

# Raphael - Relatório de Issues

## Resumo Executivo

Este relatório documenta três issues identificadas durante os testes da API do sistema de e-commerce, abrangendo falhas funcionais, bugs críticos e sugestões de melhoria no sistema.

---

### [SVT-001] Falha no Retorno de Cadastro de Produtos

**Tipo:** Falha Funcional

**Prioridade:** Média

**Status:** Aberta

**ID:** SVT-001

#### Descrição

Identificada uma falha no endpoint de criação de produtos onde o response body não retorna informações completas sobre o produto cadastrado, apenas ID e mensagem genérica de sucesso.

#### Passos para Reprodução

1. Autenticar como vendedor e obter token válido
2. Realizar POST para `/produtos` com body:

```
1 {  
2   "nome": "Geladeira",  
3   "preco": 1800,  
4   "descricao": "500W eletrolux",  
5   "quantidade": 2  
6 }
```

#### Resultados

- **Esperado:** Response body com todas informações do produto cadastrado
- **Obtido:** Apenas ID e mensagem "cadastro realizado"

#### Impacto

Vendedores não têm confirmação visual imediata dos dados cadastrados, necessitando de consulta adicional para verificação.

#### Evidência

```
1 // Response atual  
2 {  
3   "id": "pqpq-02i2e",
```

```
4   "message": "Cadastro realizado com sucesso"
5 }
6
7 // Response esperado
8 {
9   "id": "pqqq-02i2e",
10  "nome": "Geladeira",
11  "preco": 1800,
12  "descricao": "500W eletrolux",
13  "quantidade": 2,
14  "message": "Cadastro realizado com sucesso"
15 }
```

#### Recomendações

- Modificar endpoint POST `/produtos` para retornar objeto completo do produto criado
- Manter compatibilidade com consumers existentes

---

## [SVT-002] Validação Insuficiente no Cadastro de Produtos

**Tipo:** Bug

**Prioridade:** Alta

**Status:** Aberta

**ID:** SVT-002

#### Descrição

A API permite cadastro de produtos com dados aleatórios sem validações adequadas, possibilitando criação em massa de registros inválidos ou teste. Fora a ausência de validação de tipagem.

#### Passos para Reprodução

1. Configurar pre-request script no Postman:

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

2. Executar POST para `/produtos` com body usando variáveis aleatórias

#### Resultados

- **Esperado:** Validação e rejeição de dados aleatórios/inválidos
- **Obtido:** Criação bem-sucedida de produtos com dados aleatórios e em validação entre Integer s e String

## Impacto

- Base de dados poluída com registros inválidos
- Possível degradação de performance
- Risco de segurança por possível abuso do endpoint
- Uso indevido para SQL Injection

## Evidência

Request com dados aleatórios retorna status 201 Created:

```
1 {  
2   "id": "def456",  
3   "message": "Cadastro realizado com sucesso"  
4 }
```

## Recomendações

- Implementar validações de schema no request body
  - Adicionar rate limiting por usuário/vendedor
  - Validar se dados fazem sentido contextualmente (ex: preço > 0)
- 

## [SVT-003] Melhoria no Controle de Requisições Massivas

**Tipo:** Sugestão de Melhoria

**Prioridade:** Baixa

**Status:** Aberta

**ID:** SVT-003

### Descrição

Sugestão para implementar mecanismos adicionais de proteção contra requisições massivas em endpoints de atualização.

### Contexto

Testes com Postman Runner (100 iterações) para atualizar produtos com IDs inexistentes foram bloqueados, mas sistema pode beneficiar-se de proteções adicionais.

### Cenário de Teste

- Requisições PUT para `/produtos/{id}` com IDs aleatórios
- Script de validação no pre-request:

```
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 }
```

#### Resultados

- **Esperado:** Bloqueio de requisições massivas
- **Obtido:** Requisições recusadas individualmente (404)

#### Sugestão de Melhoria

Implementar rate limiting adicional para prevenir:

- Tentativas de força bruta
- Ataques de negação de serviço (DoS)
- Abuso de endpoints críticos

#### Recomendações

- Implementar limite de requisições por período (ex: 100 PUTs/hora por usuário)
- Adicionar mecanismo de timeout após múltiplas falhas
- Considerar uso de Redis para tracking de requisições

---

### [SVT-004] Melhoria no Retorno de Dados dos Usuários

**Tipo:** Sugestão de Melhoria

**Prioridade:** Alta

**Status:** Aberta

**ID:** SVT-003

#### Descrição

Sugestão para ocultar dados privados das conta dos usuários do sistema.

#### Contexto

Testes com Postman Flow para retornar os dados dos usuários terão senha. Mesmo que não foram bloqueados, o sistema deve minimamente criptografar a senha para beneficiar-se de proteções adicionais.

## Cenário de Teste

- Requisições GET para `/usuarios`
- Script de validação no pre-request:

```
1 let idUsado = pm.variables.get("UserId");
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 }
```

## Resultados

- **Esperado:** Bloqueio de requisições massivas
- **Obtido:** Requisições recusadas individualmente (404)

## Sugestão de Melhoria

Implementar rate limiting adicional para prevenir:

- Tentativas de força bruta
- Ataques de negação de serviço (DoS)
- Abuso de endpoints críticos

## Recomendações

- Implementar limite de requisições por período (ex: 100 PUTs/hora por usuário)
- Adicionar mecanismo de timeout após múltiplas falhas
- Considerar uso de Redis para tracking de requisições

---

## Ambiente de Teste

- **Hardware:** Notebook Intel i5 11ª Geração
- **Software:** Postman v10.0+
- **Navegador:** Opera GX Browser
- **Executor de Testes:** Postman Flow

**Responsável**

Raphael Rates - QA Analyst (NoBugs)

**Data:** 03/09/25

**Versão API Testada:** 1.24

---

*Este relatório contém informações confidenciais para uso interno da equipe de desenvolvimento e qualidade.*