Relatório – Challenge ServeRest API

Responsável do relatório

- **Emanuel Felipe Avelino Silva**

1. Introdução

O presente relatório tem como objetivo apresentar os resultados obtidos durante a execução do Challenge 01, cujo foco foi validar a API **ServeRest**.

A atividade proposta envolveu:

- A elaboração de um plano de testes com base nas User Stories fornecidas.
- A execução prática dos cenários planejados utilizando o Postman.
- O registro de bugs e melhorias encontrados durante o processo.

O propósito principal foi garantir que a aplicação atendesse às regras de negócio descritas, assegurando que os fluxos funcionais e alternativos estivessem cobertos, além de observar a clareza das respostas fornecidas pela API.

2. Plano de Testes (Previsto)

O plano de testes foi desenvolvido com base na documentação do Swagger e nas User Stories (US-001: Usuários, US-002: Login, US-003: Produtos e US-004: Carrinhos).

2.1 Escopo

Dentro do escopo estavam incluídos:

- Testes de criação, atualização, listagem e exclusão de usuários, produtos e carrinhos.
- Testes de autenticação e gerenciamento de sessão por token.
- Validação de regras de negócio, como unicidade de e-mails, senhas com restrição de tamanho, proibição de provedores específicos e uso de token válido.

Fora do escopo:

- Integrações externas.
- Performance, carga e stress test.

2.2 Técnicas Aplicadas

- **Testes funcionais manuais**: verificação prática das rotas via Postman.
- **Testes exploratórios**: além dos cenários previstos, foram exploradas possibilidades não documentadas.
- **Testes de validação de regras de negócio**: comparação do comportamento real da API com os Acceptance Criteria.

2.3 Matriz de Risco

Foram classificados os principais riscos:

- **Alta prioridade:** falhas no login e no cadastro de usuários/produtos, por impactarem diretamente a utilização da API.
- **Média prioridade:** inconsistências em mensagens de erro ou exclusões de dados.
- **Baixa prioridade:** cenários de consulta de registros inexistentes.

2.4 Cenários Planejados

Entre os cenários planejados estavam:

- Cadastro de usuário válido e duplicado.
- Login com sucesso, senha inválida e usuário inexistente.
- Cadastro de produto válido, duplicado e sem autenticação.
- Criação, consulta e exclusão de carrinhos.

3. Testes Executados (Realizado)

Os testes foram executados em todos os endpoints previstos, e em alguns casos foram realizados cenários além dos planejados para validar comportamentos inesperados.

3.1 Usuários

Foram realizados 15 testes. Principais achados:

- Cadastro de usuários com senhas fora do limite permitido (menor que 5 e maior que 10) foi aceito, caracterizando bug.
- Cadastro com provedores proibidos (Gmail/Hotmail) também foi aceito, em desacordo com as regras.
- A exclusão de usuários retorna mensagens genéricas (“Nenhum registro excluído”), mesmo em casos distintos (usuário inexistente ou já excluído).

3.2 Login

Foram realizados 4 testes. Principais achados:

- Login com credenciais válidas gera corretamente um token.
- Tentativas com senha inválida ou usuário inexistente foram rejeitadas conforme esperado.
- Validação de campos obrigatórios também funcionou corretamente.
- Nenhum bug identificado nesta rota.

3.3 Produtos

Foram realizados 12 testes. Principais achados:

- Cadastro de produtos válidos funciona corretamente.

- Cadastro sem autenticação ou com usuário não-admin é bloqueado.
- Ao atualizar produto com ID inexistente, a mensagem retornada é incorreta (“Já existe produto com esse nome”), configurando bug.
- A exclusão de produtos inexistentes gera mensagem pouco clara.

3.4 Carrinhos

Foram realizados 11 testes. Principais achados:

- É possível criar, consultar e excluir carrinhos corretamente.
- O sistema impede que o mesmo usuário tenha mais de um carrinho.
- Concluir ou cancelar compra duas vezes retorna mensagens genéricas que poderiam ser mais claras.
- Cancelar compra com token inválido retorna erro de autenticação como esperado.

4. Diferenças entre Planejado x Executado

4.1 Convergências

- A maior parte dos cenários planejados foi executada com sucesso.
- Os fluxos principais (CRUD de usuários/produtos, autenticação, carrinhos) foram validados.

4.2 Divergências

- Durante a execução, foram explorados cenários não previstos, revelando bugs não mapeados no plano inicial, como a falha nas regras de senha e de provedores de email.
- Algumas mensagens de erro não atendem aos critérios de clareza, o que não estava previsto no plano inicial mas foi identificado durante os testes exploratórios.

5. Bugs Identificados

Foram identificados 4 bugs:

1. Cadastro permitido com Gmail/Hotmail.
2. Cadastro permitido com senha menor que 5 caracteres.
3. Cadastro permitido com senha maior que 10 caracteres.
4. Atualização de produto com ID inválido retorna mensagem incorreta.

6. Melhorias Identificadas

Foram sugeridas 7 melhorias:

1. Evitar criação automática ao atualizar usuário inexistente.
2. Mensagem mais clara ao deletar usuário já excluído.
3. Mensagem mais clara ao deletar usuário inexistente.

4. Mensagem mais clara ao cadastrar produto duplicado.
5. Mensagem mais clara ao deletar produto inexistente.
6. Mensagem mais clara ao concluir compra sem carrinho ativo.
7. Mensagem mais clara ao cancelar compra sem carrinho ativo.

7. Conclusão e Recomendações

De forma geral, a API ServeRest apresenta uma boa cobertura funcional e atende parcialmente às regras de negócio.
Os fluxos principais estão implementados corretamente, porém há falhas críticas nas validações de senha e email, que comprometem a confiabilidade.

Outro ponto importante é a clareza das mensagens de retorno: em muitos casos, o sistema retorna mensagens genéricas que não ajudam o usuário a entender a causa do erro.

Recomendações:

- Correção prioritária dos bugs relacionados a cadastro de usuário (senha e email).
- Revisão das mensagens de erro, para melhorar a clareza e usabilidade.
- Avaliar ajuste no comportamento de atualização de usuários inexistentes.
- Investir em automação de testes nos cenários críticos (login, cadastro e autenticação de produtos).

8. Anexos e Evidências

Anexo – A:

Mapeamento De Testes

/USUARIOS				
ID	**Cenário de Teste**	**Resultado Esperado**	**Resultado Obtido**	**Status**
US-01	Buscar usuário por ID válido	Retornar os dados do usuário	**200**: informações do usuário retornadas	✅ Correto
US-02	Buscar usuário por ID inexistente	Retornar erro informando que não existe	**400**: "Usuário não encontrado"	✅ Correto
US-03	Cadastrar usuário válido	Usuário deve ser cadastrado com sucesso	**201**: "Cadastro realizado com sucesso" + ID	✅ Correto
US-04	Cadastrar usuário com e-mail Gmail ou Hotmail	Sistema deveria bloquear o cadastro	**201**: cadastro realizado → **Bug**	❌ Bug
US-05	Cadastrar usuário com senha menor que 5 caracteres	Sistema deveria bloquear o cadastro	**201**: cadastro realizado → **Bug**	❌ Bug

US-06	Cadastrar usuário com senha maior que 10 caracteres	Sistema deveria bloquear o cadastro	**201**: cadastro realizado → **Bug**	❌ Bug
US-07	Cadastrar usuário duplicado	Bloquear cadastro e avisar que já existe	**400**: "Este email já está sendo usado"	✅ Correto
US-08	Cadastrar usuário com e-mail inválido	Retornar erro indicando formato incorreto	**400**: "email deve ser um email válido"	✅ Correto
US-09	Atualizar usuário existente com body correto	Usuário deve ser atualizado com sucesso	**200**: "Registro alterado com sucesso"	✅ Correto
US-10	Atualizar usuário inexistente	Retornar erro informando que não existe	**201**: cria novo usuário automaticamente → ⚠️ Pode ser melhorado	⚠️ Melhoria
US-11	Atualizar campo administrador com valor inválido	Retornar erro informando os valores aceitos	**400**: "administrador deve ser 'true' ou 'false'"	✅ Correto
US-12	Atualizar usuário sem enviar body	Sistema deve informar campos obrigatórios	**400**: mensagens indicando campos obrigatórios	✅ Correto
US-13	Deletar usuário existente	Usuário deve ser excluído com sucesso	**200**: "Registro excluído com sucesso"	✅ Correto
US-14	Deletar usuário já excluído	Informar que não há nada para excluir	**200**: "Nenhum registro excluído"	⚠️ Mensagem pode ser mais clara
US-15	Deletar usuário inexistente	Informar que não há nada para excluir	**200**: "Nenhum registro excluído"	⚠️ Mensagem pode ser mais clara

/LOGIN					
ID	**Cenário**	**Entrada/Descrição**	**Resultado Esperado**	**Resultado Obtido**	**Status**
CT-L01	Login com dados corretos	POST /login com email e senha válidos	200 OK + token Bearer	200 OK – token gerado com sucesso	✅ OK
CT-L02	Login sem body	POST /login sem email/senha no body	400 Bad Request – campos obrigatórios	400 – mensagem exibindo que email e password	✅ OK

				são obrigatórios	
CT-L03	Login com senha incorreta	POST /login com email válido e senha inválida	401 Unauthorized	401 Unauthorized – “Email e/ou senha inválidos”	✅ OK
CT-L04	Login com email inválido	POST /login com formato inválido de email	400 Bad Request	400 – “email deve ser um email válido”	✅ OK

/PRODUTOS				
ID	**Cenário de Teste**	**Resultado Esperado**	**Resultado Obtido**	**Status**
PR-01	Cadastrar produto com autenticação expirada	Retornar erro de autenticação	Erro **401**: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	✅ Correto
PR-02	Cadastrar produto já existente	Retornar erro informando que o produto já existe	Erro **400**: "Já existe produto com esse nome" → mensagem poderia ser mais clara	⚠️ Melhoria
PR-03	Cadastrar produto válido com token válido	Produto cadastrado com sucesso (201)	**201**: "Cadastro realizado com sucesso" + ID do produto	✅ Correto
PR-04	Buscar produto por ID válida	Retornar informações do produto (200)	**200** com dados do produto	✅ Correto
PR-05	Buscar produto com ID inexistente	Retornar erro informando que produto não existe	Erro **400**: "Produto não encontrado"	✅ Correto
PR-06	Atualizar produto válido	Produto atualizado com sucesso (200)	**200**: "Registro alterado com sucesso"	✅ Correto
PR-07	Atualizar produto com ID inválido	Retornar erro informando que ID não existe	**400**: "Já existe produto com esse nome" → provavelmente um **bug** (mensagem incorreta)	❌ Bug
PR-08	Atualizar produto com token de usuário não-admin	Retornar erro de permissão	**403**: "Rota exclusiva para administradores"	✅ Correto

PR-09	Cadastrar produto com token de usuário não-admin	Retornar erro de permissão	**403**: "Rota exclusiva para administradores"	✅ Correto
PR-10	Deletar produto válido	Produto excluído com sucesso (200)	**200**: "Registro excluído com sucesso"	✅ Correto
PR-11	Deletar produto com ID inexistente	Retornar erro ou aviso claro que não foi excluído	**200**: "Nenhum registro excluído com sucesso" → mensagem confusa	⚠️ Melhoria
PR-12	Buscar todos os produtos cadastrados	Retornar lista de produtos (200)	**200**: lista de todos os produtos cadastrados	✅ Correto

/CARRINHOS				
ID	**Cenário de Teste**	**Resultado Esperado**	**Resultado Obtido**	**Status**
CA-01	Buscar todos os carrinhos (GET)	Retornar lista de carrinhos (200)	**200**: lista dos carrinhos existentes	✅ Correto
CA-02	Buscar carrinho por ID válida (GET)	Retornar lista de produtos do carrinho (200)	**200**: lista dos produtos dentro do carrinho	✅ Correto
CA-03	Buscar carrinho por ID inexistente (GET)	Retornar erro informando que não existe	**400**: "Carrinho não encontrado"	✅ Correto
CA-04	Cadastrar carrinho válido (POST)	Criar carrinho e retornar sucesso (201)	**201**: "Cadastro realizado com sucesso" + ID do carrinho	✅ Correto
CA-05	Cadastrar carrinho repetido (POST)	Retornar erro indicando que já existe carrinho	**400**: "Não é permitido ter mais de 1 carrinho"	✅ Correto
CA-06	Adicionar produto além da quantidade disponível (POST)	Retornar erro informando falta de estoque	**400**: "Produto não possui quantidade suficiente"	✅ Correto
CA-07	Concluir compra (DELETE)	Excluir carrinho e retornar sucesso (200)	**200**: "Registro excluído com sucesso"	✅ Correto
CA-08	Concluir compra novamente no mesmo usuário (DELETE)	Retornar erro ou aviso de que não há carrinho	**200**: "Não foi encontrado carrinho para esse usuário"	⚠️ Melhoria (mensagem poderia ser mais clara)
CA-09	Cancelar compra (DELETE)	Cancelar carrinho e	**200**: "Registro excluído com sucesso. Estoque dos	✅ Correto

		reabastecer estoque (200)	produtos reabastecido"	
CA-10	Cancelar compra novamente no mesmo usuário (DELETE)	Retornar erro ou aviso de que não há carrinho	**200**: "Não foi encontrado carrinho para esse usuário"	⚠️ Melhoria (mensagem poderia ser mais clara)
CA-11	Cancelar compra com token inválido/expirado (DELETE)	Retornar erro de autenticação (401)	**401**: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	✅ Correto

Evidência – A:

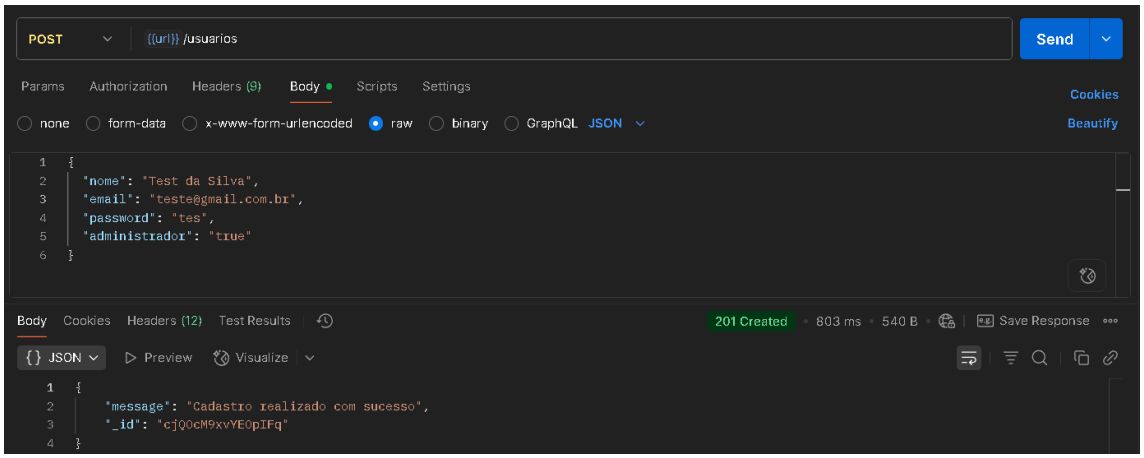
Detalhamento Das Issues

ID da Issue: ISS-001

Título: Cadastro permitido com Gmail/Hotmail
Descrição: O sistema permite cadastrar usuários com provedores de email proibidos (Gmail e Hotmail).
Passos para Reproduzir:

1. Fazer uma requisição POST para /usuarios.
2. Informar um email do tipo teste@gmail.com.
3. Verificar resposta da API.

Resultado Obtido: 201 Created – usuário cadastrado com sucesso.
Resultado Esperado: A API deveria bloquear o cadastro e retornar erro 400.
Gravidade/Prioridade: Alta.
Evidências:

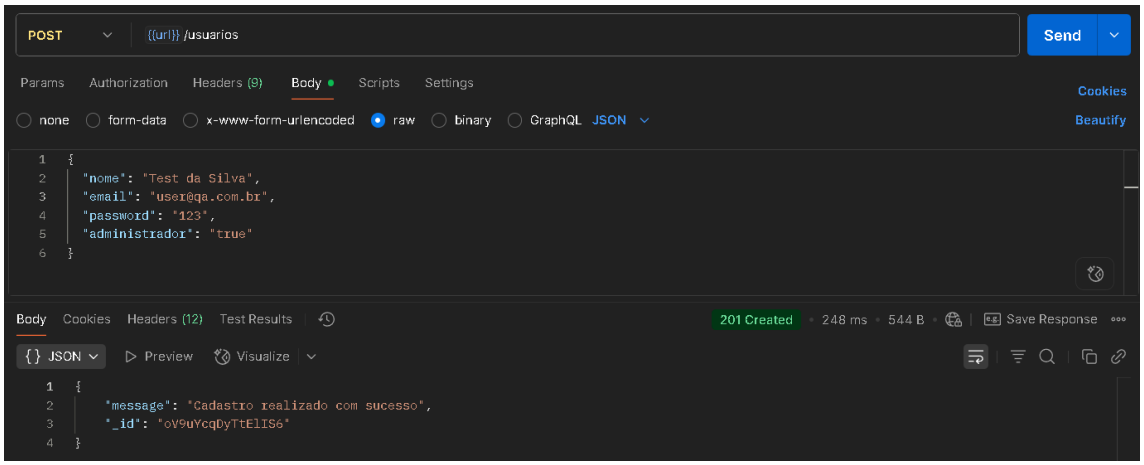


ID da Issue: ISS-002

Título: Cadastro permitido com senha menor que 5 caracteres
Descrição: O sistema permite cadastrar usuários com senha menor que 5 dígitos, contrariando as regras definidas.
Passos para Reproduzir:

1. Fazer uma requisição POST para /usuarios.
2. Informar senha com 3 caracteres, ex.: "123".
3. Verificar resposta da API.

Resultado Obtido: 201 Created – usuário cadastrado com sucesso.
Resultado Esperado: A API deveria retornar erro 400, informando que a senha deve ter no mínimo 5 caracteres.
Gravidade/Prioridade: Alta.
Evidência:

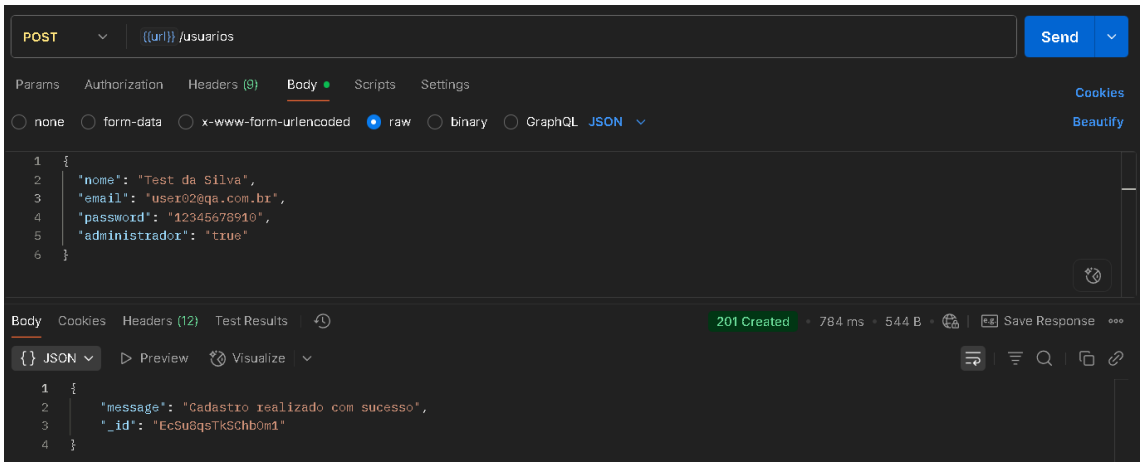


ID da Issue: ISS-003

Título: Cadastro permitido com senha maior que 10 caracteres
Descrição: O sistema permite cadastrar usuários com senha maior que 10 dígitos, contrariando as regras definidas.
Passos para Reproduzir:

1. Fazer uma requisição POST para /usuarios.
2. Informar senha com 11 caracteres, ex.: "12345678901".
3. Verificar resposta da API.

Resultado Obtido: 201 Created – usuário cadastrado com sucesso.
Resultado Esperado: A API deveria retornar erro 400, informando que a senha deve ter no máximo 10 caracteres.
Gravidade/Prioridade: Alta.
Evidência:

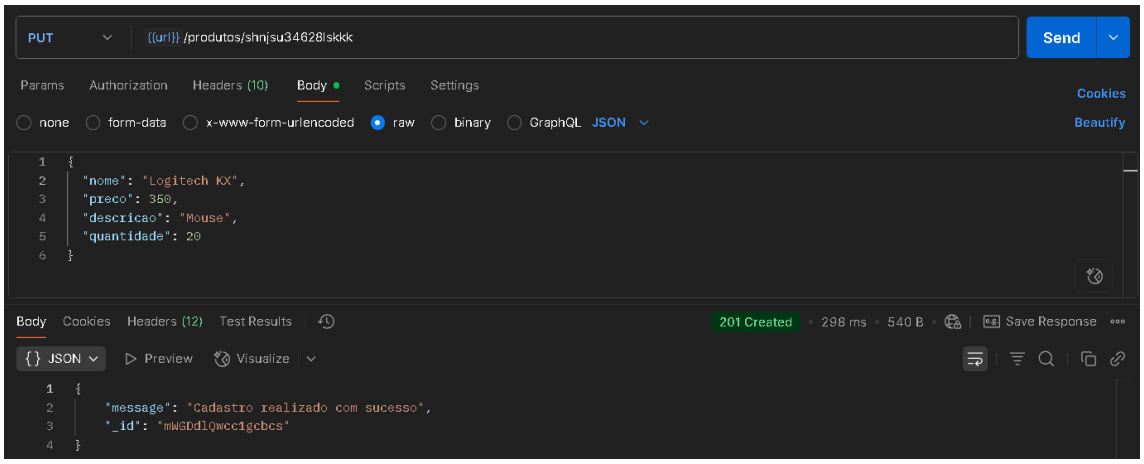


ID da Issue: ISS-004

Título: Atualização de produto com ID inválido retorna mensagem incorreta
Descrição: Ao atualizar um produto com ID inexistente, a API retorna mensagem incorreta de erro.
Passos para Reproduzir:

1. Fazer uma requisição PUT para /produtos/{id_invalido}.
2. Enviar body com dados de produto.
3. Verificar resposta da API.

Resultado Obtido: 400 – mensagem: "Já existe produto com esse nome".
Resultado Esperado: A API deveria retornar mensagem clara: "Produto não encontrado" ou "ID inválido".
Gravidade/Prioridade: Média.
Evidência:



Anexo – B:

Mapeamento De Melhorias

Relatório De Melhorias						
ID	**Título**	**Descrição**	**Passos para Reprodução**	**Resultado Obtido**	**Resultado Esperado**	**Prioridade**
MEL-01	Atualização de usuário inexistente cria novo usuário	O sistema cria usuário automaticamente ao tentar atualizar um inexistente.	1. PUT /usuarios/{id_invalido}2. Verificar resposta	**201 Created** – novo usuário criado	Retornar erro informando que o usuário não existe	Média
MEL-02	Mensagem ao deletar usuário já excluído	Mensagem pode causar confusão.	1. DELETE /usuarios/{id_valido} (2x)2. Verificar resposta	**200** – "Nenhum registro excluído"	Mensagem mais clara, ex.: "Usuário já excluído"	Baixa
MEL-03	Mensagem ao deletar usuário inexistente	Mensagem pode causar confusão.	1. DELETE /usuarios/{id_invalido}2. Verificar resposta	**200** – "Nenhum registro excluído"	Mensagem mais clara, ex.: "Usuário não encontrado"	Baixa
MEL-04	Mensagem ao cadastrar produto duplicado	Mensagem poderia ser mais específica.	1. POST /produtos com nome já existente2. Verificar resposta	**400** – "Já existe produto com esse nome"	Mensagem clara: "Produto já cadastrado"	Média
MEL-05	Mensagem ao deletar produto inexistente	Mensagem pouco clara.	1. DELETE /produtos/{id_invalido}2. Verificar resposta	**200** – "Nenhum registro excluído com sucesso"	Mensagem clara: "Produto não encontrado"	Baixa
MEL-06	Mensagem ao concluir compra sem carrinho	Mensagem pode ser melhorada.	1. DELETE /carrinhos concluir compra sem carrinho2. Verificar resposta	**200** – "Não foi encontrado carrinho para esse usuário"	Mensagem clara: "Usuário não possui carrinho ativo"	Média
MEL-07	Mensagem ao cancelar compra sem carrinho	Mensagem pode ser melhorada.	1. DELETE /carrinhos cancelar compra sem carrinho2. Verificar resposta	**200** – "Não foi encontrado carrinho para esse usuário"	Mensagem clara: "Usuário não possui carrinho ativo"	Média