

# **Mapeamento e teste dos requests (API PetStore)**

## **Squad 3 - CapsLock**

Andressa Mota

Guilherme Taveira

Marcelo Ferreira

Maycon Douglas

Willian

## **1. Objetivo**

O propósito deste relatório é verificar se a API PetStore atende aos contratos e cenários definidos na documentação Swagger, garantindo a robustez, conformidade e estabilidade dos endpoints principais. Focamos em validar operações CRUD, fluxos de autenticação e manipulação de dados, identificando falhas de conformidade e comportamentos inesperados.

## **2. Escopo dos Testes**

- Cadastro, consulta, atualização e remoção de Pets
- Geração e consulta de pedidos (Orders)
- Cadastro, autenticação e gerenciamento de usuários
- Validação de autenticação por token (Auth)
- Verificação de códigos de status HTTP e conformidade de esquemas JSON

## **3. Ferramentas Utilizadas**

- Swagger UI 3.x : Para consultas interativas e exportação do JSON/YAML da API para referência.
- Postman v10.9.0 : Para criação de Collection com todos os endpoints e variáveis de ambiente (baseUrl, apiKey).

## **4. Metodologia de Teste**

Estratégia Utilizada: Testes Baseados em Especificação (Specification Testing)

Justificativa: permite validar rigorosamente cada campo e comportamento esperado, detectando inconsistências entre implementação e documentação.

Heurísticas Aplicadas:

- Verificação de códigos de erro
- Consistência de esquema JSON (tipos, obrigatoriedade)
- Tratamento de dados inválidos e limites de input
- Autenticação: validade e expiração de tokens

## **5. Técnica de Execução**

Execução Orientada por Collection no Postman:

1. Configuração dos ambientes (dev, staging) com variáveis globais.
2. Importação da Collection gerada a partir do Swagger.
3. Execução sequencial dos requests em modo Runner, com gravação de logs.
4. Uso de scripts "Tests" para assertivas de status code.

## 6. Mapeamento de Endpoints e Casos de Teste

- ENDPOINTS /PET

Endpoint	Método	Descrição	Retorno do Teste	Resultado Esperado
/pet	POST	Adiciona novo pet	<pre>{ "id": 101, "name": "Rex", "status": "available" }</pre>	200 OK, JSON com dados gravados Deveria retornar 201 no Status Code
/pet	PUT	Atualiza dados de pet	<pre>{ "id": 101, "name": "Rexy", "status": "sold" }</pre>	200 OK, JSON atualizado
/pet/{petId}	DELETE	Remove pet	Pet deleted	200 OK, mensagem de sucesso
/pet/findByStatus	GET	Consulta todos Pets por status	<pre>{ "id": 27555, "category": { "id": 1, "name": "Dogs" }, "name": "Fluffy", "photoUrls": [ "https://example.com/photo.jpg" ] }</pre>	200 OK, JSON retornando consulta todos os Pets por status
/pet/findByTags	GET	Consulta Pets por tags	<pre>"tags": [ { "id": 1, "name": "cute" } ], "status": "sold" },</pre>	200 ok, JSON retornando consulta por tags

- **ENDPOINTS /STORE**

Endpoint	Método	Descrição	Retorno do Teste	Resultado Esperado
/store/inventory	GET	Retorna contagem de pets por status	—	200 OK, JSON: <pre>{   "available" : 5, ... }</pre>
/store/order	POST	Cria novo pedido	<pre>{ "id": 201,   "petId": 101,   "quantity": 1, "status": "placed" }</pre>	200 OK, JSON do pedido criado. <b>Deveria retornar 201 no Status Code</b>
/store/order/{orderId}	GET	Consulta pedido por ID	—	200 OK, JSON com o pedido
/store/order/{orderId}	DELETE	Cancela pedido	—	200 OK, confirmação de exclusão

- ENDPOINTS /USER

Endpoint	Método	Descrição	Retorno do Teste	Resultado Esperado
/user	POST	Cria usuário	<pre>{ "id": 301, "username": "andressa", "firstName": "Andressa", "email": "a@example.com" }</pre>	200 OK, JSON do usuário criado Deveria retornar 201 no Status Code
/user/createWithArray	POST	Cria múltiplos usuários (array)	<pre>[ { "id": 302, "username": "maria" }, { "id": 303, "username": "joao" } ]</pre>	200 OK, lista de usuários criados
/user/createWithList	POST	Cria múltiplos usuários (lista)	<i>mesma estrutura de array acima</i>	200 OK, lista de usuários criados
/user/login	GET	Autentica usuário	<pre>Logged in user session: 1748902251825163146</pre>	200 OK, token em X-API-KEY
/user/logout	GET	Encerra sessão do usuário	—	200 OK, mensagem de logout

/user/{username}	GET	Consulta dados de um usuário por username	—	200 OK, JSON com dados do usuário
/user/{username}	PUT	Atualiza dados de um usuário	{ "id": 301, "username": "andressa", "firstName": "Andre", "email": "a@novo.com" }	200 OK, JSON com dados atualizados
/user/{username}	DELETE	Remove usuário	—	200 OK, confirmação de exclusão

## 7. Registro dos resultados

### GET user/login

<b>Data e hora</b>	07/08/2025 às 15:30
<b>testador</b>	Squad CapsLock
<b>Funcionalidade explorada</b>	GET /user/login
<b>Observações</b>	O endpoint /user/login aceita qualquer valor para username e password
<b>Melhorias sugeridas</b>	Se faz necessário a criação de diferentes formas de autenticação de usuário e senha, como por exemplo o uso de api-key ou token.

The screenshot displays a web application interface on the left and a Postman API client window on the right.

**Web Application Interface:**

- A large heading "Evidências" is visible at the top.

**Postman API Client Window:**


- Tab:** PetStore / GET Login
- Method:** GET
- URL:** {{url}} /user/login?username=sdfsdfsd sdf&password=234234
- Params:** Authorization Headers (7) Body Scripts Settings
- Query Params:**

Key	Value	Description
username	sdfsdfsd sdf	
password	234234	
- Body:** Cookies Headers (11) Test Results (1/2)
- Status:** 200 OK • 748 ms • 481 B
- Response Format:** JSON
- Preview:**

```
1 Logged in user session: 6660636254409146896
```

Data e hora	07/08/2025 às 15:42
testador	Squad CapsLock
Funcionalidade explorada	GET /user/{username}
Observações	/user/{username} está com instabilidade no servidor e nos primeiros testes retornou com erro 500. Hoje (08/08/2025) está funcionando como mostra a imagem capturada abaixo.
Melhorias sugeridas	Se faz necessário conferir o motivo da instabilidade do servidor de api.

# Evidências

 PetStore / GET Username

GET

▼

{{url}}/user/test

Params

Authorization ●

Headers (8)

Body

Scripts ●

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON ▼

1

Ctrl+Alt+P for Postbot

Body

Cookies

Headers (9)

Test Results (0/1)

🔄

500 Internal Server Error

• 177 ms • 500

{{}} JSON ▼

▶ Preview

🔗 Visualize ▼

1

{

2


"code": 500,

3

"message": "There was an error processing your request. It has been logged (ID: 35eb47ab0370f3be)"

4

}

 PetStore / GET Username

GET

▼

{{url}}/user/test

Params

Authorization ●

Headers (8)

Body

Scripts ●

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (9)

Test Results (1/1)

🔄

200 OK

•

{{}} JSON ▼

▶ Preview

🔗 Visualize ▼

1

{

2

"id": 10,

3

"username": "test",

4

"firstName": "John",

5

"lastName": "James",

6

"email": "john@email.com",

7

"password": "12345",

8

"phone": "12345",

9

"userStatus": 1

10

}



POST /user

Data e hora	07/08/2025 às 15:58
testador	Squad CapsLock
Funcionalidade explorada	Post /user
Observações	Post /user não está cadastrando usuário e retornando erro 500.
Melhorias sugeridas	Se faz necessário reparar a função de cadastro de usuário
Evidências	<div><div><div><div><div><div><span></span></div><div>PetStore / POST cadastro de usuario</div></div></div><div><div>POST</div><div>{{url}} /user</div></div></div><div><div>Params</div><div>Authorization</div><div>Headers (9)</div><div>Body</div><div>Scripts</div><div>Settings</div></div><div><div><div><div>none</div></div><div><div>form-data</div></div><div><div>x-www-form-urlencoded</div></div><div><div>raw</div></div><div><div>binary</div></div><div><div>GraphQL</div></div><div><div>JSON</div></div></div></div><div><div><div>1</div><div>{</div></div><div><div>2</div><div>  "id": 10,</div></div><div><div>3</div><div>  "username": "Michael",</div></div><div><div>4</div><div>  "firstName": "John",</div></div><div><div>5</div><div>  "lastName": "James",</div></div><div><div>6</div><div>  "email": "mail@email.com",</div></div><div><div>7</div><div>  "password": "12345",</div></div><div><div>8</div><div>  "phone": "12345",</div></div><div><div>9</div><div>  "userStatus": 1</div></div><div><div>10</div><div>}</div></div></div><div><div>Body</div><div>Cookies</div><div>Headers (9)</div><div>Test Results</div></div><div><div>500 Internal Server Error</div><div>903 ms</div><div>501 B</div></div><div><div><div>{}</div><div>JSON</div></div><div>Preview</div><div>Visualize</div></div><div><div><div>1</div><div>{</div></div><div><div>2</div><div>  "code": 500,</div></div><div><div>3</div><div>  "message": "There was an error processing your request. It has been logged (ID: f815d3d0d860159a)"</div></div><div><div>4</div><div>}</div></div></div></div></div>

8. Resultados e Observações

- Todos os endpoints CRUD de Pet e Store responderam conforme o esquema descrito, sem desvios de tipo ou dados faltantes.
- Alguns endpoints de User apresentam erros ou comportamentos não esperados.

## 9. Conclusão

Com a identificação de alguns bugs nos testes dos endpoints, temos que, embora a base funcional esteja em funcionamento, há pontos críticos que impedem a estabilidade e a confiabilidade da API em um ambiente de produção.

## 10. Collection Postman

