Semana 10 - Report de Issues (Petstore API)

SQUAD 1: Andrey Felipe de Lima; Daniela Siana Pabis; Eduardo Neves de Souza; João Pedro de Jesus Perin.

ISSUE 01:

Título: API permite criação de pet sem campos obrigatórios (name e photoUrls) e retorna 200 OK

Descrição:

A API permite a criação de um pet via endpoint POST /pet exigindo apenas a presença do body da requisição, sem validar os campos obrigatórios necessários para a persistência correta do recurso. Ao enviar um body vazio ou contendo campos obrigatórios ausentes ou vazios, como ‘name' e ‘photoUrls’, a API processa a requisição normalmente e retorna '200 ok’.

Esse comportamento é incorreto, pois permite a criação de recursos sem informações mínimas, resultando em dados inconsistentes e sem valor funcional no sistema.

Reprodução:

1. Executar uma requisição POST /pet;
2. Enviar um body vazio ou com campos obrigatórios ausentes;
3. Enviar a requisição;
4. Observar que a API retorna '200 OK' e cria o recurso mesmo sem dados válidos.

Evidências:

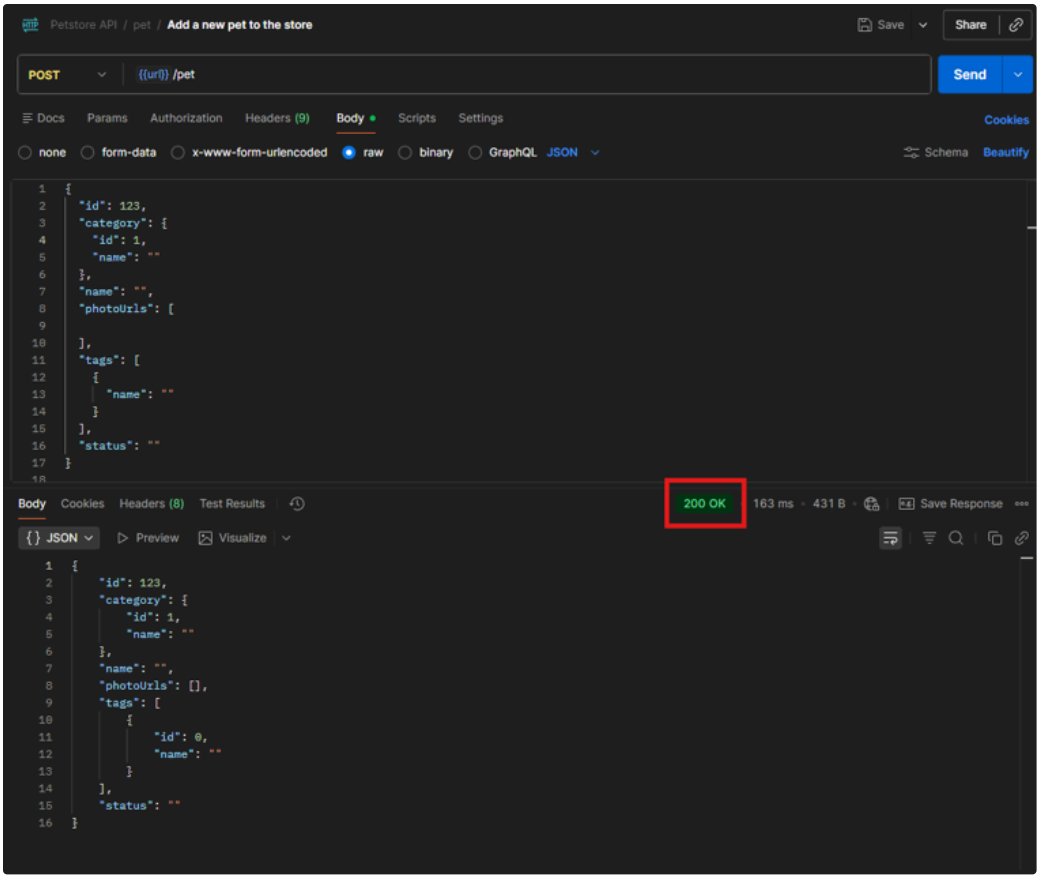

Figura 1 - Retorno de status code incorreto (caso 1)

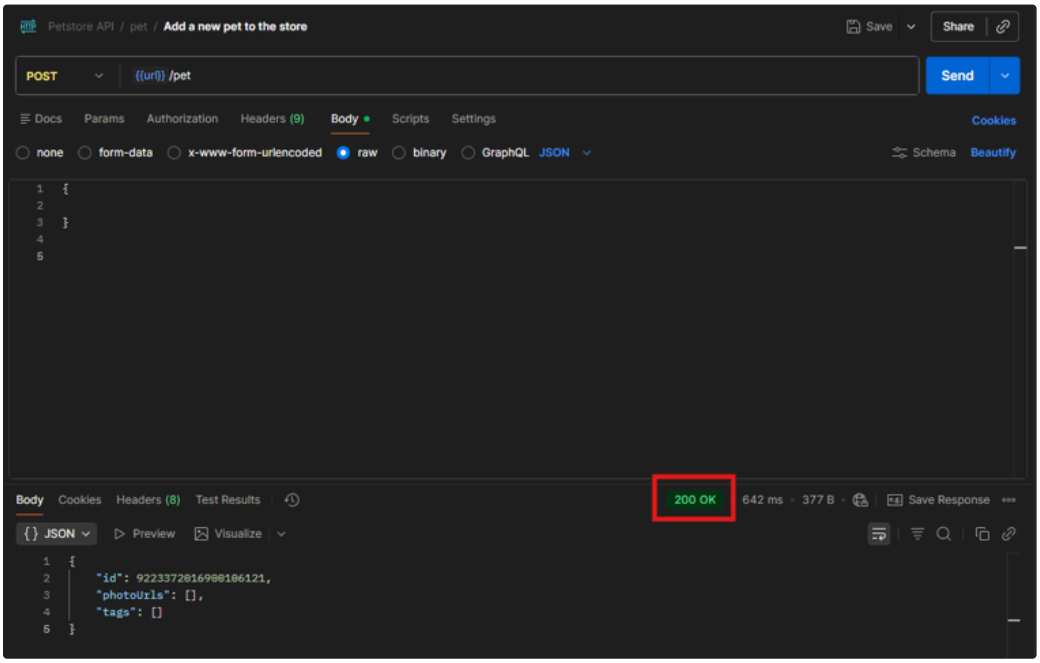

Figura 2 - Retorno de status code incorreto (caso 2)

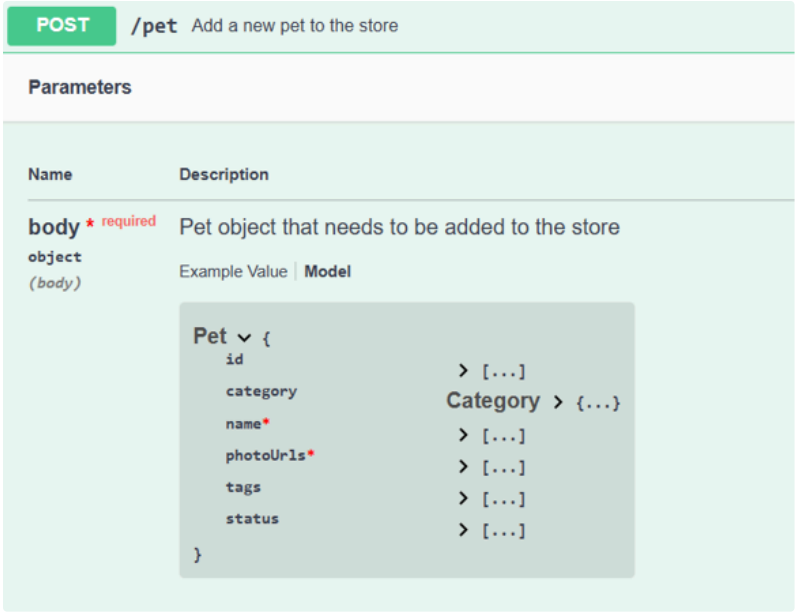

Figura 3 - Documentação explicita os campos obrigatórios

Resultado Esperado:

O endpoint POST /pet deveria validar a presença e o conteúdo dos campos obrigatórios (name e photoUrls) antes de persistir o recurso, retornando:

- 400 Bad Request com mensagem indicando campos obrigatórios ausentes ou inválidos, ou
- 422 Unprocessable Entity com detalhes de validação dos campos inválidos

A criação do recurso deveria ser rejeitada caso informações mínimas não sejam fornecidas.

Ambiente:

https://petstore.swagger.io/v2

Acesso:

Swagger UI (Swagger UI) e Postman

Sistema operacional:

Windows 11 Home Single Language

Data de identificação:

27/01/2026

Prioridade:

ALTA

Impacto:

Alto: Permite a criação de registros sem dados essenciais, comprometendo a integridade das informações, dificultando validações futuras e podendo gerar inconsistências funcionais no consumo da API.

Identificado por: Daniela Siana Pabis.

ISSUE 02:

Título: Endpoint POST /pet/{petId}/uploadImage retorna sucesso, mas não persiste nem associa a imagem ao pet

Descrição:

O endpoint POST /pet/{petId}/uploadImage aceita o envio de arquivos via multipart/form-data e retorna resposta de sucesso (200 OK), porém a imagem enviada não é persistida em nenhum local conhecido, nem associada ao pet informado. Após o upload, não existe forma de recuperar a imagem, nem por meio do recurso do pet, nem por outro endpoint da API.

Esse comportamento torna a funcionalidade inefetiva, pois o upload não gera nenhum impacto real no estado do sistema, caracterizando uma operação sem efeito funcional.

Reprodução:

1. Executar uma requisição POST /pet/{petId}/uploadImage;
2. Informar um petId válido existente;
3. Enviar um arquivo de imagem no campo file;
4. Enviar a requisição;
5. Observar que a API retorna 200 OK, porém a imagem não é persistida nem vinculada ao pet.

Evidências:

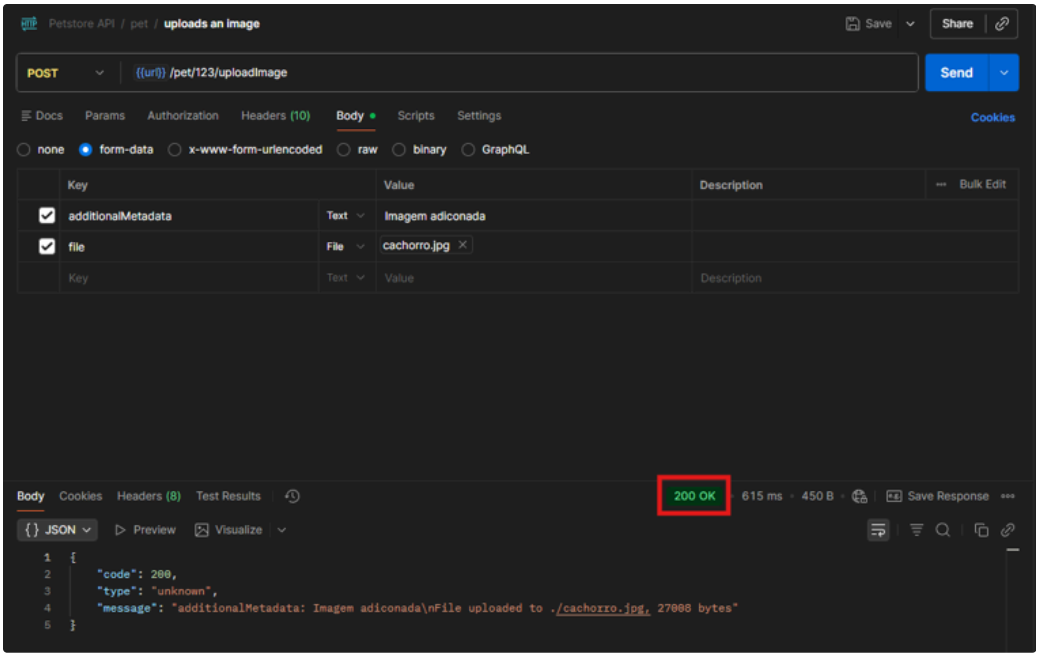

Figura 4 - Adição de imagem e retorno esperados

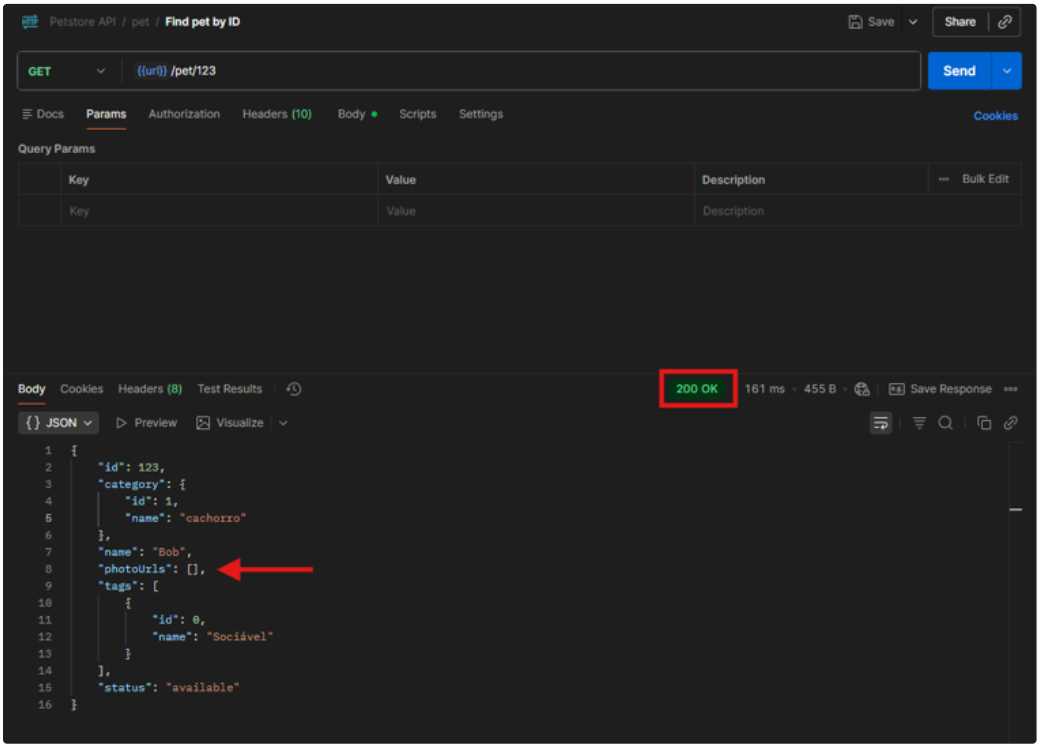

Figura 5 - Imagem não persiste

Resultado Esperado:

O endpoint POST /pet/{petId}/uploadImage deveria persistir a imagem enviada ou associá-la ao pet informado, retornando alguma referência (URL ou identificador), ou documentar explicitamente que o upload é apenas ilustrativo e não possui persistência.

Ambiente:

https://petstore.swagger.io/v2

Acesso:
Swagger UI (Swagger UI) e Postman

Sistema operacional:
Windows 11 Home Single Language

Data de identificação:
27/01/2026

Prioridade:
ALTA

Impacto:
Alta: Endpoint documentado e funcional do ponto de vista técnico, porém sem efeito prático, podendo induzir consumidores da API ao erro.

Identificado por:
Daniela Pabis

ISSUE 03:

Título: Endpoint PUT /pet cria novo pet quando informado ID inexistente e retorna sucesso

Descrição:
O endpoint PUT /pet permite a criação de um novo pet quando é informado um ID inexistente no body da requisição. Ao invés de retornar erro indicando que o recurso não foi encontrado, a API cria um novo pet e retorna resposta de sucesso.

Esse comportamento é incorreto do ponto de vista REST, pois o método PUT deveria atualizar apenas recursos existentes. A criação silenciosa de um novo recurso com PUT, sem documentação explícita desse comportamento, pode causar inconsistências e duplicação de dados.

Reprodução:

1. Executar uma requisição PUT /pet;
2. Informar no body um ID que não existe previamente no sistema;
3. Enviar a requisição;
4. Observar que a API retorna sucesso e cria um novo pet ao invés de retornar erro.

Evidências:

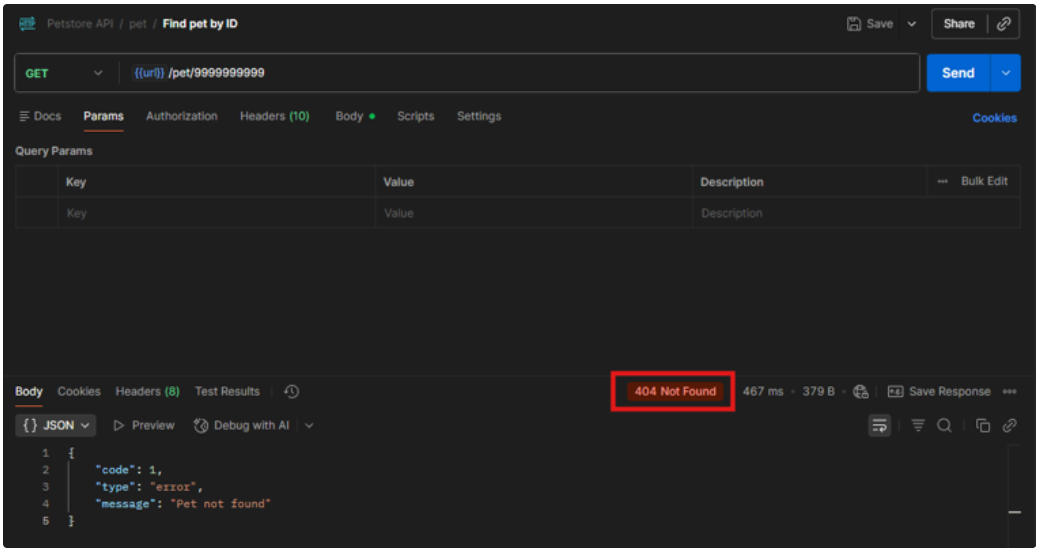

Figura 6 - Verificar existência de pet com id fantasia (9999999999)

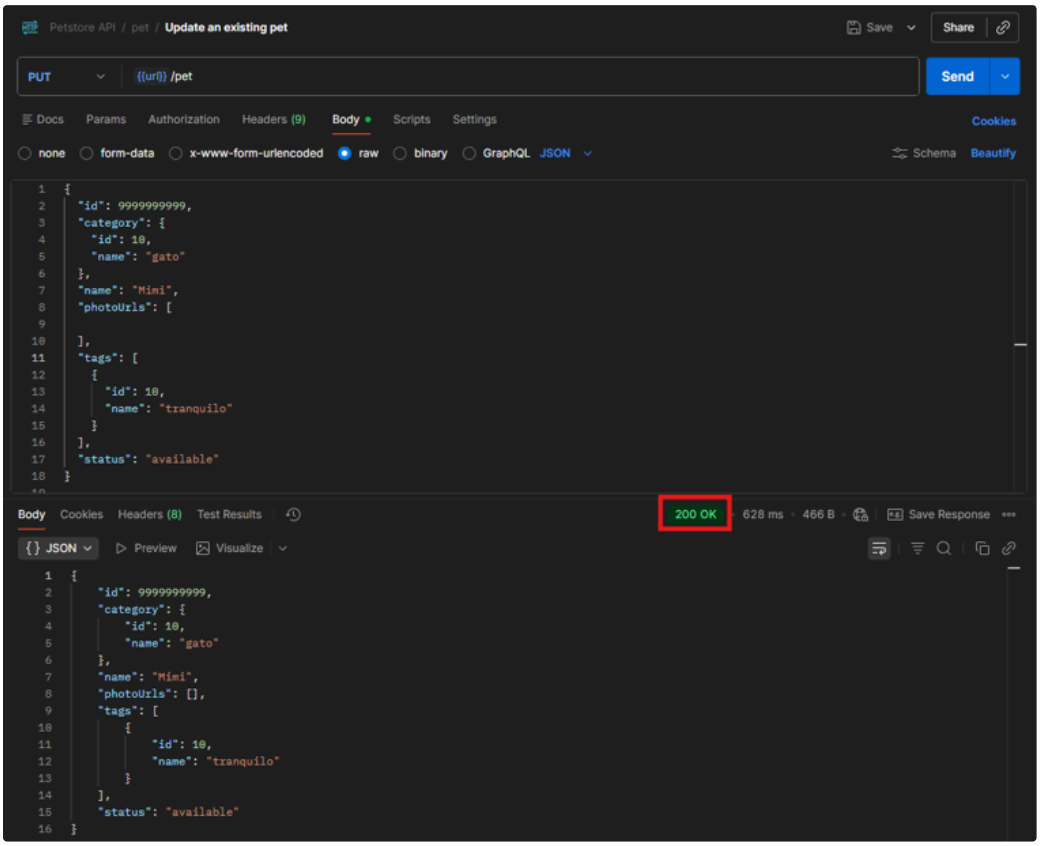

Figura 7 - Pet criado através do verbo PUT

Resultado Esperado:

O endpoint PUT /pet deveria validar a existência do recurso antes da atualização e retornar 404 Not Found quando o ID informado não existir, ou

Documentar explicitamente o comportamento de criação (upsert), caso essa seja a intenção da API.

Ambiente:
https://petstore.swagger.io/v2

Acesso:
Swagger UI (Swagger UI) e Postman

Sistema operacional:
Windows 11 Home Single Language

Data de identificação:
27/01/2026

Prioridade:
ALTA

Impacto:
Alto: Permite criação involuntária de recursos via PUT, quebra semântica REST, dificulta controle de dados e pode gerar duplicações inesperadas.

Identificado por:
Daniela Siana Pabis

ISSUE 4:

Título: API aceita status inválido ou ausente no cadastro de pet e retorna 200 OK, contrariando a documentação

Descrição:
De acordo com a documentação do Swagger Petstore, o campo status do recurso Pet possui valores predefinidos e a API deveria retornar erro 400 (Invalid input) quando um valor inválido é informado. No entanto, ao cadastrar um pet com um status inexistente ou sem informar o campo status, a API aceita a requisição e retorna 200 OK.

Além disso, ao recuperar o pet criado por meio do endpoint GET /pet/{petId}, o status inválido ou vazio é retornado normalmente, confirmando que o dado foi persistido sem validação. Esse comportamento diverge do que está descrito na documentação e permite a persistência de valores fora do domínio esperado.

Reprodução:

1. Executar uma requisição POST /pet ou PUT /pet;
2. Informar um valor inválido no campo status ou omitir completamente o campo;

3. Enviar a requisição;
4. Observar que a API retorna 200 OK;
5. Executar GET /pet/{petId} utilizando o ID retornado;
6. Observar que o status inválido ou vazio é retornado na resposta.

Evidências:

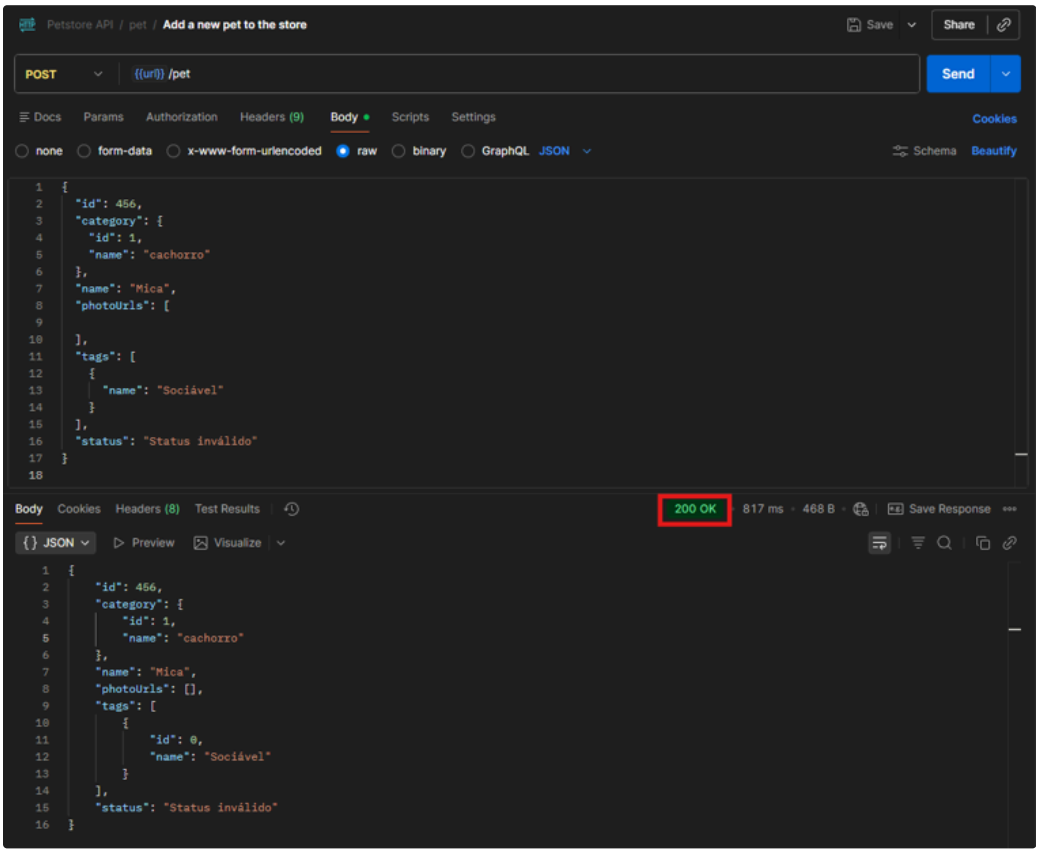

Figura 8 - Pet criado com status inexistente

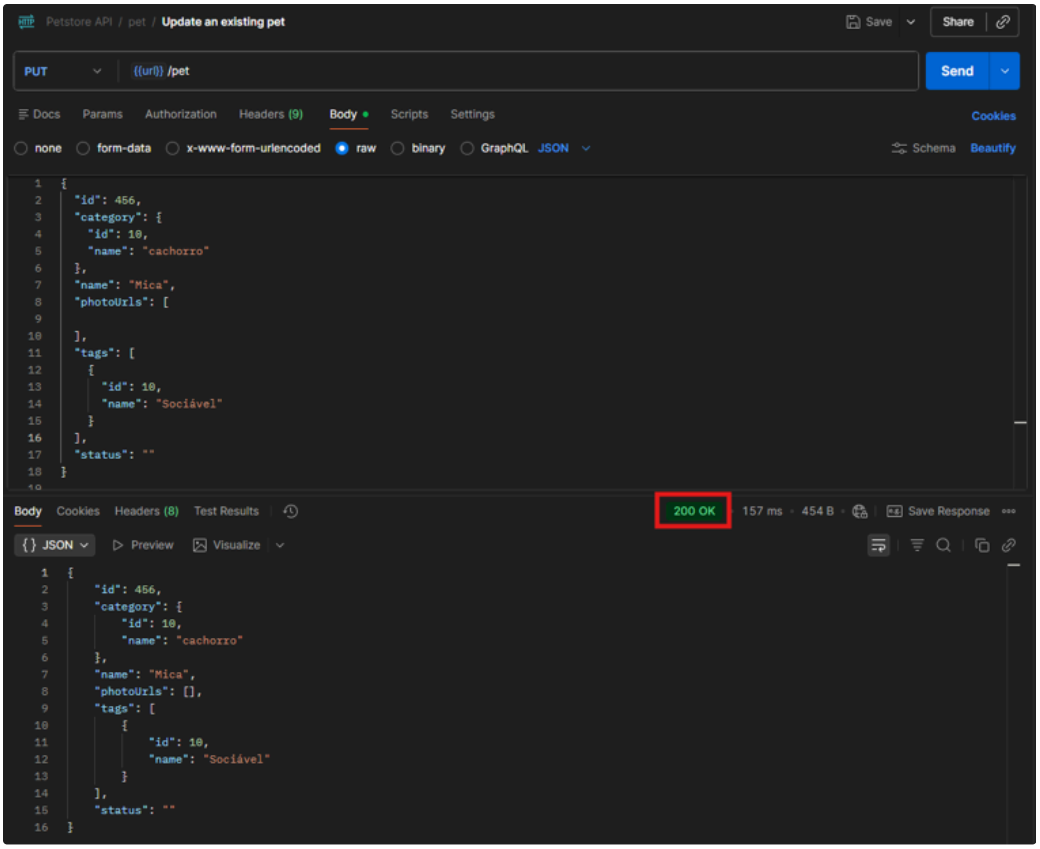

Figura 9 - Pet editado e salvo status vazio

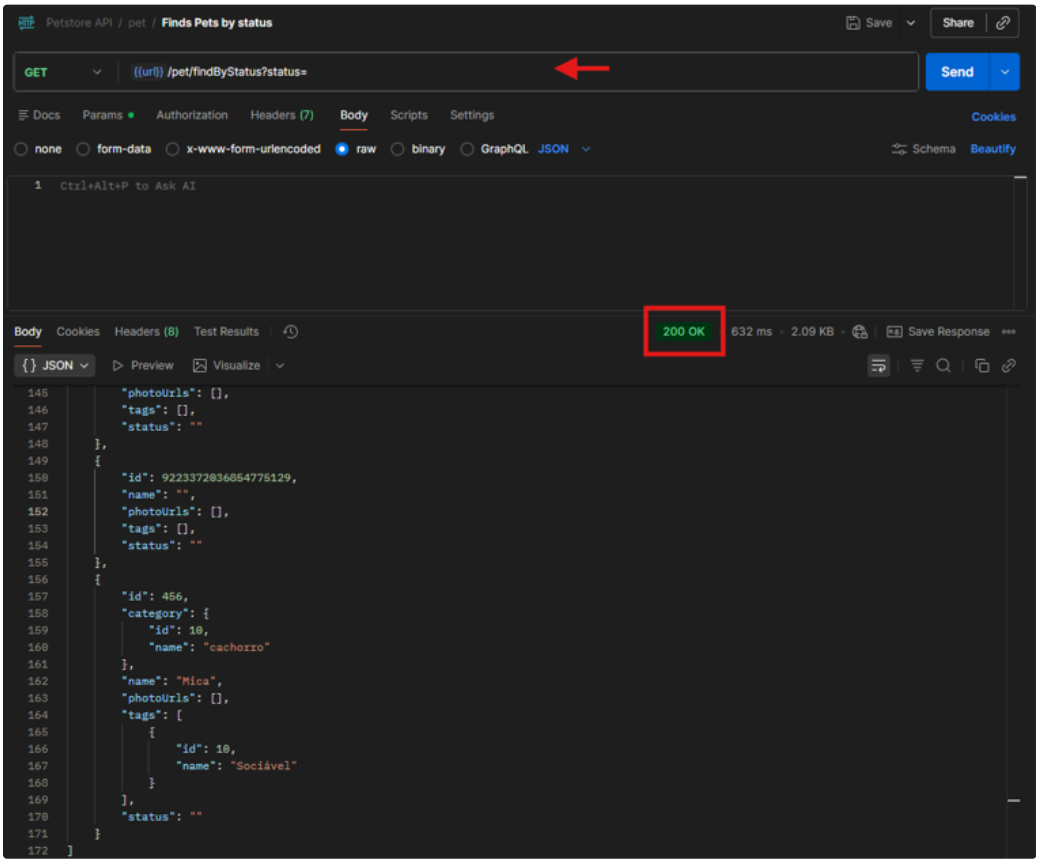

Figura 10 - Permite o retorno de status vazios e fora das opções descritas na documentação

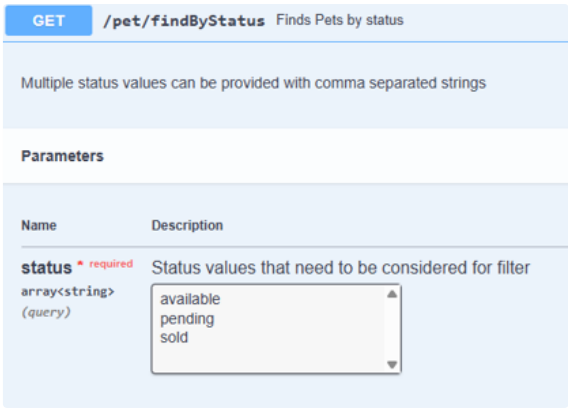

Figura 11 - Documentação informa a obrigatoriedade de status

Resultado Esperado:

A API deveria validar o valor do campo status conforme definido na documentação e retornar 400 Bad Request ao receber valores inválidos ou ausentes para o campo status, ou rejeitar a criação/atualização do recurso quando o status não estiver entre os valores permitidos.

Ambiente:

https://petstore.swagger.io/v2

Acesso:

Swagger UI (Swagger UI) e Postman

Sistema operacional:

Windows 11 Home Single Language

Data de identificação:

27/01/2026

Prioridade:

ALTA

Impacto:

Alto: Permite persistência de valores inválidos para o campo status, quebra o contrato documentado da API e pode causar falhas de lógica em sistemas consumidores que dependem de estados válidos.

Identificado por:

Daniela Siana Pabis

ISSUE 5:

Título: Sistema aceita criação de pedidos para pets deletados

Descrição: A API permite criar pedidos (orders) referenciando pets que foram previamente deletados do sistema. Após executar DELETE /pet/{petId} com sucesso (200 OK), é possível criar um pedido usando POST /store/order com o mesmo petId, resultando em pedidos órfãos sem referência válida a um produto. Isso quebra a integridade referencial dos dados e pode causar falhas no processamento de pedidos.

Reprodução:

1. Executar DELETE /pet/123 (ou qualquer ID válido);
2. Verificar que o retorno é 200 OK, confirmando a exclusão do pet;
3. Executar POST /store/order com o body contendo "petId": 123;
4. Observar que a API retorna 200 OK e cria o pedido normalmente, apesar do pet não existir mais.

Evidências:

Request 1:

Figura 12 - Retorno de status 200 com pet inexistente

DELETE https://petstore.swagger.io/v2/pet/123

Headers: api_key: special-key

Response 1:

Status: 200 OK

Request 2:

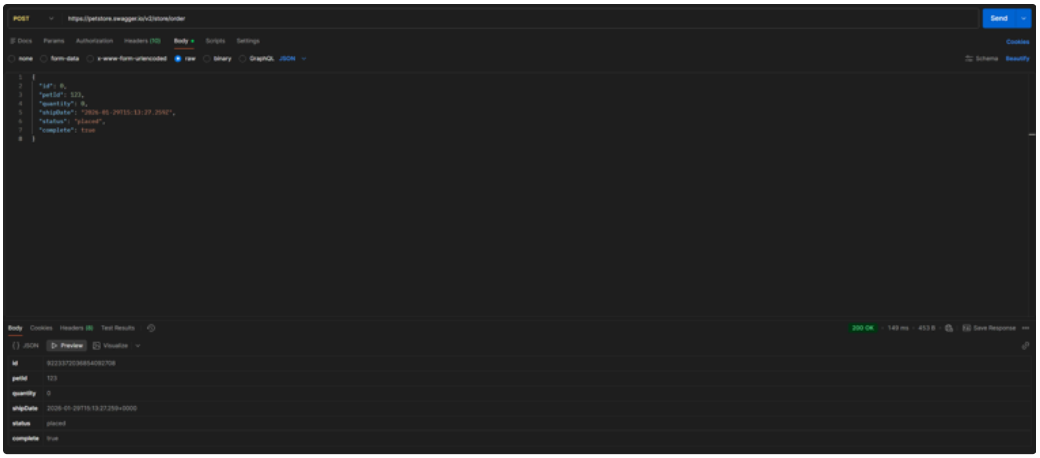

Figura 13 - Retorno de status 200com pet inexistente

POST https://petstore.swagger.io/v2/store/order

Content-Type: application/json

Body:

{

"id": 1,

"petId": 123,

"quantity": 1,

"shipDate": "2026-01-23T00:00:00.000Z",

"status": "placed",

"complete": false

}

Response 2:

Status: 200 OK

{

"id": 1,

"petId": 123,

"quantity": 1,

"shipDate": "2026-01-23T00:00:00.000+0000",

"status": "placed",

"complete": false

}

Resultado Esperado: POST /store/order deveria validar a existência do petId e retornar:

- 404 Not Found (Pet não encontrado), ou
- 400 Bad Request (petId inválido)

Ambiente: https://petstore.swagger.io/v2

Acesso: Swagger UI (Swagger UI) e Postman

Sistema operacional: Windows / Linux / macOS (independente de SO)

Data de identificação: 23/01/2026

Prioridade: ALTA

Impacto: Crítico - Quebra integridade referencial, gera pedidos inválidos, pode causar falhas no fulfillment

Identificado por: Andrey Felipe de Lima

ISSUE 6:

Título: Integer Overflow ao criar usuário com ID negativo

Descrição: A API aceita valores negativos no campo "id" ao criar usuários via POST /user, mas converte incorretamente o valor para um número extremamente grande (9223372036854775576), indicando Integer Overflow (conversão de signed para unsigned int64). Isso demonstra falta de validação de entrada e pode causar corrupção de dados, inconsistências no banco de dados e potenciais vulnerabilidades de segurança.

Reprodução:

1. Executar POST /user com body contendo "id": -1;
2. Verificar que a API retorna 200 OK;

3. Observar que o campo "message" no response contém "9223372036854775576" ao invés de "-1";
4. Comparar com comportamento normal: ao enviar "id": 1, o retorno é "message": "1" (correto).

Evidências:

1 - (ID positivo - comportamento normal):

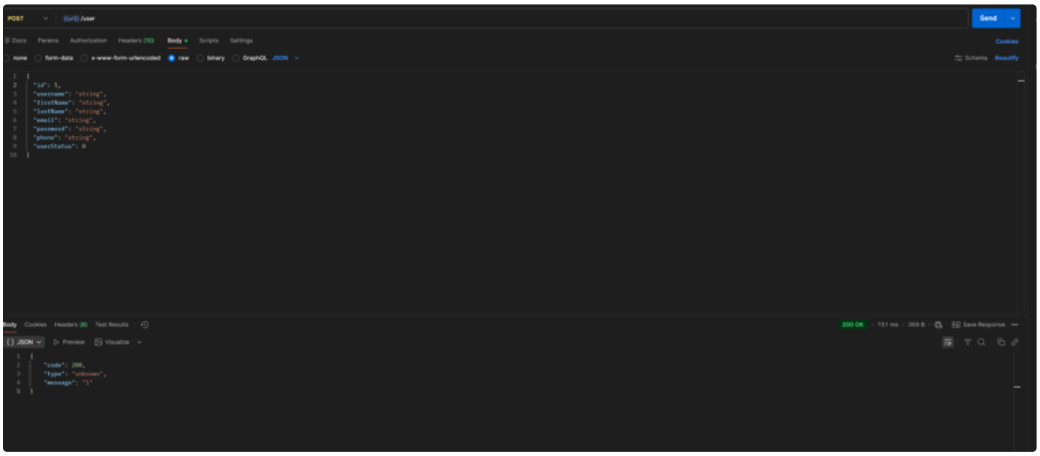

Figura 14 - Comportamento esperado

POST https://petstore.swagger.io/v2/user

Content-Type: application/json

Body:

```
{
  "id": 1,
  "username": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "password": "string",
  "phone": "string",
  "userStatus": 0
}
```

Response 1:

Status: 200 OK

{

"code": 200,

"type": "unknown",

"message": "1"

}

Request 2 (ID negativo - comportamento incorreto):

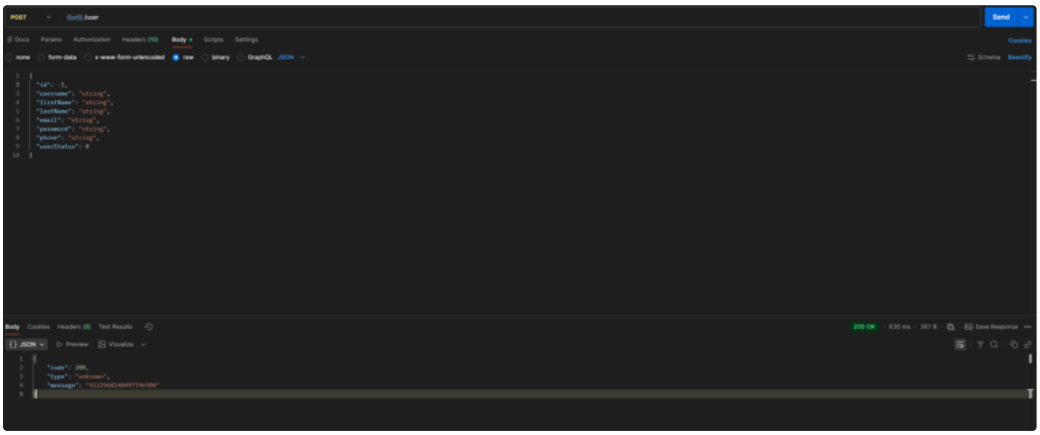

Figura 15 - Comportamento inesperado

POST https://petstore.swagger.io/v2/user

Content-Type: application/json

Body:

{

"id": -1,

"username": "string",

"firstName": "string",

"lastName": "string",

"email": "string",

"password": "string",

"phone": "string",

"userStatus": 0

}

Response 2:

Status: 200 OK

{

"code": 200,

"type": "unknown",

"message": "9223372036854775576"

}

Resultado Esperado: POST /user deveria validar o campo "id" e retornar:

- 400 Bad Request com mensagem: "ID deve ser um número positivo" ou similar
- Rejeitar a criação do usuário com ID inválido

Ambiente: https://petstore.swagger.io/v2

Acesso: Swagger UI (Swagger UI) e Postman

Sistema operacional: Windows / Linux / macOS (independente de SO)

Data de identificação: 23/01/2026

Prioridade: MÉDIA-ALTA

Impacto: Médio-Alto: Falta de validação de entrada, risco de corrupção de dados, potencial vulnerabilidade de segurança

Observação Técnica: O valor 9223372036854775576 sugere conversão de int64 signed (-1) para unsigned, comportamento típico em linguagens como Java/C++. Indica que outros campos numéricos podem ter o mesmo problema.

Identificado por: Andrey Felipe de Lima

ISSUE 7:

Título: API aceita ID inválido em criação de usuários em lote e retorna 200 OK

Descrição: A API aceita valores negativos no campo "id" ao criar múltiplos usuários via POST /user/createWithArray, retornando 200 OK com mensagem de sucesso ao invés de validar a entrada. Ao enviar um array de usuários contendo "id": -1, a API processa a requisição normalmente sem retornar erro de validação. Este comportamento indica falta de validação de entrada em operações batch, o que pode resultar em corrupção de dados em massa e inconsistências no banco de dados.

Reprodução:

1. Executar POST /user/createWithArray com body contendo array de usuários;
2. Incluir no array pelo menos um usuário com "id": -1 (valor negativo);
3. Enviar a requisição;
4. Observar que a API retorna 200 OK com message: "ok", aceitando o valor inválido.

Evidências:

Request:

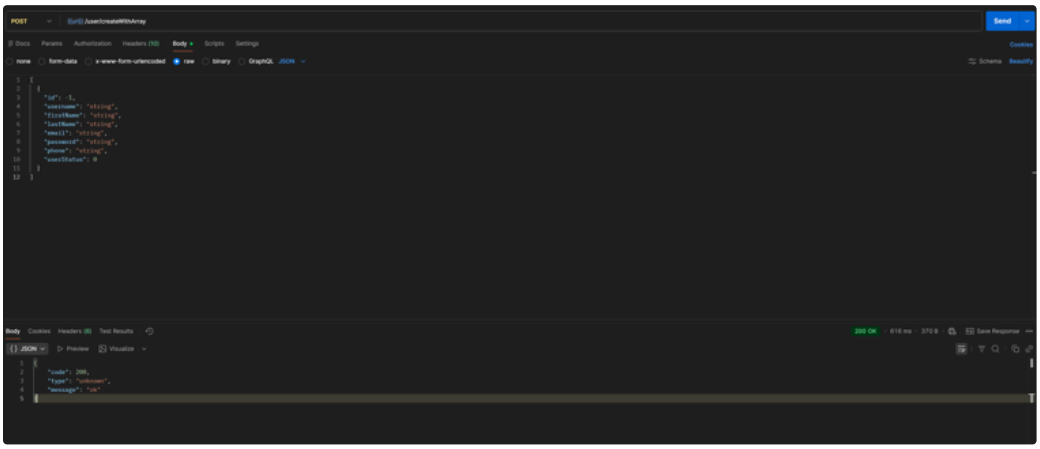

Figura 16 - Request aceita com valores inválidos

POST https://petstore.swagger.io/v2/user/createWithArray

Content-Type: application/json

Body:

```
[
  {
```

```
  "id": -1,
  "username": "user_test",
  "firstName": "Test",
  "lastName": "User",
  "email": "test@example.com",
  "password": "senha123",
  "phone": "123456789",
  "userStatus": 0
 }
]
```


Response:

Status: 200 OK


```
{
 "code": 200,
 "type": "unknown",
 "message": "ok"
}
```


Resultado Esperado: POST /user/createWithArray deveria validar todos os campos do array antes de processar e retornar:

- 400 Bad Request com mensagem: "ID inválido: valor deve ser positivo", ou
- 422 Unprocessable Entity com detalhes dos campos inválidos
- Rejeitar toda a operação batch se houver dados inválidos

Ambiente: https://petstore.swagger.io/v2

Acesso: Swagger UI (Swagger UI) e Postman

Sistema operacional: Windows 10

Data de identificação: 29/01/2026

Prioridade: ALTA

Impacto: Alto - Permite inserção de dados inválidos em massa, falta de validação em operações batch, risco de corrupção massiva de dados

Observação: Este bug é similar ao Bug #2, mas afeta o endpoint de criação em lote (createWithArray), demonstrando que a falta de validação é sistêmica e não isolada a um único endpoint. Outros endpoints batch (createWithList) podem ter o mesmo problema.

Identificado por: Andrey Felipe de Lima

ISSUE 8:

Título: Caractere especial em ID causa erro 500 Internal Server Error

Descrição: A API retorna 500 Internal Server Error ao receber caracteres especiais (como "@") no campo "id" via POST /user/createWithArray, ao invés de validar a entrada e retornar erro apropriado (400 Bad Request). Este comportamento indica falha crítica no tratamento de exceções do backend, expondo erros internos do servidor que deveriam ser capturados e tratados adequadamente. A mensagem genérica "something bad happened" não fornece informações úteis e pode indicar vulnerabilidade a ataques de injection.

Reprodução:

1. Executar POST /user/createWithArray com body contendo array de usuários;
2. Incluir no campo "id" um caractere especial (ex.: "@");
3. Enviar a requisição;
4. Observar que a API retorna 500 Internal Server Error ao invés de validar e retornar 400 Bad Request.

Evidências:

Request:

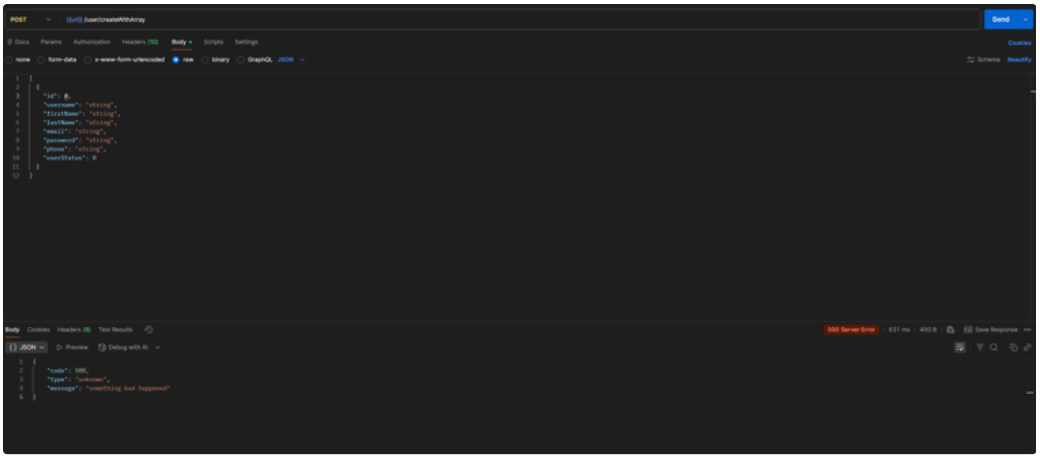

Figura 17 - Retorno de status code incorreto

POST https://petstore.swagger.io/v2/user/createWithArray

Content-Type: application/json

Body:

```
[
  {
    "id": "@",
    "username": "user_test",
    "firstName": "Test",
    "lastName": "User",
    "email": "test@example.com",
    "password": "senha123",
    "phone": "123456789",
    "userStatus": 0
  }
]
```

Response:

Status: 500 Internal Server Error

{

"code": 500,

"type": "unknown",

"message": "something bad happened"

}

Resultado Esperado: POST /user/createWithArray deveria validar o tipo de dado do campo "id" antes de processar e retornar:

- 400 Bad Request com mensagem: "ID inválido: deve ser um número inteiro", ou
- 422 Unprocessable Entity com detalhes do erro de validação
- NUNCA expor erros internos do servidor (500) por falha de validação de entrada

Ambiente: https://petstore.swagger.io/v2

Acesso: Swagger UI (Swagger UI) e Postman

Sistema operacional: Windows 10

Data de identificação: 29/01/2026

Prioridade: CRÍTICA

Impacto: Crítico - Falha no tratamento de exceções, expõe erros internos do servidor, mensagem genérica não informativa, possível vulnerabilidade a ataques de injection, degrada experiência do usuário

Observação Crítica: Este bug é especialmente grave porque:

1. Erro 500 indica que o servidor falhou ao processar a requisição (responsabilidade do servidor)
2. Deveria ser 400 (responsabilidade do cliente com input inválido)
3. Mensagem "something bad happened" é inadequada para produção
4. Sugere falta de validação de tipo de dados e tratamento de exceções
5. Pode ser explorado para identificar vulnerabilidades do sistema
6. Demonstra que a API não segue boas práticas de tratamento de erros

Relacionamento com outros bugs: Este bug se relaciona diretamente com issue #6 e issue #7, confirmando que a API tem falha sistêmica de validação de entrada. A progressão da severidade é alarmante:

- Issue #6: ID negativo → aceita silenciosamente
- Issue #7: ID negativo em batch → aceita silenciosamente
- Issue #8: ID com caractere especial → quebra o servidor

Identificado por: Andrey Felipe de Lima

ISSUE 9:

Título:
API aceita payload inválido no endpoint POST /store/order e retorna 200 OK criando pedido inconsistente

Descrição:

A API permite a criação de pedidos via endpoint **POST /store/order** mesmo quando o body da requisição não segue o contrato definido na documentação OpenAPI.

Ao enviar um payload contendo:

- campos inexistentes no schema (teste)
- campos duplicados
- tipos de dados inválidos
- ausência total dos campos esperados (status, complete, shipDate)

a API processa a requisição normalmente, retorna 200 OK e cria um pedido com valores default (petId: Integer Overflow, quantity: 0)

Esse comportamento é incorreto, pois demonstra ausência de validação de schema e permite a persistência de recursos inconsistentes, violando o contrato da API e comprometendo a confiabilidade do endpoint.

Reprodução:

1. Executar uma requisição POST /store/order
2. Enviar um body contendo apenas campos inexistentes e inválidos, por exemplo:
   - campos duplicados
   - tipos inconsistentes
3. Enviar a requisição
4. Observar que a API retorna 200 OK e cria um pedido com valores default

Evidências:

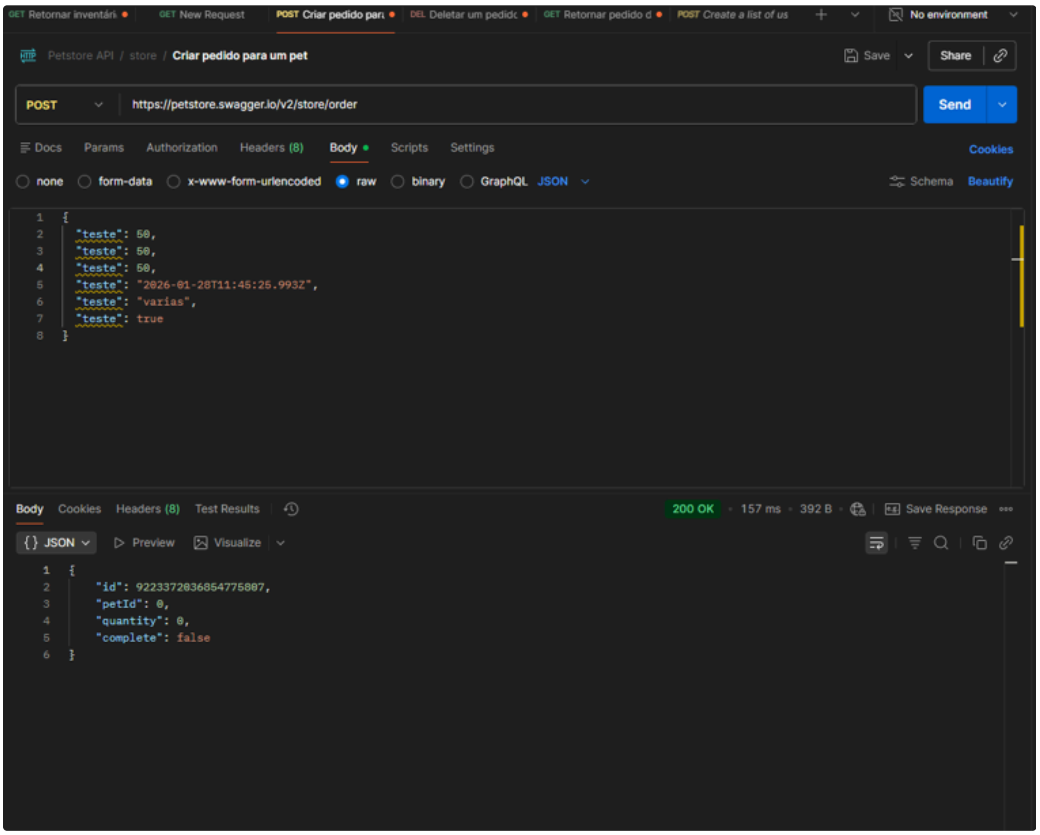

Figura 18 – Payload enviado contendo apenas campos inválidos

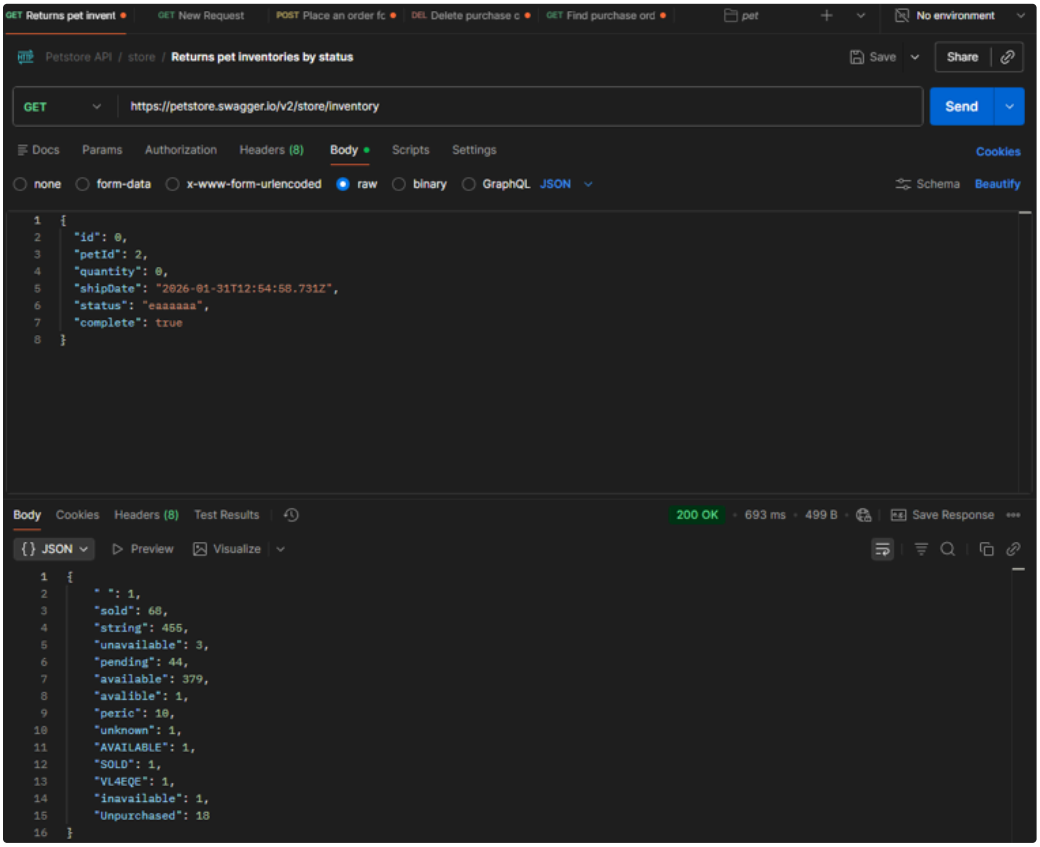

Figura 19 – Pedido criado com status vazio, retornado pelo get by status

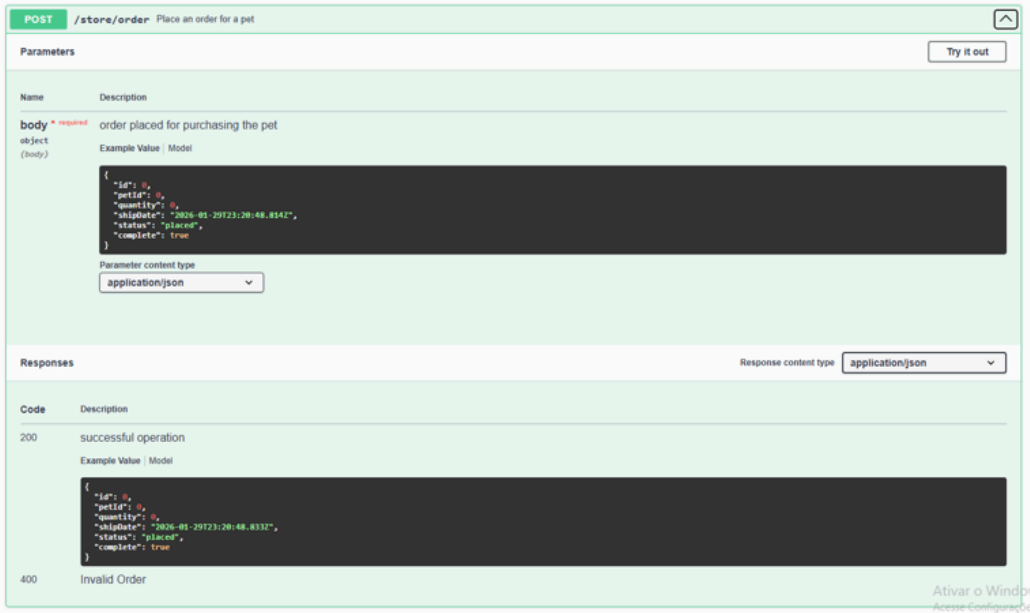

Figura 20 – Documentação Swagger indicando os campos esperados do recurso Order

Resultado Esperado:

O endpoint **POST /store/order** deveria validar rigorosamente o payload recebido, rejeitando requisições que não estejam em conformidade com o schema definido.

O comportamento esperado seria:

- Retornar **400 Bad Request** ou **422 Unprocessable Entity**
- Informar claramente que o body está inválido ou fora do contrato
- Não persistir o recurso em caso de erro de validação

Ambiente:
https://petstore.swagger.io/v2

Acesso:
Swagger UI e Postman

Sistema Operacional:
Windows 11 Home Single Language

Data de Identificação:
28/01/2026

Prioridade:
ALTA

Impacto:

Alto. A ausência de validação de schema permite:

- criação de pedidos inconsistentes
- quebra de contrato com consumidores da API
- dificuldade de rastreamento e validação de dados
- comportamento imprevisível em integrações futuras

Identificado por:
Eduardo Neves de Souza