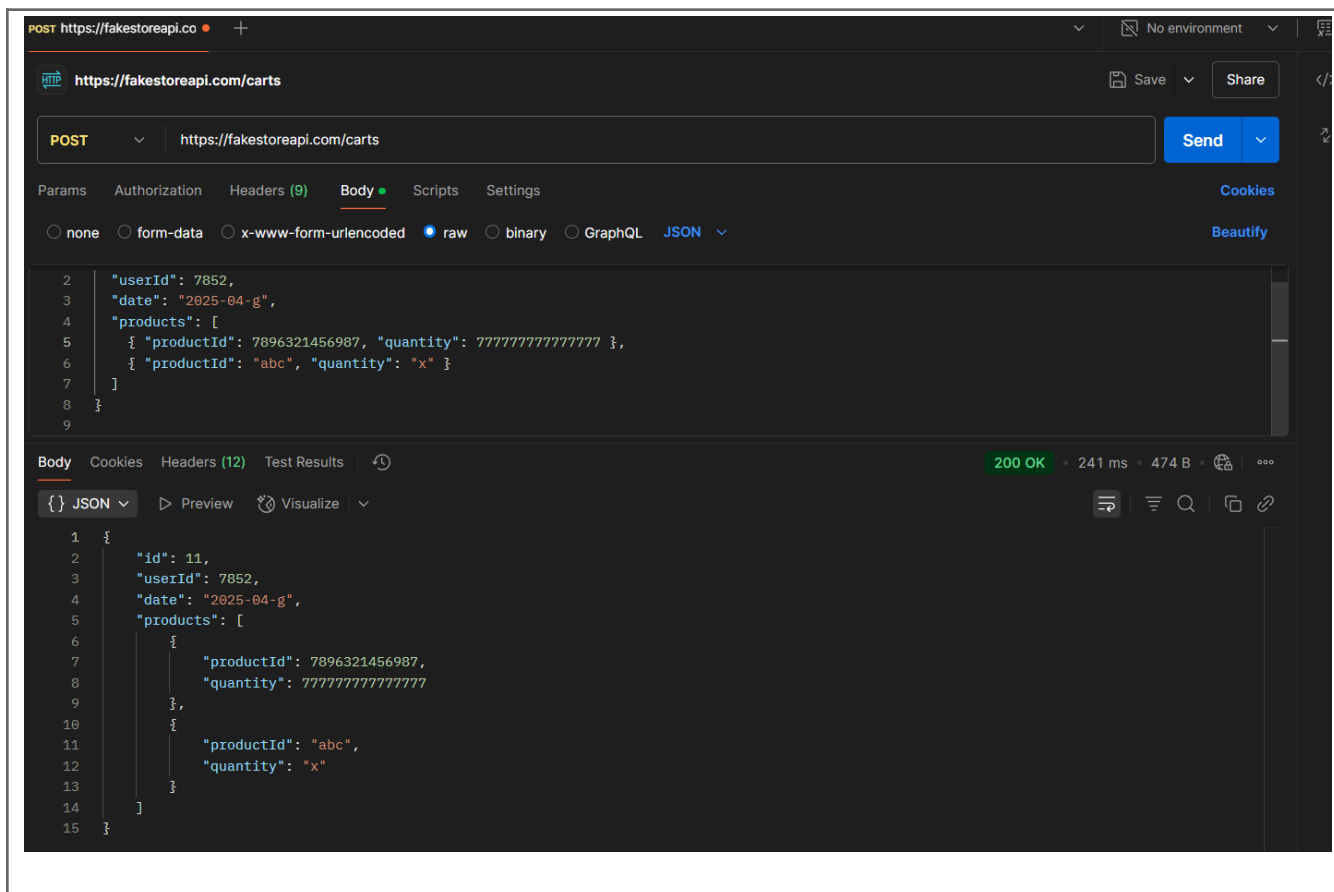
TC015	<p>Este teste avalia se a API de carrinhos <b>impede a criação ou alteração de dados com informações inconsistentes ou inválidas</b>, como a tentativa de adicionar produtos inexistentes, <b>productId</b> inválidos ou campos fora do padrão. O objetivo é garantir que a API possua <b>validações e restrições adequadas</b> para preservar a integridade dos dados.</p>
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Segurança</li> <li>• Validação</li> <li>• Funcional / Negativo</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar acessível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• Ferramenta de testes de API instalada e operante.</li> <li>• Conexão ativa com a internet.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Criar requisição <b>POST</b> no endpoint <b>/carts</b> com o payload acima.</li> <li>2. Enviar a requisição.</li> <li>3. Verificar o status da resposta (esperado: <b>400</b>, <b>422</b>, ou erro tratado).</li> <li>4. Analisar se a API rejeitou corretamente os dados inválidos.</li> <li>5. Validar a mensagem de erro: deve ser clara, indicando o problema.</li> <li>6. Realizar um <b>GET /carts</b> para garantir que nenhum carrinho inválido foi criado.</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: <b>400 Bad Request</b> ou <b>422 Unprocessable Entity</b></li> </ul>	<ul style="list-style-type: none"> <li>• A API <b>não retornou o status apropriado</b> (400 ou 402),</li> </ul>

<ul style="list-style-type: none"> <li>Mensagem: <pre> {    "error": "Produto inválido ou tipo de dado incorreto"  } </pre> </li> <li>Nenhum carrinho criado</li> <li>Tempo de resposta &lt; 2 segundos</li> </ul>	<p>respondendo incorretamente com status <b>200 OK</b>.</p> <ul style="list-style-type: none"> <li>Carrinho criado com valores inválidos</li> <li>Novos Dados gerados</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>A API deve <b>impedir</b> a criação de carrinhos com produtos inválidos.</li> <li>Deve retornar um status de erro adequado (<b>400</b>, <b>422</b>, etc.). A mensagem de erro deve ser <b>clara, específica e amigável</b>.</li> <li>Nenhum recurso deve ser criado com dados inconsistentes.</li> <li>Tempo de resposta inferior a 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhum carrinho inválido deve constar na base de dados.</li> <li>Os dados existentes devem permanecer íntegros e inalterados.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: <b>POST</b></li> <li>Endpoint: <a href="https://fakestoreapi.com/carts">https://fakestoreapi.com/carts</a></li> </ul>	<p><b>Bug Encontrado</b></p>

- Headers: **Content-Type:**  
**application/json**
- Corpo da requisição com dados inválidos:

```
{  
  
  "userId": 7852,  
  
  "date": "2025-04-g",  
  
  "products": [  
  
    { "productId": 7896321456987,  
      "quantity": 7777777777777777 },  
  
    { "productId": "abc", "quantity": "x" }  
  
  ]  
  
}
```

**Evidência(s)**



## Usuários (/users)

### Cenário de Teste 01

Cenário de Teste: Listar todos os usuários	
ID	Descrição
TC016	Este teste tem como objetivo verificar se o endpoint de usuários da API retorna corretamente <b>todos os usuários cadastrados</b> , garantindo que o corpo da resposta esteja estruturado corretamente, com os campos esperados como <b>id</b> , <b>email</b> , <b>username</b> , entre outros.

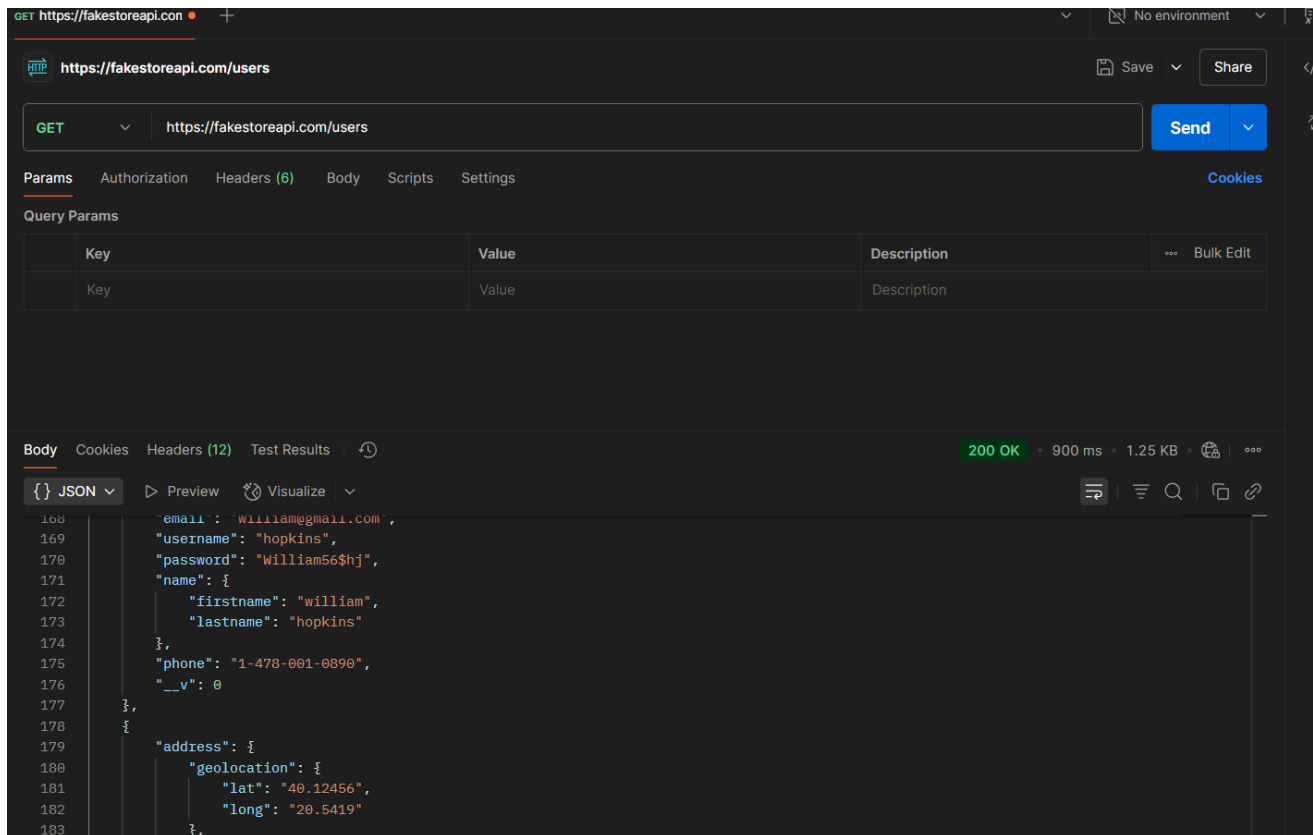


Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• Positivo</li> <li>• Listagem de dados</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar disponível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• A ferramenta de testes de API deve estar funcionando.</li> <li>• Deve haver ao menos 1 usuário cadastrado na base.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Realizar uma requisição <b>GET</b> para o endpoint <b>/users</b>.</li> <li>2. Verificar se a resposta retorna o <b>status 200 OK</b>.</li> <li>3. Validar se o corpo da resposta é um <b>array de objetos</b>.</li> <li>4. Para cada objeto (usuário), verificar a presença dos campos: <ul style="list-style-type: none"> <li>• <b>id</b>: número inteiro</li> <li>• <b>email</b>: string</li> <li>• <b>username</b>: string</li> <li>• <b>name</b>: objeto com <b>firstname</b> e <b>lastname</b></li> <li>• <b>address</b>: objeto com <b>city</b>, <b>street</b>, <b>zipcode</b>, etc.</li> </ul> </li> <li>5. Confirmar que a estrutura é consistente entre os objetos do array.</li> <li>6. Verificar o tempo de resposta (deve ser inferior a 2 segundos).</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: <b>200 OK</b></li> <li>• Corpo: Array de usuários</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200 OK</li> <li>• Itens retornados: 10 usuários</li> <li>• Validação de campos: OK</li> <li>• Tempo: 900 ms</li> </ul>

<ul style="list-style-type: none"> <li>Campos validados:</li> </ul> <p><code>id</code>: integer</p> <p><code>email</code>: string</p> <p><code>username</code>: string</p> <p><code>name</code>: objeto com <code>firstname</code>, <code>lastname</code></p> <p><code>address</code>: objeto</p> <ul style="list-style-type: none"> <li>Tempo de resposta: &lt; 2000 ms</li> <li>Dados consistentes entre os usuários retornados</li> </ul>	
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>Status da resposta deve ser <code>200 OK</code>.</li> <li>O corpo da resposta deve ser um array de usuários.</li> <li>Cada objeto deve conter os campos essenciais e corretamente estruturados.</li> <li>Nenhum campo essencial pode estar ausente ou com algum tipo de dado inválido.</li> <li>Tempo de resposta inferior a 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhuma alteração é feita nos dados — apenas leitura.</li> <li>O ambiente permanece estável.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: <code>GET</code></li> </ul>	<code>Sucesso</code>

- Endpoint:  
`https://fakestoreapi.com/users`

## Evidência(s)



## Cenário de Teste 02

### Cenário de Teste: Buscar com Dados Válidos

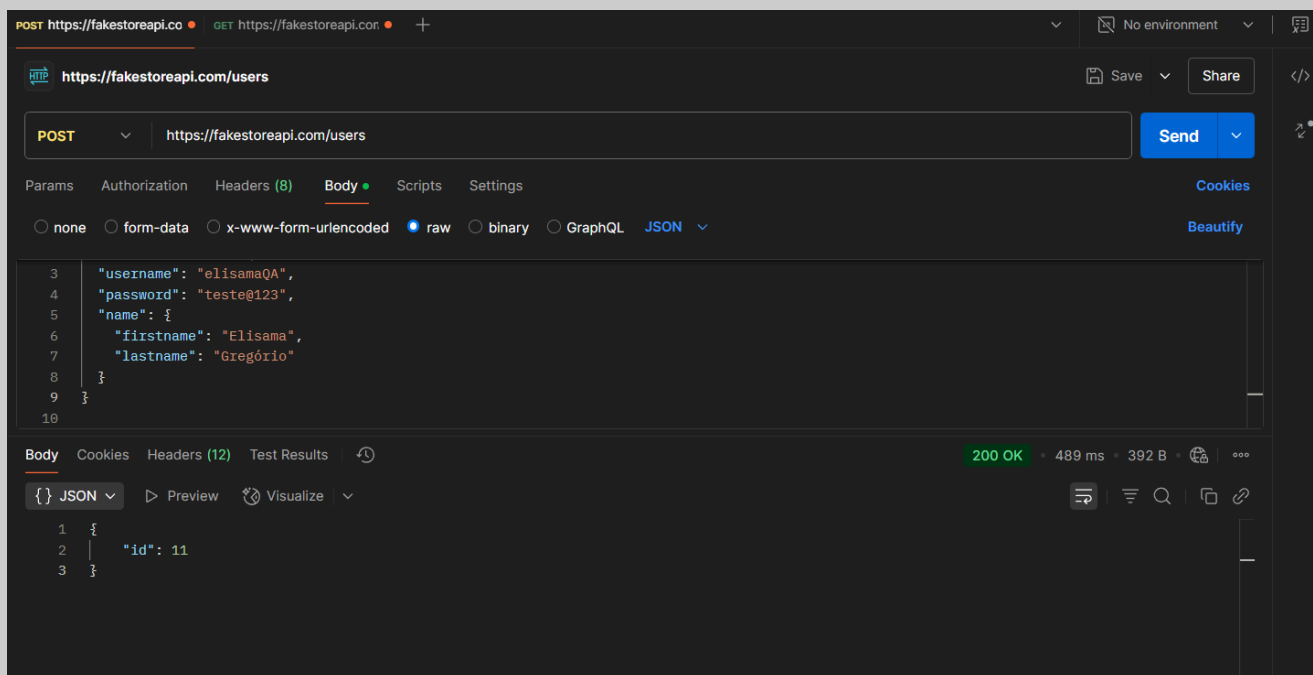
ID	Descrição
----	-----------

TC017	Este teste tem como objetivo validar se o endpoint de criação de usuários aceita corretamente <b>dados válidos</b> e realiza o cadastro com sucesso, retornando o <b>status apropriado, os dados criados e o ID gerado</b> .
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Funcional</li> <li>• Positivo</li> <li>• Criação de recurso</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar acessível em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li> <li>• O endpoint <code>/users</code> deve permitir requisições <code>POST</code>.</li> <li>• Ferramenta de testes de API disponível.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Enviar uma requisição <code>POST</code> para o endpoint <code>/users</code> com o corpo acima.</li> <li>2. Validar se o status da resposta é <code>200 OK</code> ou <code>201 Created</code>.</li> <li>3. Verificar se o corpo da resposta contém: <ul style="list-style-type: none"> <li>• <code>id</code>: gerado pela API</li> <li>• Todos os dados enviados</li> </ul> </li> <li>4. Validar os tipos de dados dos campos retornados.</li> <li>5. Conferir se os dados enviados batem com os retornados.</li> <li>6. Medir tempo de resposta (deve ser inferior a 2 segundos).</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: <code>201 Created</code></li> <li>• Corpo da resposta:</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200</li> <li>• ID: 11</li> </ul>

<pre>{   "id": 21,   "email": "elisama.qa@teste.com",   "username": "elisamaQA", }</pre> <ul style="list-style-type: none"> <li>• Tempo de resposta: &lt; 2 segundos</li> <li>• Dados correspondem ao payload enviado</li> </ul>	<ul style="list-style-type: none"> <li>• Campos conferem com o envio: Não</li> <li>• Dados não correspondem com o payload enviado</li> </ul>
CrITÉRIOS de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A resposta deve ser 200 OK ou 201 Created.</li> <li>• A API deve retornar os dados do usuário criado com um id único.</li> <li>• Todos os campos devem ser persistidos corretamente.</li> <li>• Os tipos de dados devem estar corretos.</li> <li>• Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>• Um novo usuário válido será criado e armazenado.</li> <li>• O ID gerado poderá ser reutilizado para consultas futuras (ex: GET /users/{id}).</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: POST</li> <li>• Endpoint: <a href="https://fakestoreapi.com/users">https://fakestoreapi.com/users</a></li> <li>• Headers: Content-Type: application/json</li> <li>• Corpo da requisição:</li> </ul>	<p><b>Bug Encontrado</b></p>

```
{  
  
  "email": "elisama.qa@teste.com",  
  
  "username": "elisamaQA",  
  
  "password": "teste@123",  
  
  "name": {  
  
    "firstname": "Elisama",  
  
    "lastname": "Gregório"  
  
  }  
  
}
```

## Evidência(s)



## Cenário de Teste 03

## Cenário de Teste: Criar usuário com tipos inválidos

ID	Descrição	
TC018	Este teste verifica se a API é capaz de identificar e rejeitar <b>requisições com tipos de dados incorretos</b> , como por exemplo enviar uma string onde deveria ser um número, ou vice-versa. O objetivo é garantir que o sistema faça a validação correta dos tipos de cada campo.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Funcional</li><li>• Negativo</li><li>• Validação de campos obrigatórios</li><li>• Black-box</li></ul>
Pré-condições		
<ul style="list-style-type: none"><li>• A API deve estar funcionando em <a href="https://fakestoreapi.com">https://fakestoreapi.com</a>.</li><li>• O endpoint <code>/users</code> deve aceitar requisições <b>POST</b>.</li><li>• Ferramenta de teste de API disponível.</li></ul>		
Passos		
<ol style="list-style-type: none"><li>1. Enviar uma requisição <b>POST</b> para o endpoint <code>/users</code> com os dados contendo tipos errados.</li><li>2. Verificar o código de status da resposta.</li><li>3. Validar se a resposta:<ul style="list-style-type: none"><li>• Retorna <b>400 Bad Request</b> ou</li><li>• Retorna uma mensagem clara informando os campos com tipos inválidos</li></ul></li><li>4. Checar se o usuário não foi criado.</li><li>5. Avaliar o tempo de resposta (inferior a 2 segundos).</li></ol>		
Resultado Esperado		Resultado Obtido

<ul style="list-style-type: none"> <li>Status: <b>400 Bad Request</b></li> <li>Mensagem de erro: <pre> {   "error": "Campo 'number' deve ser numérico. Campo 'phone' deve ser string." } </pre> </li> <li>Tempo de resposta: &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>A API <b>não retornou o status apropriado</b> (200), respondendo incorretamente com status <b>400</b>.</li> <li>ID: 11</li> <li>Campos conferem com o envio: Não</li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>Status deve ser <b>400 Bad Request</b>.</li> <li>A resposta deve conter mensagem explicativa sobre os erros de tipo.</li> <li>O usuário não deve ser criado.</li> <li>Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>Nenhum registro de usuário criado no sistema.</li> <li>Ambiente permanece inalterado.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: <b>POST</b></li> <li>Endpoint: <pre> https://fakestoreapi.com/users </pre> </li> <li>Headers: <b>Content-Type:</b> <pre> application/json </pre> </li> <li>Corpo da requisição (exemplo com tipos errados): <pre> { </pre> </li> </ul>	<p><b>Bug Encontrado</b></p>



```
"email": "teste@falso.com",

"username": "usuarioErrado",

"password": "senha123",

"name": {

  "firstname": "Teste",

  "lastname": "QA"

},

"address": {

  "city": "Salvador",

  "street": "Rua Exemplo",

  "number": "cento e vinte", // deveria
ser número

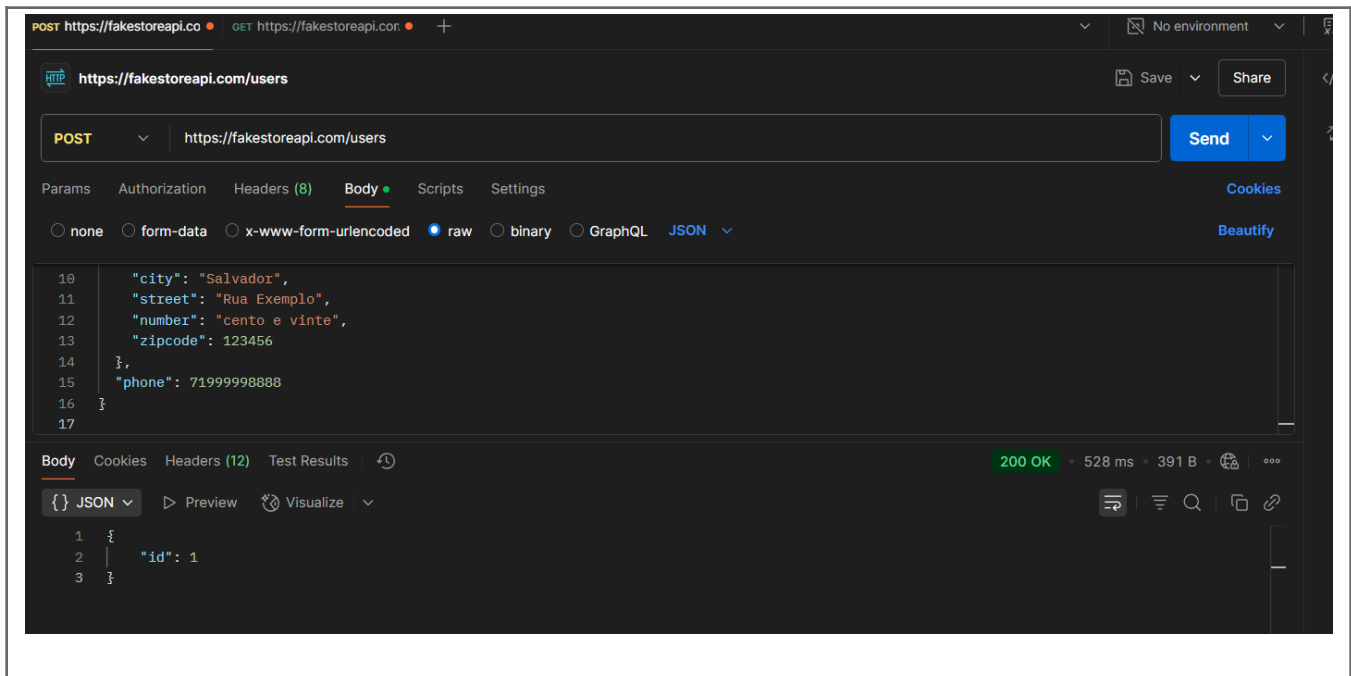
  "zipcode": 123456         // deveria ser
string

},

"phone": 71999998888        // deveria
ser string

}
```

**Evidência(s)**



## Cenário de Teste 04

### Cenário de Teste: Verificar tipos de dados nos campos do usuário

ID	Descrição	
TC019	Este teste valida se os campos retornados pela API ao buscar um usuário possuem <b>tipos de dados consistentes e corretos</b> , conforme o esperado no contrato. O foco é garantir que, por exemplo, <b>email</b> seja uma string, <b>address</b> um objeto, <b>zipcode</b> uma string, etc.	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>Validação</li><li>Funcional</li><li>Contrato de API</li><li>Black-box</li></ul>
Pré-condições		
<ul style="list-style-type: none"><li>Deve haver pelo menos um usuário cadastrado na API (ex: ID 1).</li></ul>		

- A API deve estar funcionando normalmente.
- O endpoint `/users/{id}` deve aceitar requisição `GET`.

### Passos

1. Enviar uma requisição `GET` para `/users/1`.
2. Validar o código de status da resposta.
3. Verificar os tipos dos seguintes campos:
  - `email` → string
  - `username` → string
  - `name` → objeto (`firstname` e `lastname` como string)
  - `address` → objeto
  - `address.zipcode` → string
  - `address.number` → number
  - `phone` → string
4. Validar que **nenhum campo contém tipo inesperado ou inconsistente**.
5. Verificar o tempo de resposta (deve ser inferior a 2 segundos).

### Resultado Esperado

- Status: `200 OK`
- Tipos de dados:
 

```
{
  "email": "string",
  "username": "string",
  "name": {
```

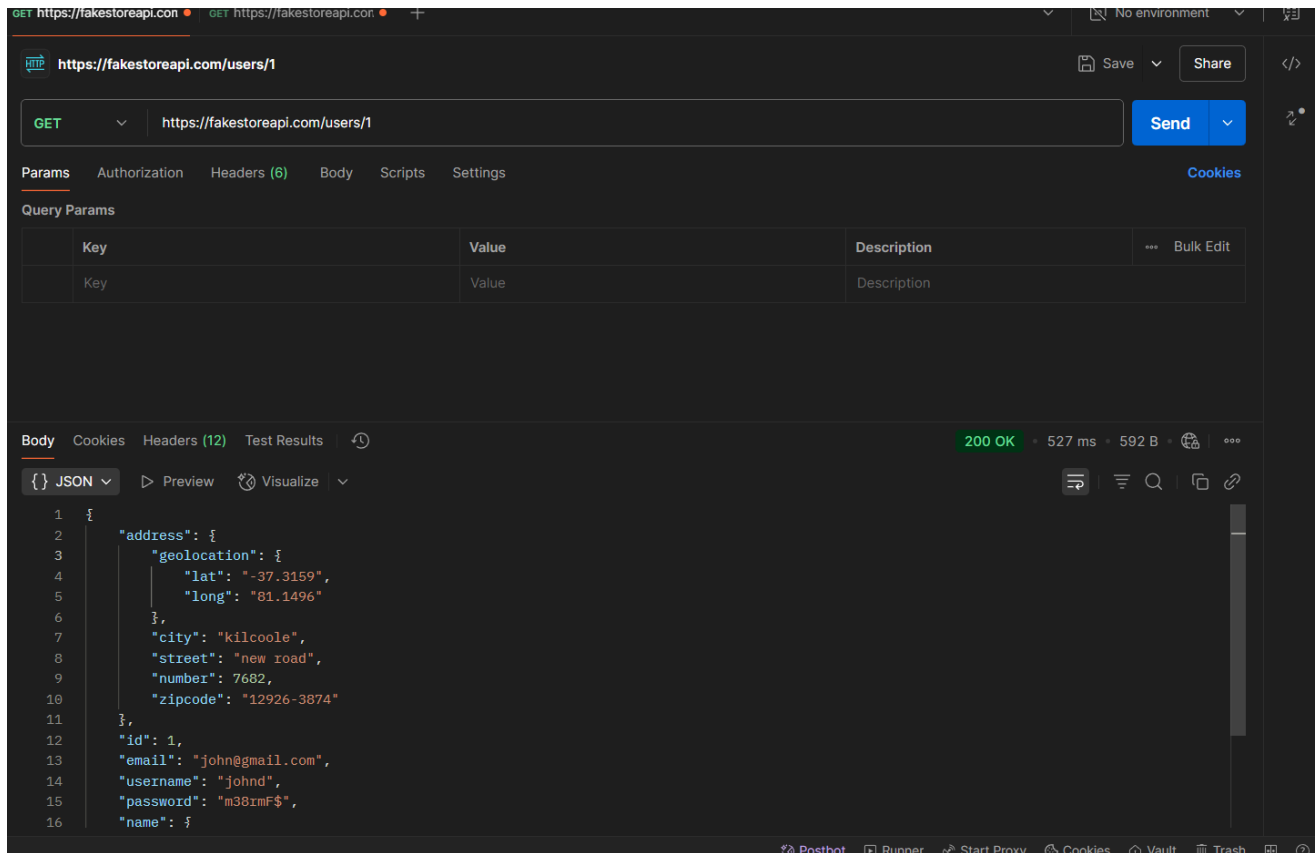
### Resultado Obtido

- Status: 200 OK
- Todos os tipos corretos? Sim
- Tempo: 527 ms

<pre> "firstname": "string",  "lastname": "string"  },  "address": {    "city": "string",    "street": "string",    "number": "number",    "zipcode": "string"  },  "phone": "string"  } </pre> <ul style="list-style-type: none"> <li>• Tempo de resposta: &lt; 2 segundos</li> </ul>	
<b>Critérios de aceitação</b>	<b>Pós-condições</b>
<ul style="list-style-type: none"> <li>• Todos os campos devem possuir o tipo de dado conforme especificado no contrato.</li> <li>• Nenhum campo pode vir com tipo incorreto (ex: número em vez de string).</li> <li>• Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma alteração deve ser feita nos dados da API.</li> </ul>
<b>Dados de teste</b>	<b>Status</b>
<ul style="list-style-type: none"> <li>• Método: <b>GET</b></li> </ul>	<b>Sucesso</b>

- Endpoint:  
`https://fakestoreapi.com/users/1`

## Evidência(s)



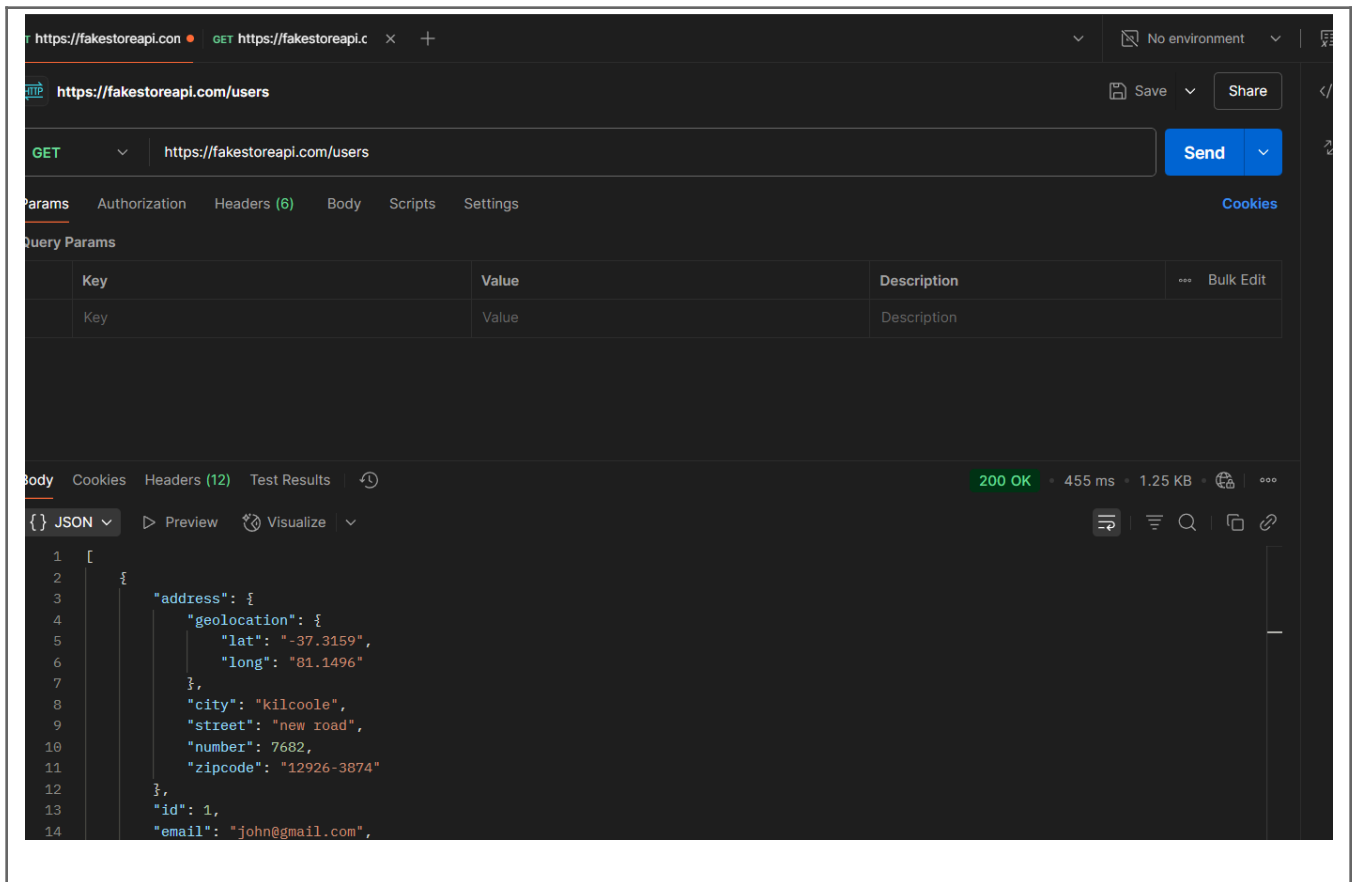
## Cenário de Teste 5

### Cenário de Teste: Performance na listagem de usuários

ID	Descrição
----	-----------

<b>TC020</b>	Este teste mede o tempo de resposta da API ao listar todos os usuários cadastrados, garantindo que o desempenho esteja dentro dos parâmetros aceitáveis definidos como SLA (ex: menos de 2 segundos).
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"> <li>• Performance</li> <li>• Carga leve</li> <li>• Não-funcional</li> <li>• Black-box</li> </ul>
Pré-condições	
<ul style="list-style-type: none"> <li>• A API deve estar ativa e estável.</li> <li>• Deve haver pelo menos alguns usuários cadastrados.</li> <li>• O endpoint <code>/users</code> deve estar funcional e com resposta previsível.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>1. Iniciar um cronômetro ou ferramenta de medição de tempo (ex: Postman, JMeter, k6 ou cronômetro manual).</li> <li>2. Enviar uma requisição <code>GET</code> para <code>/users</code>.</li> <li>3. Medir o tempo entre o envio da requisição e o recebimento da resposta completa.</li> <li>4. Verificar se o tempo de resposta é inferior a 2 segundos.</li> <li>5. Validar se o status da resposta é <code>200 OK</code>.</li> <li>6. Confirmar que o corpo da resposta é um array de usuários.</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>• Status: <code>200 OK</code></li> <li>• Tempo de resposta: <code>&lt; 2000 ms</code></li> <li>• Corpo: Array de usuários válidos</li> </ul>	<ul style="list-style-type: none"> <li>• Status: 200 OK</li> <li>• Tempo: 455 ms</li> <li>• Corpo válido? Sim</li> </ul>

Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• O tempo de resposta <b>deve ser inferior a 2 segundos</b>.</li> <li>• A resposta deve retornar com status <b>200 OK</b>.</li> <li>• O corpo deve conter uma lista de usuários válida (array com campos como <b>id</b>, <b>email</b>, etc).</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma modificação é feita nos dados da API.</li> <li>• O teste não afeta os dados armazenados ou o estado da aplicação.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <b>GET</b></li> <li>• Endpoint: <b><a href="https://fakestoreapi.com/users">https://fakestoreapi.com/users</a></b></li> <li>• Headers: Nenhum necessário</li> <li>• Corpo: Não aplicável</li> </ul>	<p><b>Sucesso</b></p>
Evidência(s)	



## Login (/auth/login)

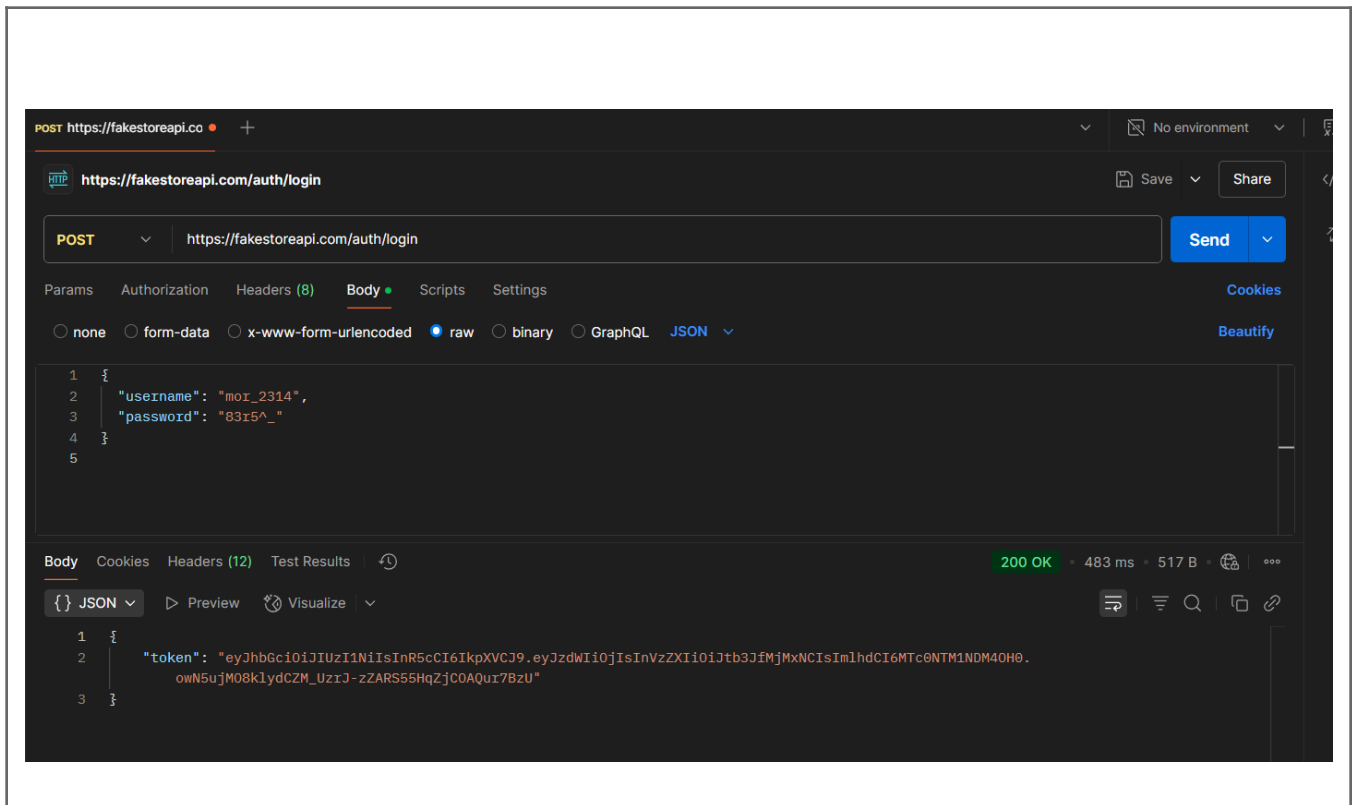
### Cenário de Teste 1

Cenário de Teste: Performance na listagem de usuários		
ID	Descrição	
TC021	Este teste verifica se a API permite a autenticação de um usuário com credenciais válidas, retornando um token de acesso (string) como confirmação do login bem-sucedido.	
Prioridade		Tipo de Teste



Alta	<ul style="list-style-type: none"><li>• Funcional</li><li>• Positivo</li><li>• Autenticação</li><li>• Black-box</li></ul>
Pré-condições	
<ul style="list-style-type: none"><li>• Deve existir um usuário válido cadastrado com <b>username</b> e <b>password</b>.</li><li>• A API de autenticação (<b>/auth/login</b>) deve estar funcionando.</li><li>• O método <b>POST</b> deve ser aceito no endpoint de login.</li></ul>	
Passos	
<ol style="list-style-type: none"><li>1. Enviar requisição <b>POST</b> para <b>/auth/login</b> com o corpo contendo credenciais válidas.</li><li>2. Verificar se o status de resposta é <b>200 OK</b>.</li><li>3. Confirmar que o corpo da resposta contém um campo <b>token</b>.</li><li>4. Validar que o valor de <b>token</b> é uma <b>string não vazia</b>.</li><li>5. (Opcional) Validar o tempo de resposta (&lt; 2 segundos).</li></ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"><li>• Status: <b>200 OK</b></li><li>• Corpo da resposta: <pre>{     "token": "eyJhbGciOiJIUzI1NiIsInR..." }</pre></li><li>• <b>token</b> presente e válido</li><li>• Tempo de resposta: &lt; 2 segundos</li></ul>	<ul style="list-style-type: none"><li>• Status: 200 OK</li><li>• Token retornado? Sim</li><li>• Tipo do token: String</li><li>• Tempo: 483 ms</li></ul>

Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• Status da resposta: <b>200 OK</b>.</li> <li>• A resposta deve conter um campo <b>token</b>.</li> <li>• O <b>token</b> deve ser uma string válida (não nula, não vazia).</li> <li>• Tempo de resposta aceitável (&lt; 2 segundos).</li> </ul>	<ul style="list-style-type: none"> <li>• O token obtido pode ser usado para chamadas autenticadas (dependendo do suporte da API).</li> <li>• Nenhuma modificação é feita nos dados do usuário.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <b>POST</b></li> <li>• Endpoint: <b>https://fakestoreapi.com/auth/login</b></li> <li>• Headers: <b>Content-Type:</b> <b>application/json</b></li> <li>• Corpo da requisição: <pre> {   "username": "mor_2314",   "password": "83r5^_" } </pre> </li> </ul>	<p><b>Sucesso</b></p>
Evidência(s)	

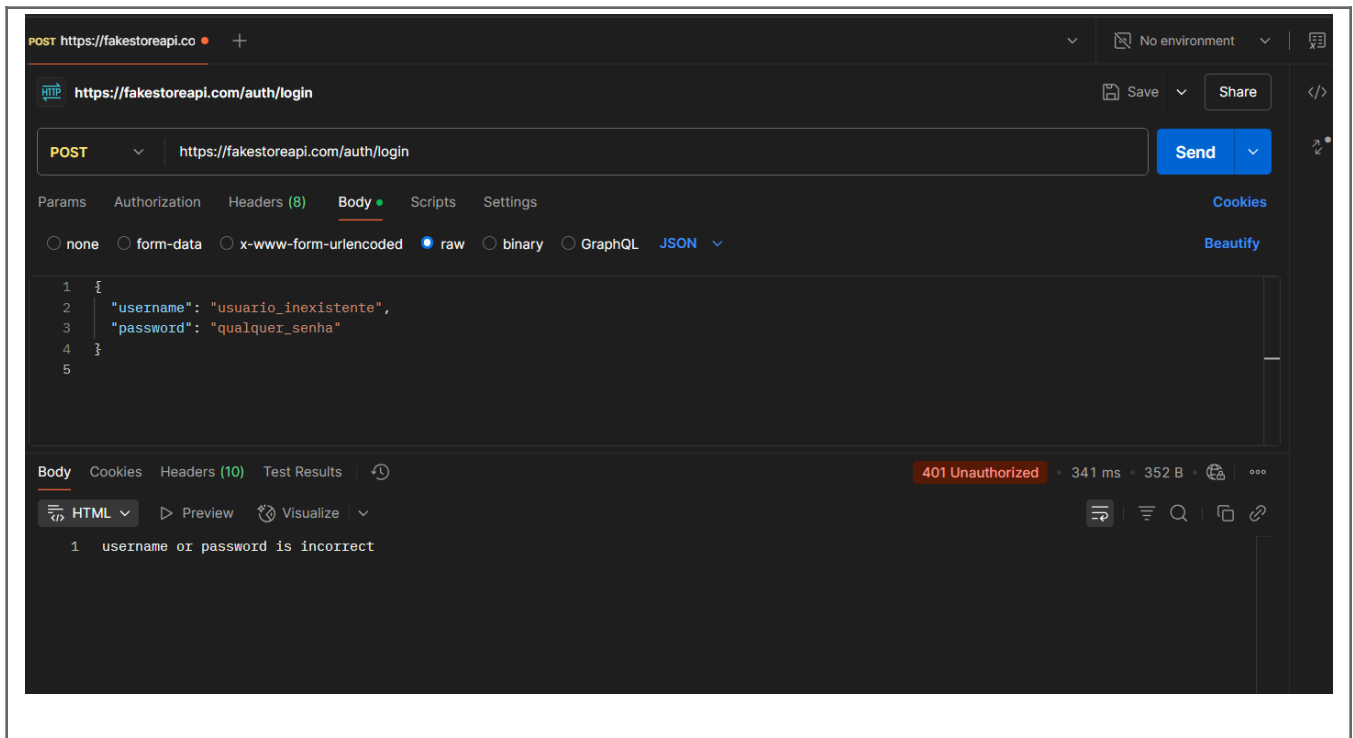


## Cenário de Teste 2

Cenário de Teste: Login com usuário inválido		
ID	Descrição	
TC022	Este teste valida o comportamento da API ao tentar autenticar um usuário <b>com nome de usuário inexistente</b> , garantindo que o sistema retorne um erro apropriado e <b>não forneça token de acesso</b> .	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>• Funcional</li><li>• Negativo</li><li>• Autenticação</li><li>• Segurança</li><li>• Black-box</li></ul>

Pré-condições	
<ul style="list-style-type: none"> <li>A API de login deve estar operacional.</li> <li>O usuário informado no teste <b>não pode existir</b> no sistema.</li> <li>O endpoint <code>/auth/login</code> deve aceitar requisições <code>POST</code>.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>Enviar uma requisição <code>POST</code> para <code>/auth/login</code> com credenciais inválidas (usuário inexistente).</li> <li>Verificar o código de status da resposta.</li> <li>Verificar se a resposta <b>não contém</b> um token.</li> <li>Validar se a <b>mensagem de erro</b> retornada é clara e coerente (ex: "Usuário não encontrado" ou "Credenciais inválidas").</li> <li>(Opcional) Validar tempo de resposta (&lt; 2 segundos).</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>Status: <code>401 Unauthorized</code> ou <code>403 Forbidden</code></li> <li>Corpo da resposta: <pre>{   "message": "User not found" // ou algo similar }</pre> </li> <li>Token ausente</li> <li>Tempo de resposta: &lt; 2 segundos</li> </ul>	<ul style="list-style-type: none"> <li>Status: 401 Unauthorized</li> <li>Token presente? Não</li> <li>Mensagem clara? Sim</li> <li>Tempo: 341 ms</li> </ul>

Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A API deve retornar status <b>401 Unauthorized</b> ou <b>403 Forbidden</b>.</li> <li>• A resposta <b>não pode conter um token</b>.</li> <li>• A mensagem de erro deve indicar falha de autenticação de forma clara.</li> <li>• Tempo de resposta &lt; 2 segundos.</li> </ul>	<ul style="list-style-type: none"> <li>• Nenhuma modificação nos dados da API.</li> <li>• Nenhuma sessão deve ser iniciada.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <b>POST</b></li> <li>• Endpoint: <b>https://fakestoreapi.com/auth/login</b></li> <li>• Headers: <b>Content-Type: application/json</b></li> <li>• Corpo da requisição: <pre>{   "username": "usuario_inexistente",   "password": "qualquer_senha" }</pre> </li> </ul>	<p><b>Sucesso</b></p>
Evidência(s)	



### Cenário de Teste 3

Cenário de Teste: Verificar formato do token retornado		
ID	Descrição	
TC023	Este teste valida se, ao realizar login com credenciais válidas, a API retorna um <b>token de autenticação</b> com formato correto, verificando se é uma <b>string</b> que segue o padrão JWT (3 partes separadas por pontos).	
Prioridade		Tipo de Teste
Alta		<ul style="list-style-type: none"><li>Validação</li><li>Funcional</li><li>Autenticação</li><li>Segurança</li><li>Black-box</li></ul>
Pré-condições		

- O usuário utilizado no login deve ser válido.
- A API `/auth/login` deve estar funcional.
- O método `POST` deve estar habilitado para autenticação.

### Passos

1. Enviar uma requisição `POST` para `/auth/login` com credenciais válidas.
2. Capturar o corpo da resposta.
3. Verificar se há um campo `token` na resposta.
4. Validar se o valor do token:
  - É uma **string**
  - Contém **3 partes** separadas por `.` (ex: `xxxxx.yyyyy.zzzzz`)
  - Cada parte é uma string codificada em base64 (opcional)
5. Verificar o status da resposta.

### Resultado Esperado

- Status: `200 OK`
- Exemplo de resposta:
 

```
{
  "token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im1vcl8yMzE0Ln0.a
  bc123def456ghi789"
}
```
- Token presente
- Token é string

### Resultado Obtido

- Status: 200
- Token: Sim
- Formato JWT: Sim
- Tempo de resposta: 483ms

<ul style="list-style-type: none"> <li>• Contém 3 partes separadas por .</li> </ul>	
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>• A resposta deve ter <b>status 200 OK</b>.</li> <li>• O campo <b>token</b> deve estar presente.</li> <li>• O valor do token deve: <ul style="list-style-type: none"> <li>Ser do tipo <b>string</b></li> <li>Conter exatamente <b>2 pontos (.)</b>, formando 3 partes</li> <li>Ter comprimento compatível com um token JWT (geralmente &gt;100 caracteres)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Sessão autenticada disponível (caso o token seja utilizado).</li> <li>• Nenhuma alteração nos dados do sistema.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>• Método: <b>POST</b></li> <li>• Endpoint: <pre>https://fakestoreapi.com/auth/login</pre> </li> <li>• Headers: <b>Content-Type:</b> <pre>application/json</pre> </li> <li>• Corpo da requisição: <pre>{   "username": "mor_2314",   "password": "83r5^_" }</pre> </li> </ul>	<p><b>Sucesso</b></p>



The screenshot displays a REST client interface with a dark theme. At the top, a navigation bar shows the current request: **POST** <https://fakestoreapi.com/auth/login>. Below this, the request details are shown, including the method **POST** and the URL <https://fakestoreapi.com/auth/login>. The request body is set to **raw** and contains a JSON object: 

```
{  "username": "mor_2314",  "password": "83i5^_"}
```

. The response status is **200 OK**, with a response time of 483 ms and a response size of 517 B. The response body is shown in **JSON** format: 

```
{  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWI0IjIsInVzZXI6Ijtb3JfMjMNCiIsImh0bCI6MTc0NTM1NDM0M0H0.owN5uJm08klydCZM_UzrJ-zZARS55HqZjC0AQu7BzU"}
```

## Cenário de Teste 4

Cenário de Teste: Performance no login	
ID	Descrição
TC024	Este teste avalia o <b>tempo de resposta</b> da API ao realizar login com credenciais válidas. O objetivo é verificar se o endpoint <code>/auth/login</code> responde em um tempo aceitável, conforme o SLA definido (inferior a <b>1 segundo</b> ).
Prioridade	Tipo de Teste
Alta	<ul style="list-style-type: none"><li>• Performance</li><li>• Funcional</li><li>• Autenticação</li><li>• Black-box</li></ul>

Pré-condições	
<ul style="list-style-type: none"> <li>API <code>/auth/login</code> deve estar acessível.</li> <li>O usuário de teste deve estar previamente cadastrado com credenciais válidas.</li> <li>Ferramenta de medição de tempo (Postman.) deve estar configurada.</li> </ul>	
Passos	
<ol style="list-style-type: none"> <li>Enviar uma requisição <code>POST</code> para <code>/auth/login</code> com as credenciais acima.</li> <li>Medir o tempo total de resposta da requisição.</li> <li>Verificar o status retornado.</li> <li>Validar se o tempo de resposta está <b>abaixo de 1000ms (1 segundo)</b>.</li> </ol>	
Resultado Esperado	Resultado Obtido
<ul style="list-style-type: none"> <li>Status: <code>200 OK</code></li> <li>Tempo de resposta: <code>&lt; 1 segundo</code></li> </ul>	<ul style="list-style-type: none"> <li>Status: 200</li> <li>Token recebido: Sim</li> <li>Tempo de resposta: <code>900ms</code></li> </ul>
Critérios de aceitação	Pós-condições
<ul style="list-style-type: none"> <li>A resposta deve ter status <b>200 OK</b>.</li> <li>O tempo de resposta deve ser <b>&lt; 1 segundo</b> (idealmente &lt; 800ms).</li> <li>O corpo da resposta deve conter um <b>token válido</b>.</li> </ul>	<ul style="list-style-type: none"> <li>Sessão iniciada com token válido (caso o login seja bem-sucedido).</li> <li>Nenhum outro efeito colateral no sistema.</li> </ul>
Dados de teste	Status
<ul style="list-style-type: none"> <li>Método: <code>POST</code></li> </ul>	<b>Sucesso</b>

- Endpoint:  
`https://fakestoreapi.com/auth/login`
- Headers: `Content-Type: application/json`
- Corpo da requisição:  

```
{  
  "username": "mor_2314",  
  "password": "83r5^_"  
}
```

### Evidência(s)

