

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):** 4 passaram, 5 falharam (validação de e-mail, exclusão, status).
- **US 002 (Login):** 4 passaram, sem falhas.
- **US 003 (Produtos):** 7 passaram, 2 falharam (exclusão de produto, autenticação). Total: **15 passaram, 7 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 	POST /usuarios com e-mail único, senha válida, admin válido 	201 – Usuário criado 	201 	Passou 	 Requisição e resposta no Postman 
Criar usuário com e-mail inválido 	POST /usuarios com e-mail do Gmail/Hotmail ou malformatado 	400 – Erro de validação 	201 	Falhou 	 Requisição com e-mail inválido e resposta 201 
Criar usuário com e-mail duplicado 	POST /usuarios com e-mail já existente 	400 – Erro de duplicidade 	400 	Passou 	 Requisição com e-mail duplicado, resposta 400 
Atualizar usuário com ID inexistente 	POST /usuarios/{id} com ID não existente 	201 – Novo usuário criado 	200 	Falhou 	 Requisição PUT, resposta 200 
Consultar usuário inexistente 	GET /usuarios/{id} com ID inválido 	400 – Usuário não encontrado 	400 	Passou 	 Requisição GET, resposta 400 
Excluir usuário válido 	DELETE /usuarios/{id} com ID existente 	200 – Usuário excluído 	400 	Falhou 	 Requisição DELETE, resposta 400 com mensagem de erro 
Testar senha no limite 	POST /usuarios com senha de 5 ou 10 caracteres 	201 – Usuário criado 	201 	Passou 	 Requisição com senha no limite, resposta 201 
Atualizar com e-mail duplicado 	PUT /usuarios/{id} com e-mail já existente 	400 – Erro de duplicidade 	400 	Falhou 	 Requisição PUT com e-mail duplicado, resposta 400 

Excluir usuário inexistente ↗	DELETE /usuarios/{id} com ID inválido ↗	400 – Usuário não encontrado ↗	200 ↗	Falhou ↗	📎 Requisição DELETE com ID inválido, resposta 200 ↗
---	--	--	-----------------------	--------------------------	---

US 002 - Login [↗](#)

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

US 003 - Produtos [↗](#)

Cenário	Ação	Resultado Esperado	Resultado Obtido	Status	Evidências
Criar produto com dados válidos	POST /produtos com token válido e nome único	201 – Produto criado	201	Passou	📎 Requisição POST, resposta 201
Criar produto com nome duplicado	POST /produtos com nome já existente	400 – Erro de duplicidade	400	Passou	📎 Requisição POST, resposta 400
Excluir produto vinculado a carrinho	DELETE /produtos/{id} com produto em carrinho	400 – Erro de dependência	200	Falhou	📎 Requisição DELETE, resposta 200
Atualizar produto com ID inexistente	PUT /produtos/{id} com ID não existente	201 – Novo produto criado	201	Passou	📎 Requisição PUT, resposta 201
Acessar produtos sem autenticação	GET /produtos sem token	401 – Erro de autenticação	200	Falhou	📎 Requisição GET, resposta 200

Fluxo integrado usuário-produto-carrinho	POST /usuarios, POST /login, POST /produtos, adicionar ao carrinho	200/201 em cada etapa	201	Passou	Requisições da sequência, respostas 200/201
Atualizar produto com nome duplicado	PUT /produtos/{id} com nome já existente	400 – Erro de duplicidade	400	Passou	Requisição PUT, resposta 400
Criar produto com preço/quantidade inválidos	POST /produtos com preço ou quantidade ≤ 0	400 – Erro de validação	400	Passou	Requisição POST, resposta 400
Criar produto como não-admin	POST /produtos com token de usuário não-admin	403 – Erro de permissão	403	Passou	Requisição POST, resposta 403

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 hiperlink para rastreamento.

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

- **Descrição:** POST /usuarios aceitou e-mail de Gmail/Hotmail (ex.: 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 #3: Bloqueio de exclusão de usuário com carrinho (Cenário 6) [↗](#)

- **Descrição:** DELETE /usuarios/{id} retornou 400 para usuário válido, indicando que o usuário está vinculado a um carrinho.
- **Gravidade:** Alta (não documentado na US 001, impede exclusão legítima).
- **Impacto:** Usuários não podem ser excluídos se tiverem carrinhos, violando a expectativa de CRUD.
- **Sugestão de Melhoria:** Atualizar a documentação para explicitar a restrição de carrinhos na exclusão de usuários ou permitir exclusão com remoção automática de carrinhos associados.

- Jira: [SRVREST-3](#)

⚠ Issue #4: Status incorreto ao excluir usuário inexistente (Cenário 9) [↗](#)

- **Descrição:** DELETE /usuarios/{id} com ID inválido retornou 200 em vez de 400.
- **Gravidade:** Média (indica sucesso quando deveria falhar).
- **Impacto:** Pode levar a falsa percepção de exclusão bem-sucedida.
- **Sugestão de Melhoria:** Corrigir o endpoint para retornar 400 com mensagem de erro (ex.: "Usuário não encontrado") quando o ID é inválido.
- Jira: [SRVREST-4](#)

⚠ Issue #5: Exclusão indevida de produto vinculado a carrinho (Cenário 16) [↗](#)

- **Descrição:** DELETE /produtos/{id} com produto vinculado a carrinho retornou 200 em vez de 400.
- **Gravidade:** Alta (viola regra de negócio: produtos em carrinhos não podem ser excluídos).
- **Impacto:** Permite exclusão indevida, comprometendo a integridade dos carrinhos.
- **Sugestão de Melhoria:** Implementar validação no backend para verificar se o produto está em carrinhos antes da exclusão, retornando 400 com mensagem (ex.: "Produto vinculado a carrinho").
- Jira: [SRVREST-5](#)

⚠ 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.
- Jira: [SRVREST-6](#)

i Conclusão [↗](#)

Dos 22 cenários testados, 15 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, 7 falhas (5 em US 001, 2 em US 003) indicam problemas críticos: validação fraca de e-mails, status inconsistentes, restrições não documentadas na exclusão de usuários, e falhas graves em exclusão de produtos e autenticação. Os issues de alta gravidade (cenários 2, 6, 16, 18) 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.
