

## Kauã Raffaello – Relatório de Issue/melhoria

Será mostrado o relatório dos 3 cenários de testes feitos, separados cada um por sua ID de Falha ou melhoria.

### [SVT 001]: Criar produto com autenticação válida (POST)

- *Tipo:* Falha funcional.
  - *Prioridade:* Média.
  - *Status:* Aberta.
- 

#### [SVT 001] Descrição:

- Foi visto uma melhoria a ser feita no body do Json que é retornado ao vendedor, falta informações do próprio produto que o mesmo cadastra no e-commerce, é algo muito simples que volta para o mesmo. É sugerido melhorar o corpo do JSON que retorna após ser criado por um método POST.
- 

#### [SVT 001] Passo a passo para verificação:

1. Autenticar como vendedor e obter o seu token.
2. Realizar um POST/produtos com o estilo de body abaixo:

```
1 {  
2   "nome": "I5 2°",  
3   "preco": 1800,  
4   "descricao": "CPU",  
5   "quantidade": 2  
6 }
```

3. Observar a resposta vaga da API.
- 

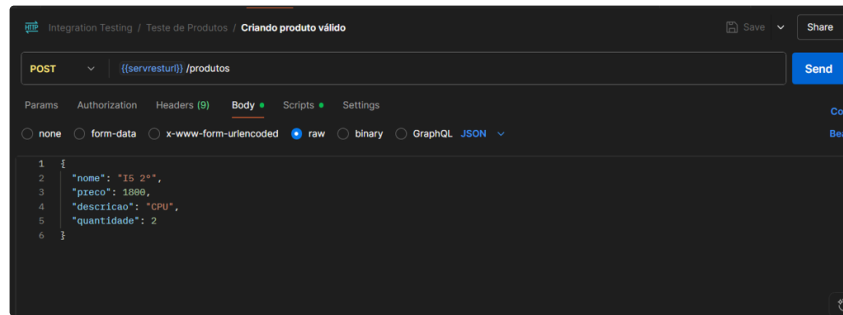
#### [SVT 001] Resultado esperado:

- Era esperado um body com todas as informações do que foi cadastrado além de ter apenas uma ID do produto e de uma mensagem simples de “cadastro realizado” porém, assim não causando conflito.

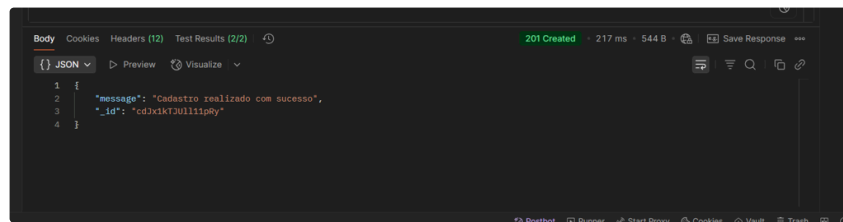
#### [SVT 001] Resultado obtido:

- Foi obtido um simples retorno de JSON, faltando informações do devido produto cadastrado, tais como: qual produto foi cadastrado; qual o tipo de produto e o preço.
  - É retornado apenas um ID do produto e uma mensagem bem simples de cadastro realizado. O que fica faltando informações de qual produto foi cadastrado.
-

## [SVT 001] Evidências:



Exemplo de body



Retorno simples e sem informações do que o vendedor cadastra

## [SVT 002]: Criar produto com dados inválidos(aleatórios) (POST)

- *Tipo:* Bug.
- *Prioridade:* Alta.
- *Status:* Aberta.

### [SVT 002] Descrição:

- Por meio de um script feito no pre-request do Postman, para cadastrar produtos aleatórios de forma errada, foi verificado um bug. Pois a API está deixando passar informações de cadastro sem verificar o produto.

### [SVT 002] Passo a passo para verificação:

1. Gerar token de vendedor.
2. Na parte de “Pre-request” colocar o script em javascript, abaixo:

```
1 {
2   const randomNum = Math.floor(Math.random() * 10000);
3
4   pm.environment.set("nomeProduto", `Produto Teste ${randomNum}`);
5   pm.environment.set("precoProduto", Math.floor(Math.random() * 200) + 1);
6   pm.environment.set("descricaoProduto", `Descrição automática ${randomNum}`);
7   pm.environment.set("quantidadeProduto", Math.floor(Math.random() * 50) + 1);
8 }
```

3. Agora no body que é obrigatório, trocar pelas variáveis que foram feitas, conforme o JSON abaixo:

```
1 {
2   "nome": "{{nomeProduto}}",
3   "preco": {{precoProduto}},
4   "descricao": "{{descricaoProduto}}",
5   "quantidade": {{quantidadeProduto}}
6 }
```

4. Salvar e clicar em “SEND”.

5. Como nos gera apenas uma ID, iremos usar um GET/produtos/{id} para verificar que foi cadastrado.

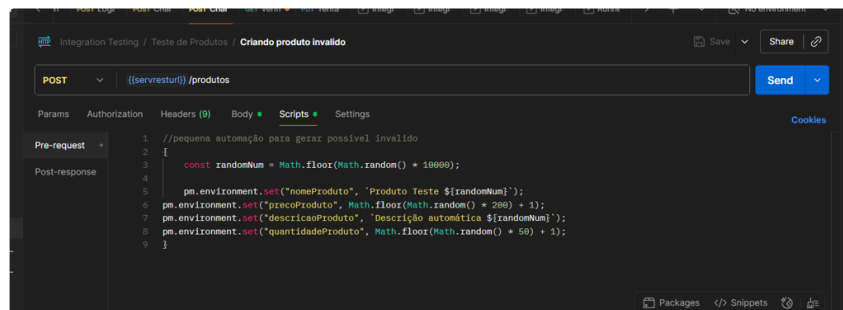
**[SVT 002] Resultado esperado:**

- Era esperado que a API impedisse o cadastro de um produto feito de forma aleatória ou que pelo menos tivesse um limite por vendedor.

**[SVT 002] Resultado obtido:**

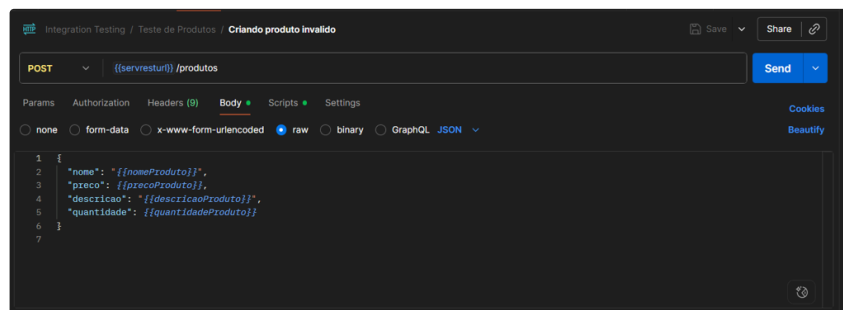
- A API gera normalmente o produto aleatório gerado por script de automação, sendo uma falha no código da mesma.

**[SVT 002] Evidências:**



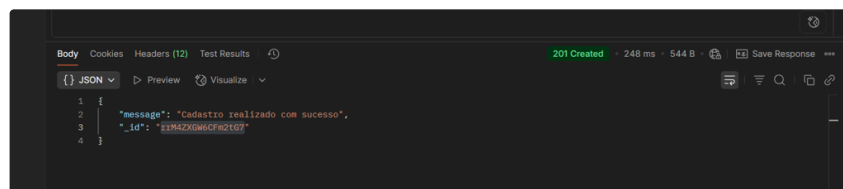
```
1 //pequena automação para gerar possível inválido
2
3 {
4   const randomNum = Math.floor(Math.random() * 10000);
5
6   pm.environment.set("nomeProduto", "Produto Teste ${randomNum}");
7   pm.environment.set("precoProduto", Math.floor(Math.random() * 200) + 1);
8   pm.environment.set("descricaoProduto", "Descrição automática ${randomNum}");
9   pm.environment.set("quantidadeProduto", Math.floor(Math.random() * 50) + 1);
10 }
```

Script sendo usado para realizar cadastros



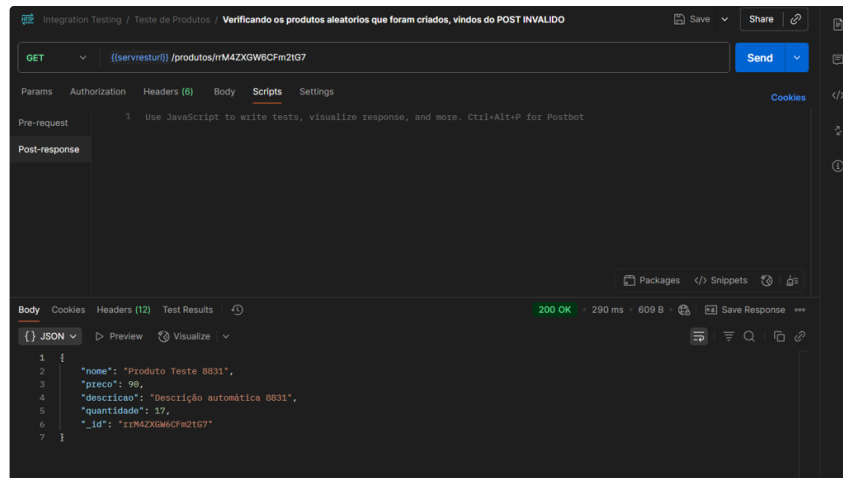
```
1 {
2   "nome": "{{nomeProduto}}",
3   "preco": {{precoProduto}},
4   "descricao": "{{descricaoProduto}}",
5   "quantidade": {{quantidadeProduto}}
6 }
7
```

Body em JSON que irá ser enviado para a API com os dados feitos pela automação



```
1 {
2   "message": "Cadastro realizado com sucesso",
3   "_id": "F1MAZX0H0CFN2T0Y"
4 }
```

201 Created, ou seja, criado normalmente



mostrando o resultado através do GET/produtos/{id}

### [SVT 003]: Atualizar produto inexistente (PUT)

- *Tipo:* Sugestão de melhoria.
- *Prioridade:* Baixa.
- *Status:* Aberta.

#### [SVT 003] Descrição:

- Através de automação foi tentado atualizar um produto existente ou inexistente, foi feito no Postman com uso de Runner para gerar a requisição várias vezes ao mesmo tempo.

#### [SVT 003] Passo a passo para verificação:

1. Gerar token de vendedor.
2. Colocar como método PUT: {{servesturl}}/prduto/{{produtoId}}
3. Em pre-request colocar o script de JavaScript:

```
1 let idUsado = pm.variables.get("produtoId");
2 let status = pm.response.code;
3
4 if(!isNaN(idUsado) && idUsado <= 5){
5   pm.test("Deve atualizar com sucesso (200)", function () {
6     pm.expect(status).to.eql(200);
7   });
8 } else {
9   pm.test("Deve retornar erro para ID inválido (404)", function () {
10    pm.expect(status).to.eql(404);
11  });
12 }
13
```

4. Em seu body para atualizar:

```
1 {
2   "nome": "Produto Teste",
3   "preco": 99.90,
4   "descricao": "Produto para teste automatizado"
5 }
6
```

5. Colocar no Runner para ser rodado automaticamente pelo menos 100 interações.

### [SVT 003] Resultado esperado:

- Era esperado que a API recusasse, pois estou tentando requisitar algo em uma ID aleatória.

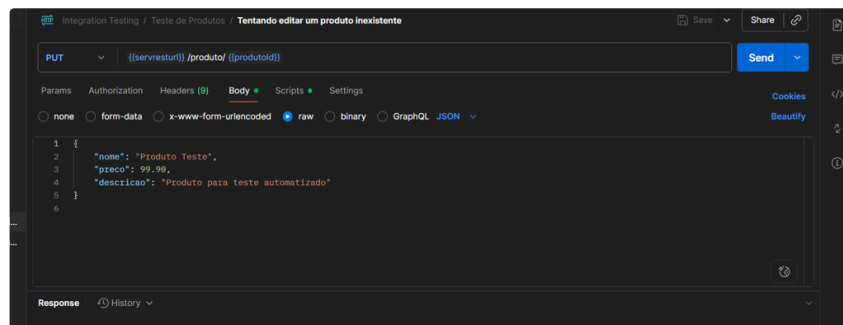
### [SVT 003] Resultado obtido:

- A API recusou todos os Runner de requisições pedidas.

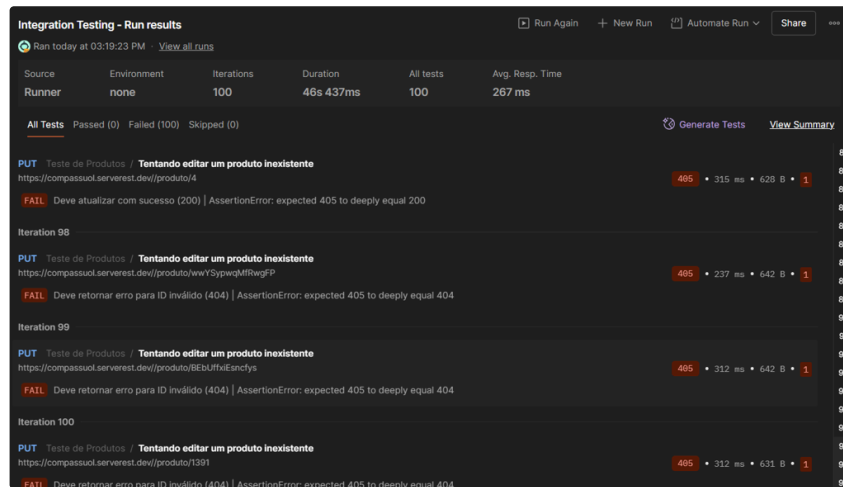
### [SVT 003] Evidências:



Script sendo usado



Body sendo usado



Total de 100 interações falhas

### [SVT 003] Sugestão de melhoria:

- Bloquear com mais segurança ao invés de apenas deixar as requisições forçadas, deverá ter um limite de requisições de PUT.

**Ambiente de teste:**

- Notebook: I5 de 11º
  - Software de teste de API: Postman
  - Usado navegador Brave Browser.
- 

**Responsável pelos testes:**

- Kauã Raffaello - QA (NoBugs).