Relátorio de Testes [Challenge Compass]

Resumo

Os testes foram executados para validar as funcionalidades de **Usuários** (US 001), **Login** (US 002), **Produtos** (US 003) e **Carrinho** (US004 - Opcional) da API ServeRest, cobrindo todos os 27 cenários planejados. Dos **27 cenários**:

- **US 001 (Usuários)**: 7 passaram, 2 falharam.
- **US 002 (Login)**: 4 passaram, sem falhas.
- **US 003 (Produtos)**: 7 passaram, 1 falharam (exclusão de produto, autenticação). Total: **18 passaram**, **3 falharam**. Ao final deste documento estarão os detalhes, issues identificados e sugestões de melhoria.
- **US 004 (Carrinhos):** 6 foram testados, 5 passaram, 1 falhou.

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 malformado	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	400 – Erro de duplicidade	201	Falhou	📎 Requisição PUT, resposta 201
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	200	Passou	📎 Requisição DELETE, resposta 200 com mensagem de sucesso
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	Passou	📎 Requisição PUT com e-mail duplicado, resposta 400
Excluir usuário inexistente	DELETE /usuarios/{id} com ID inválido	200 – Nenhum registro excluido	200	Passou	📎 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	400	Passou	Requisição DELETE, resposta 400
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 /fluxo continuo	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

US 004 - Carrinho

Cenário	Ação	Resultado Esperado	Resultado Obtido	Status	Evidências
Listar todos os carrinhos	GET /carrinhos	200 – Lista de carrinhos retornada	200	Passou	Requisição GET, resposta 200
Criar carrinho com produtos válidos	POST /carrinhos com token válido e produto existente	201 – Carrinho criado	201	Passou	Requisição POST, resposta 201

Criar carrinho sem token de autorização	POST /carrinhos sem token	400 – Erro de dependência	404	Falhou	📎 Requisição POST, resposta 404
Criar carrinho com produto inexistente	POST /carrinhos com ID de produto inválido	400 – Erro de produto inválido	400	Passou	📎 Requisição POST, resposta 400
Concluir compra do carrinho	DELETE /carrinhos/concluir-compra com token válido	200 – Compra concluída, carrinho removido	200	Passou	📎Requisições DELETE, resposta 200
Cancelar compra do carrinho	DELETE /carrinhos/cancelar-compra com token válido	200 – Compra cancelada, carrinho mantido	200	Passou	📎 Requisições DELETE, respostas 200

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).

Issues Identificados

Os cenários que falharam (2 de US 001, 1 de US 003 e 1 de US 004) 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: POST /usuarios aceita e-mails de provedores não permitidos

- **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 HTTP incorreto ao criar novo usuário com PUT e ID inválido

- **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: Acesso indevido ao endpoint GET /produtos sem token

- **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

⚠ Issue #4: Criar carrinho sem token de autorização

- **Descrição:** POST /carrinhos com produto inexistente retornou 404 em vez de 400.
- **Gravidade:** Média (inconsistência na resposta esperada, afeta validação).
- **Impacto:** Confunde o consumidor da API, que espera um erro 400 para validações de dados, não 404 (recurso não encontrado).
- **Sugestão de Melhoria:** Ajustar o endpoint POST /carrinhos para retornar 400 quando o produto for inexistente, garantindo consistência com a regra de negócio. Revisar o tratamento de erros e atualizar a documentação Swagger para esclarecer o comportamento.
- **Jira:** SRVREST-7

ℹ Conclusão

Dos 27 cenários testados, 23 passaram, confirmando funcionalidades como criação de usuários e produtos com dados válidos, validação de duplicidade, autenticação robusta, e restrições de permissão. No entanto, 4 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.
