Relatório de Testes – Swagger Petstore API - Karen

Introdução

> **Objetivo:** Este documento apresenta as execuções dos testes conduzidos na seção **User** da API Swagger Petstore, utilizando a ferramenta Postman. O foco dos testes está nas operações de **criação de usuário individual, cadastro em massa de usuários e login de usuário,** avaliando o comportamento das responses, sua conformidade com a documentação e identificando possíveis falhas ou oportunidades de melhoria.
>
> *Durante a análise:*
>
> As três seções testadas — `POST /user`, `POST /user/createWithList` e `GET /user/login` — retornaram **status 200 OK** em todas as execuções com dados válidos, conforme o esperado. Porém logo em seguida foram encontradas discrepâncias na execução.
>
> Adicionalmente, foram realizados testes complementares para avaliar a robustez da API frente a cenários negativos, como entradas inválidas ou omissões de parâmetros. Esses testes revelaram respostas inesperadas, indicando a ausência de validações adequadas e a necessidade de aprimoramento no tratamento de erros.

Escopo

API testada	Módulo	Ferramenta	Tipo de Teste
Swagger UI	User	Postman	Teste de API
Tester	**Duração**	**Método de Registro**	**S.O**
Karen K.	30 minutos	Notas manuais + Captura de Tela	Windows 10

Detalhamento de Execução e Respostas Encontradas

Principais Testes Realizados

POST /user – Criação de Usuário

POST /user/createWithList – Criação em Massa de Usuários

GET /user/login – Login de Usuário

⚠️ Criação de Usuário – POST /user

Esse endpoint é responsável por criar um único usuário com base no corpo da requisição em formato JSON. A documentação indica que, ao ser bem-sucedido, retorna status **200 OK** com os mesmos dados enviados. O schema de exemplo inclui campos como `id`, `username`, `firstName`, `lastName`, `email`, `password`, `phone` e `userStatus`.

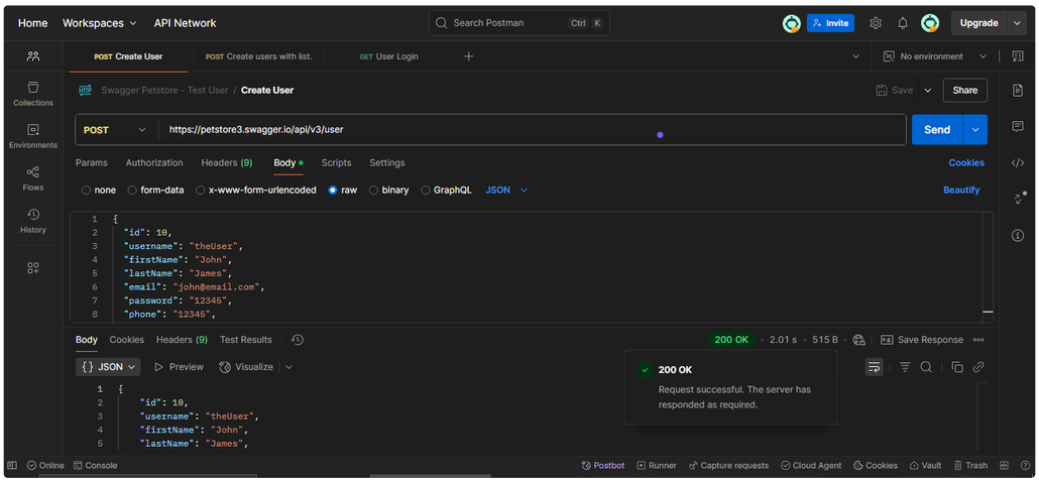
Comportamento observado:
Em primeira execução observou-se que a requisição foi processada com sucesso, retornando **status 200 OK**, conforme esperado. Porém em segunda execução posterior, foi retornado status **500 Internal Server Error.**

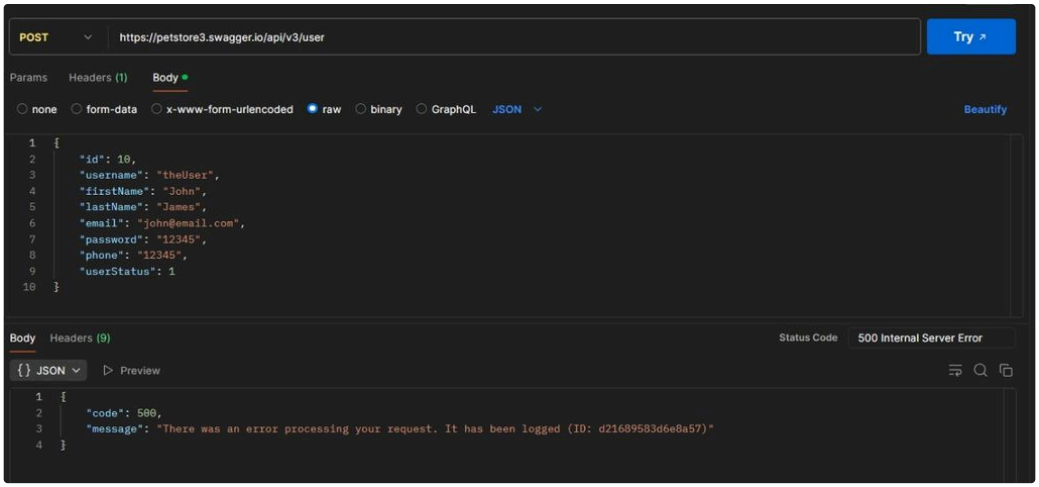
Descrição	Endpoint	Request Body
Criar um usuário individual enviando um objeto JSON.	https://petstore3.swagger.io/api/v3/user	`{ "id": 10, "username": "theUser", "firstName": "John", "lastName": "James", "email": "john@email.com", "password": "12345", "phone": "12345", "userStatus": 1 }`

Observado em Teste:

- **1 Execução:** status **200 OK**, conforme esperado.
- **2 Execução** : Status **500 Internal Server Error.**

Evidências:





⚠️ POST /user/createWithList - Criação em Massa de Usuários

Este endpoint permite criar uma **lista de usuários** a partir de um array JSON enviado no corpo da requisição. A documentação mostra um schema claro com os mesmos campos do endpoint individual. Espera-se o retorno de status **200 OK** com os dados criados.

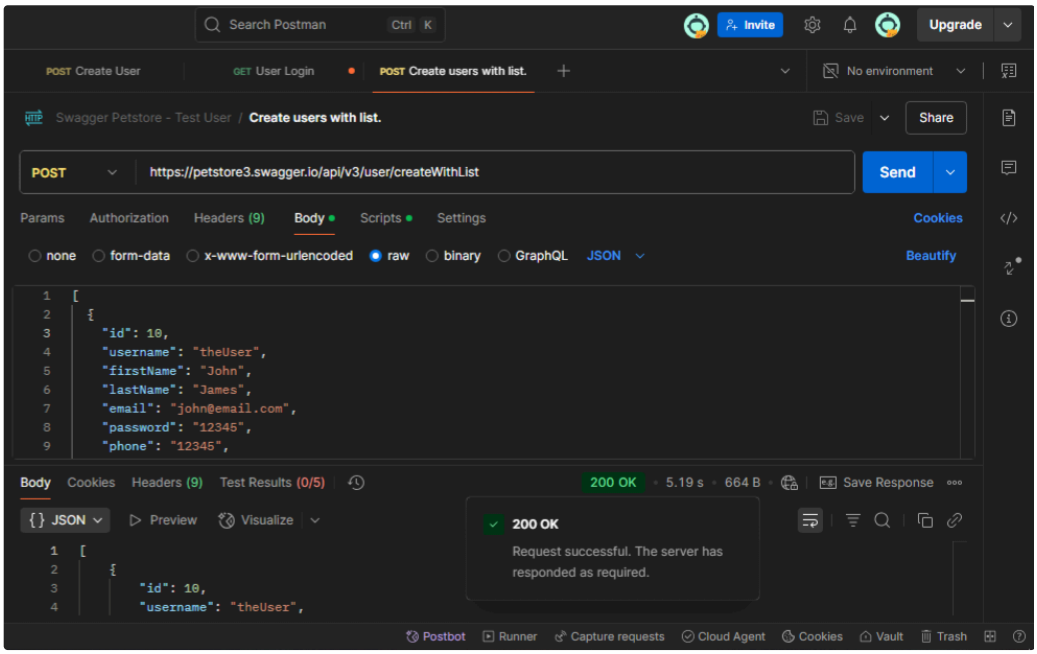
Comportamento observado:
Em primeira execução observou-se que a requisição foi processada com sucesso, retornando **status 200 OK**, conforme esperado. Porém em segunda execução posterior, foi retornado status **405 Method Not Allowed.**

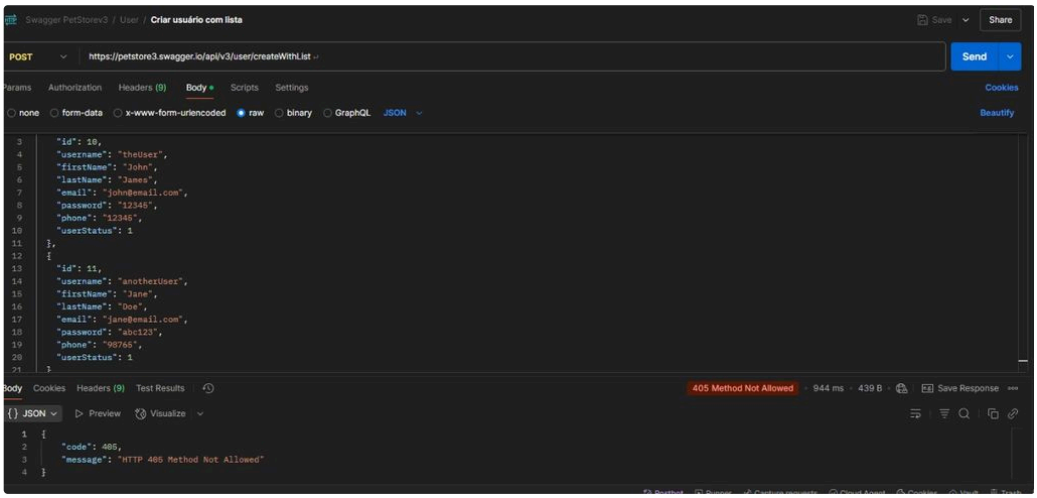
Descrição	Endpoint	Request Body
Cria vários usuários simultaneamente usando um array JSON.	https://petstore3.swagger.io/api/v3/user/createWithList	`[{ "id": 10, "username": "theUser", "firstName": "John", "lastName": "James", "email": "john@email.com", "password": "12345", "phone": "12345", "userStatus": 1 }, { "id": 11, "username": "anotherUser", "firstName": "Jane", "lastName": "Doe", "email": "jane@email.com", "password": "abc123", "phone": "98765", "userStatus": 1 }]`

Observado em Teste:

- **1 Execução:** status **200 OK**, conforme esperado.
- **2 Execução :** Status **405 Method Not Allowed.**

Evidência:





GET /user/login - Login de Usuário

Este endpoint permite autenticar um usuário via parâmetros de consulta (`username` e `password`). O retorno esperado em caso de sucesso é **200 OK**.Em caso de erro, retorna **400 Invalid username/password supplied** ou um erro padrão inesperado.

Comportamento observado:
O login foi testado com credenciais válidas, e o resultado foi **200 OK**, conforme documentado.

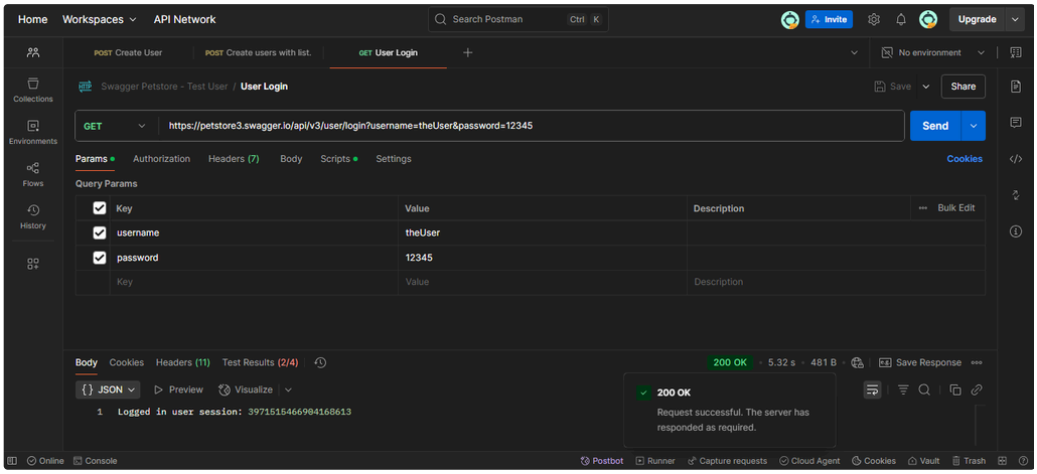
Análise:
O endpoint funcionou como esperado, demonstrando conformidade com a documentação.

Descrição	Endpoint
Realizar login com nome de usuário e senha.	https://petstore3.swagger.io/api/v3/user/login?username=theUser&password=12345

Observado em Teste:

- **Execução:** status **200 OK**, conforme esperado.

Evidência:



Busca de Aprofundamento – (Seção `user`)

POST /user – Criação de usuário individual

Teste	Descrição	Resultado Esperado	Resultado Obtido	Status do Teste
1.1	Enviar payload vazio `{}`	400 Bad Request	500 Internal Server Error	❌ Erro grave: Servidor não tratou input inválido corretamente.

1.2	Enviar tipos incorretos (ex: `"id": "abc"` em vez de número)	400 Bad Request (esperado, mas não documentado)	400 Bad Request	✅ Comportamento correto, documentação deve ser atualizada
1.3	Remover campo obrigatório `username`	400 ou erro default	400 Bad Request	✅ Comportamento correto, documentação deve ser atualizada

---

POST /user/createWithList – Criação em lote de usuários

Teste	Descrição	Resultado Esperado	Resultado Obtido	Status do Teste
2.1	Enviar array vazio `[]`	400 ou erro default	400 Bad Request	✅ Comportamento correto, documentação deve ser atualizada
2.2	Enviar lista com objetos incompletos (ex: sem `username`)	400 ou erro default	400 Bad Request	✅ Comportamento correto, documentação deve ser atualizada

---

GET /user/login – Autenticação de usuário

Teste	Descrição	Resultado Esperado	Resultado Obtido	Observações
3.1	Enviar username e password inexistentes	400 Invalid username/password	❌ 200 OK	Inconsistência: Login aceito mesmo com credenciais inválidas. Falta de validação real.
3.2	Omitir password	400 ou erro default	❌ 200 OK	Comportamento incorreto. Autenticação sem senha não

				deveria ser aceita.
3.3	Omitir username	400 ou erro default	❌ 200 OK	Login sem username retornou sucesso, contradizendo a documentação.
3.4	Enviar strings vazias `?username=&password=`	400 ou erro default	❌ 200 OK	Backend não trata campos vazios como inválidos.
3.5	Enviar tipos inválidos (números)	400 ou erro default	❌ 200 OK	Tipos incorretos (inteiros em vez de string) aceitos. Sinal de validação fraca ou ausente.

Observações Gerais

- **Validação Fraca no Login**: O endpoint `/user/login` apresenta falhas críticas de validação. Credenciais inválidas são aceitas, o que compromete a integridade da autenticação.
- **Sugestão de Melhoria**: Implementar validação robusta nos campos obrigatórios e autenticação real no login. Mensagens de erro mais específicas também ajudariam no diagnóstico.

Embora a documentação da API indique apenas os status `200 OK` e `default: Unexpected error`, nas seções **POST /user** e **POST /user/createWithList,** os testes negativos retornaram corretamente o status `400 Bad Request` ao receberem dados inválidos. **Sugere-se atualizar a documentação da API para incluir `400` como uma resposta esperada** em casos de erro de validação de input.

Durante a execução dos testes, foi identificada uma **inconsistência intermitente** na resposta da API. O endpoint retornou **status 200 (OK)** inicialmente, seguido por **respostas 500 (Internal Server Error)** em algumas requisições subsequentes, e posteriormente voltou a responder com **status 200**. A causa dessa oscilação não foi identificada no momento do teste e pode estar relacionada a fatores temporários no ambiente do servidor ou instabilidades internas na API.

Conclusão Final

Os testes realizados na seção `user` da API Swagger Petstore evidenciam a necessidade de um aprofundamento na **validação de entradas e consistência nas respostas**. Embora alguns endpoints, como `POST /user` e `POST /user/createWithList`, apresentem respostas condizentes com a documentação em cenários negativos, **a presença de erro 500 com mensagens genéricas** indica ausência de tratamento refinado para exceções esperadas.

O maior ponto de atenção recai sobre o endpoint `GET /user/login`, que **não valida corretamente as credenciais fornecidas**, permitindo autenticação com campos inexistentes, vazios ou até omitidos — comportamento que compromete a lógica de segurança da aplicação.

Dessa forma, recomenda-se:

- Reforçar validações no backend, especialmente nos campos obrigatórios.
- Implementar **mensagens de erro mais informativas** e específicas.
- Garantir que os códigos de resposta estejam alinhados com o comportamento real da API.

Este conjunto de testes contribui para destacar não só falhas técnicas, mas também **possíveis lacunas na robustez da API**, servindo como base para ajustes futuros e melhorias contínuas na qualidade do serviço.