

# Relatório de Testes [ Challenge Compass ]

## Resumo







Os testes foram executados para validar as funcionalidades de **Usuários** (US 001), **Login** (US 002) e **Produtos** (US 003) da API ServeRest, cobrindo todos os 22 cenários planejados. Dos **22 cenários**:

- **US 001 (Usuários)**: 7 passaram, 2 falharam.
- **US 002 (Login)**: 4 passaram, sem falhas.
- **US 003 (Produtos)**: 8 passaram, 1 falharam (exclusão de produto, autenticação). Total: **19 passaram, 3 falharam**. Ao final deste documento estarão os detalhes, issues identificados e sugestões de melhoria.

## Resultados dos Testes

A tabela compara os resultados obtidos com os esperados, conforme o plano de testes (seção 6), com uma coluna para evidências (ex.: capturas do Postman, logs).

### US 001 - Usuários

Cenário 	Ação 	Resultado Esperado 	Resultado Obtido 	Status 	Evidências 

Criar usuário com dados válidos <a href="#">🔗</a>	<b>POST</b> /usuarios com e-mail único, senha válida, admin válido <a href="#">🔗</a>	201 – Usuário criado <a href="#">🔗</a>	201 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição e resposta no Postman</a> <a href="#">🔗</a>
Criar usuário com e-mail inválido <a href="#">🔗</a>	<b>POST</b> /usuarios com e-mail do Gmail/Hotmail ou malformatado <a href="#">🔗</a>	400 – Erro de validação <a href="#">🔗</a>	201 <a href="#">🔗</a>	Falhou <a href="#">🔗</a>	<a href="#">Requisição com e-mail inválido e resposta 201</a> <a href="#">🔗</a>
Criar usuário com e-mail duplicado <a href="#">🔗</a>	<b>POST</b> /usuarios com e-mail já existente <a href="#">🔗</a>	400 – Erro de duplicidade <a href="#">🔗</a>	400 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição com e-mail duplicado, resposta 400</a> <a href="#">🔗</a>
Atualizar usuário com ID inexistente <a href="#">🔗</a>	<b>POST</b> /usuarios/{id} com ID não existente <a href="#">🔗</a>	400 – Erro de duplicidade <a href="#">🔗</a>	201 <a href="#">🔗</a>	Falhou <a href="#">🔗</a>	<a href="#">Requisição PUT, resposta 201</a> <a href="#">🔗</a>
Consultar usuário inexistente <a href="#">🔗</a>	<b>GET</b> /usuarios/{id} com ID inválido <a href="#">🔗</a>	400 – Usuário não encontrado <a href="#">🔗</a>	400 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição GET, resposta 400</a> <a href="#">🔗</a>
Excluir usuário válido <a href="#">🔗</a>	<b>DELETE</b> /usuarios/{id} com ID existente <a href="#">🔗</a>	200 – Usuário excluído <a href="#">🔗</a>	200 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição DELETE, resposta 200 com mensagem de sucesso</a> <a href="#">🔗</a>
Testar senha no limite <a href="#">🔗</a>	<b>POST</b> /usuarios com senha de 5 ou 10 caracteres <a href="#">🔗</a>	201 – Usuário criado <a href="#">🔗</a>	201 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição com senha no limite, resposta 201</a> <a href="#">🔗</a>
Atualizar com e-mail duplicado <a href="#">🔗</a>	<b>PUT</b> /usuarios/{id} com e-mail já existente <a href="#">🔗</a>	400 – Erro de duplicidade <a href="#">🔗</a>	400 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição PUT com e-mail duplicado, resposta 400</a> <a href="#">🔗</a>
Excluir usuário inexistente <a href="#">🔗</a>	<b>DELETE</b> /usuarios/{id} com ID inválido <a href="#">🔗</a>	200 – Nenhum registro excluído <a href="#">🔗</a>	200 <a href="#">🔗</a>	Passou <a href="#">🔗</a>	<a href="#">Requisição DELETE com ID inválido, resposta 200</a> <a href="#">🔗</a>

## US 002 - Login [🔗](#)

Cenário	Ação	Resultado Esperado	Resultado Obtido	Status	Evidências
Login com credenciais válidas	<b>POST</b> /login com e-mail e senha corretos	200 – Token gerado	200	Passou	<a href="#">Requisição POST, resposta com token</a> <a href="#">🔗</a>
Login com senha inválida	<b>POST</b> /login com senha incorreta	401 – Erro de autenticação	401	Passou	<a href="#">Requisição POST, resposta 401</a> <a href="#">🔗</a>
Login com usuário não cadastrado	<b>POST</b> /login com e-mail inexistente	401 – Erro de autenticação	401	Passou	<a href="#">Requisição POST, resposta 401</a> <a href="#">🔗</a>
Acessar rota protegida com token expirado	<b>GET</b> /produtos com token após 10 minutos	401 – Erro de token inválido	401	Passou	<a href="#">Requisição GET /produtos, resposta 401</a> <a href="#">🔗</a>

## US 003 - Produtos [🔗](#)

Cenário	Ação	Resultado Esperado	Resultado Obtido	Status	Evidências
Criar produto com dados válidos	<b>POST</b> /produtos com token válido e nome único	201 – Produto criado	201	Passou	<a href="#">Requisição POST, resposta 201</a>
Criar produto com nome duplicado	<b>POST</b> /produtos com nome já existente	400 – Erro de duplicidade	400	Passou	<a href="#">Requisição POST, resposta 400</a>
Excluir produto vinculado a carrinho	<b>DELETE</b> /produtos/{id} com produto em carrinho	400 – Erro de dependência	400	Passou	<a href="#">Requisição DELETE, resposta 400</a>
Atualizar produto com ID inexistente	<b>PUT</b> /produtos/{id} com ID não existente	201 – Novo produto criado	201	Passou	<a href="#">Requisição PUT, resposta 201</a>
Acessar produtos sem autenticação	<b>GET</b> /produtos sem token	401 – Erro de autenticação	200	Falhou	<a href="#">Requisição GET, resposta 200</a>
Fluxo integrado usuário-produto-carrinho	<b>POST</b> /usuarios, <b>POST</b> /login, <b>POST</b> /produtos, adicionar ao carrinho	200/201 em cada etapa	201	Passou	<a href="#">Requisições da sequência, respostas 200/201</a>
Atualizar produto com nome duplicado	<b>PUT</b> /produtos/{id} com nome já existente	400 – Erro de duplicidade	400	Passou	<a href="#">Requisição PUT, resposta 400</a>
Criar produto com preço/quantidade inválidos	<b>POST</b> /produtos com preço ou quantidade $\leq 0$	400 – Erro de validação	400	Passou	<a href="#">Requisição POST, resposta 400</a>
Criar produto como não-admin	<b>POST</b> /produtos com token de usuário não-admin	403 – Erro de permissão	403	Passou	<a href="#">Requisição POST, resposta 403</a>

## Issues Identificados [↗](#)

Os cenários que falharam (5 de US 001, 2 de US 003) revelaram problemas críticos na API, com sugestões de melhoria. US 002 não apresentou falhas. Cada issue está registrada no Jira com um hyperlink para rastreamento.

### Issue #1: Validação fraca de e-mail (Cenário 2) [↗](#)

- **Descrição:** POST /usuarios aceitou e-mail de Gmail/Hotmail (ex.: [teste@gmail.com](mailto:teste@gmail.com)), retornando 201 em vez de 400.
- **Gravidade:** Alta (viola regra de negócio: e-mails não podem ser de Gmail/Hotmail).

- **Impacto:** Permite cadastros inválidos, comprometendo a integridade dos dados.
- **Sugestão de Melhoria:** Implementar validação no backend para rejeitar e-mails de provedores como Gmail/Hotmail, retornando 400 com mensagem clara (ex.: "E-mail de provedor inválido").
- **Jira:** [SRVREST-1](#)

#### ⚠ Issue #2: Status incorreto ao criar usuário com ID inexistente (Cenário 4) [🔗](#)

- **Descrição:** PUT /usuarios/{id} com ID inexistente retornou 200 em vez de 201, criando um novo usuário.
- **Gravidade:** Média (não segue padrão REST, mas funcionalidade opera).
- **Impacto:** Não conformidade com convenções REST, pode confundir integrações.
- **Sugestão de Melhoria:** Ajustar o endpoint para retornar 201 Created quando um novo usuário é criado, alinhando-se ao padrão REST.
- **Jira:** [SRVREST-2](#)

#### ⚠ Issue #6: Falha na autenticação de rotas protegidas (Cenário 18) [🔗](#)

- **Descrição:** GET /produtos sem token retornou 200 em vez de 401.
- **Gravidade:** Alta (viola regra de negócio: rotas protegidas exigem autenticação).
- **Impacto:** Compromete a segurança, permitindo acesso não autorizado.
- **Sugestão de Melhoria:** Adicionar verificação de token no endpoint GET /produtos, retornando 401 se o token estiver ausente ou inválido. Atualizar documentação Swagger para refletir mudança.
- **Jira:** [SRVREST-6](#)

---

#### [🔗](#) **Conclusão** [🔗](#)

Dos 22 cenários testados, 19 passaram, confirmando funcionalidades como criação de usuários e produtos com dados válidos, validação de duplicidade, autenticação robusta (US 002), e restrições de permissão. No entanto, 3 falhas indicam problemas críticos: validação fraca de e-mails, status inconsistentes, restrições não documentadas de autenticação. Os issues de alta gravidade comprometem a confiabilidade e segurança da API. Recomenda-se priorizar as correções, especialmente em validações de e-mail, autenticação e dependências com carrinhos, antes da liberação. Um plano de reteste deve ser executado após as correções, com foco nos cenários que falharam.

---