Plano de Testes – API ServeRest

ÍNDICE

Apresentação

Este plano tem como objetivo organizar e guiar as atividades de garantia de qualidade da API **ServeRest**, com foco na cobertura funcional das rotas disponíveis no Swagger, validação de regras de negócio descritas nas User Stories, além de testes exploratórios e fluxos ponta a ponta, simulando o uso real de um e-commerce.

> **Responsável**: @Anna Santoro
> **Data de criação**: 6 de mai. de 2025
> **Última atualização**: 9 de mai. de 2025
> **Sprint:** Challenge Final
> **Ferramentas:** Postman, Swagger, Confluence, Jira

Objetivo

Garantir que a API ServeRest:

- Atenda aos critérios de aceitação estabelecidos nas User Stories.
- Siga comportamentos RESTful coerentes com o Swagger.
- Esteja funcional e segura para os fluxos principais do negócio.
- Tenha comportamentos esperados diante de cenários negativos, limites e alternativos.
  - Priorização de cenários críticos ao negócio, como login e finalização de compra.

Escopo

- Validação funcional de:
  - `/usuarios`
  - `/login`
  - `/produtos`
  - `/carrinhos`
- Testes manuais
- Testes exploratórios no endpoint de Carrinho
- Testes automatizados via Postman (candidatos definidos)
- Matriz de Rastreabilidade
- Matriz de Risco

Ambiente de Testes

Item	Valor
SO	Windows 11

Navegador	Microsoft Edge 135 (64 bits)
Ferramentas	Postman, Swagger, XMind, Confluence, Jira
API	ServeRest
Ambiente	Público (homologação)
Tipo de Teste	Funcional, Exploratórios e Automatizados
Período de Execução	9 de mai. de 2025

Análise Inicial

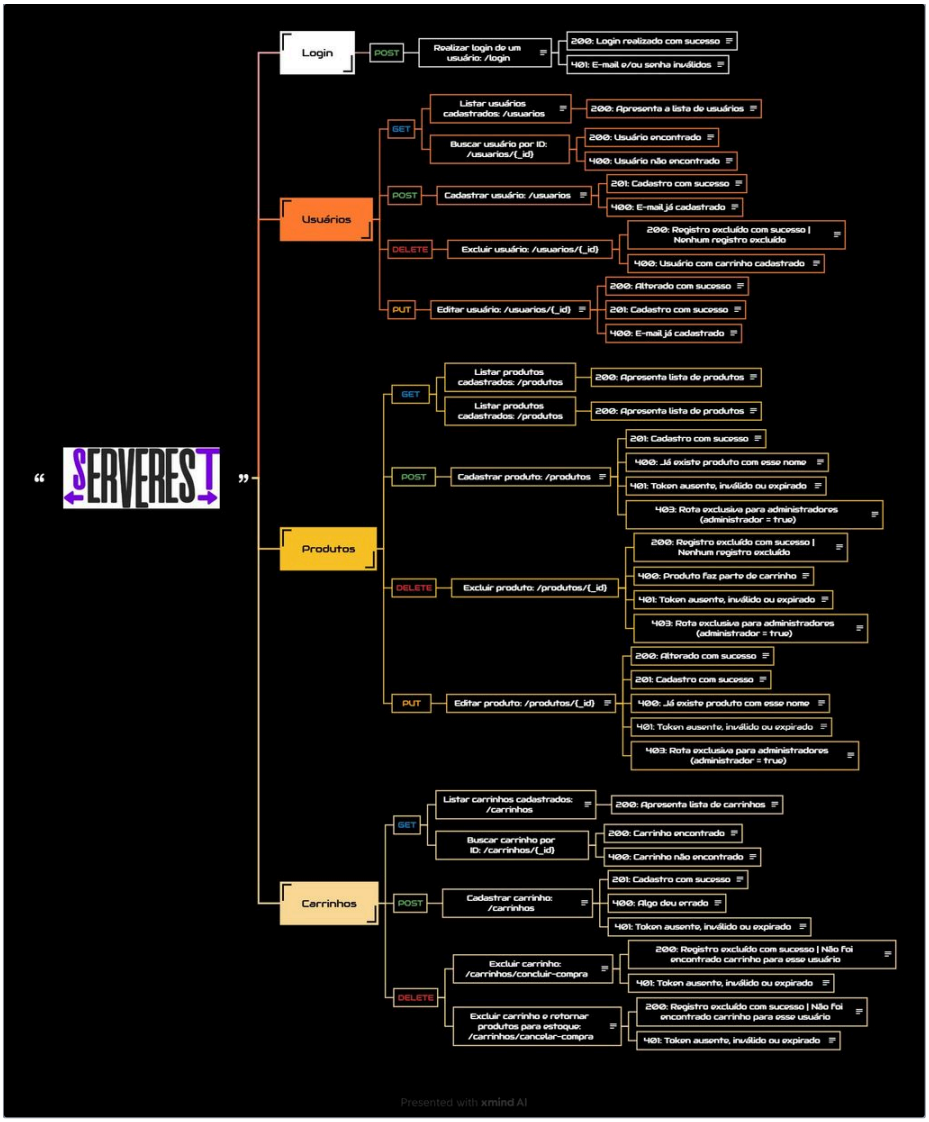
Documentos utilizados e anexos:

- **Swagger oficial da API ServeRest**
- **User Stories:**
  - **US001 – Usuários**
  - **US002 – Login**
  - **US003 – Produtos**
  - **US004 - Carrinho**
- **Mapa Mental**
- **Collection Postman**
- **Relatório de Testes**
- **Issues**

Técnicas de Teste Aplicadas	Tipo de Teste	Justificativa
CRUD	**Funcional**	Base para validar todos os métodos HTTP: `POST`, `GET`, `PUT`, `DELETE`
Teste End-to-End (E2E)	**Funcional**	Simula o fluxo real do usuário: cadastro → login → criação de produto → carrinho
Particionamento de Equivalência	**Técnicas de Caixa Preta**	Valida entradas em grupos: e-mails válidos/ inválidos, senhas corretas/ incorretas
Tabela de Decisão	**Técnicas de Caixa Preta**	Avaliação de múltiplas combinações de condições e regras em campos como `PUT /usuarios`, `carrinhos` com estoque
Análise de Valor Limite	**Técnicas de Caixa Preta**	Testa extremos de campos como senha (mín. 5, máx. 10), quantidade, preço
Goldilocks	**Heurística Exploratória**	Utilizada nos testes exploratórios no carrinho para validar cenários mínimos, ideais e excessivos
Automatização via Postman	**Funcional**	Aplicada em endpoints críticos e repetitivos (ex: login, criação de produto) para reduzir esforço manual

ID	Impedimentos Identificados	Tipo	Criticidade
CT-008	Comportamento PUT no Swgger contradiz Acceptance Criteria da user story 001	Ambiguidade	ALTA
CT-010	A response de `/DELETE` de usuário inexistente realmente deveria ser `200` `"Nenhum registro excluído"` ?	Dúvida	BAIXA
CT-036	Não é especificado se preço = zero é válido.	Dúvida	ALTA
CT-039	Não há especificação se a API deve ignorar ou rejeitar campos extras.	Dúvida	ALTA
BUG-01	Violação a regra de negócios 005 - Domínio restrito hotmail é validado	Bug	ALTA
BUG-02	Violação da regra de negócios 007 - Senhas < 5 e > 10 caracteres são aceitas	Bug	ALTA

Mapa Mental da Aplicação (Clique Aqui)



Cenários de Teste

ID	Título	Pré-condição	Passos	Resultado Esperado	Automação
CT-001	Cadastro válido de novo usuário	E-mail ainda não cadastrado	POST `/usuarios` com nome, e-mail válido, senha entre 5-10 caracteres, administrador = "true"	201 Created + Mensagem: "Cadastro realizado com sucesso" + ID retornado	SIM
CT-002	Cadastro com e-mail duplicado	E-mail já existe na base	POST `/usuarios` com e-mail já cadastrado	400 Bad Request + "Este email já está sendo usado"	SIM
CT-003	Cadastro com e-mail inválido	—	POST `/usuarios` com e-mail mal formatado (`joa@val*d.com`)	400 + erro de validação	NÃO
CT-004	Cadastro com domínio Gmail	—	POST `/usuarios` com e-mail `joao@gmail.com`	400 + Mensagem bloqueando domínios Gmail/Hotmail	NÃO
CT-005	Cadastro com senha muito curta	—	POST `/usuarios` com senha de 3 caracteres	400 + Mensagem: "A senha deve conter entre 5 e 10 caracteres"	SIM
CT-006	Cadastro com senha muito longa	—	POST `/usuarios` com senha de 11 caracteres	400 + Mensagem: "A senha deve	SIM

				conter entre 5 e 10 caracteres"	
CT-007	GET de usuário inexistente	ID não existente	GET `/usuarios/{id-invalido}`	400 ou 404 + Mensagem: "Usuário não encontrado"	NÃO
CT-008	PUT criando usuário com ID inexistente	Nenhum usuário com ID fornecido	PUT `/usuarios/{id-invalido}` com payload de novo usuário	⚠️ Ambiguidade	NÃO
CT-009	PUT com e-mail já utilizado	Dois usuários com e-mails diferentes	PUT `/usuarios/{id}` tentando alterar e-mail para um já existente	400 + "Este email já está sendo usado"	NÃO
CT-010	DELETE de usuário inexistente	ID inválido	DELETE `/usuarios/{id-invalido}`	⚠️ Em análise	NÃO
CT-011	GET de todos os usuários	Existência de usuários na base	GET `/usuarios`	200 OK + Array de usuários	NÃO
CT-012	Cadastro com campo obrigatório ausente	—	POST `/usuarios` sem campo "email"	400 + erro por campo obrigatório faltando	SIM
CT-013	Valor inválido no campo “administrador”	—	POST `/usuarios` com `"administrador": "talvez"`	400 + mensagem de tipo inválido	NÃO
CT-014	Criação de usuário com nome vazio	—	POST `/usuarios` com `"nome": ""`	400 + "Nome é obrigatório"	NÃO
CT-015	Login com credenciais válidas	Usuário válido cadastrado (ex: `joao@valido.com`, senha `123456`)	POST `/login` com payload válido	200 OK + Mensagem `"Login realizado com sucesso"` + Token Bearer retornado	SIM
CT-016	Login com usuário não cadastrado	Usuário não existe no sistema	POST `/login` com e-mail inexistente	401 Unauthorized + Mensagem `"Email e/ou senha inválidos"`	SIM
CT-017	Login com senha inválida	E-mail válido cadastrado	POST `/login` com senha errada	401 Unauthorized + Mensagem `"Email e/ou senha inválidos"`	SIM
CT-018	Login com e-mail mal formatado	—	POST `/login` com e-mail `"joaogmail.com"`	400 Bad Request ou erro de validação	NÃO
CT-019	Login sem fornecer senha	—	POST `/login` com campo `"password"` vazio ou ausente	400 + Mensagem sobre campo obrigatório	SIM
CT-020	Login sem fornecer e-mail	—	POST `/login` com campo `"email"` vazio ou ausente	400 + Mensagem sobre campo obrigatório	SIM
CT-021	Login com campo extra no payload	—	POST `/login` com campos adicionais no JSON (ex: `"admin": true`)	200 OK se campos extras forem ignorados, ou 400 se for validado	NÃO
CT-022	Verificação de validade do token (duração 10 min)	Usuário logado, token emitido	Esperar 10 minutos e usar o token em endpoint protegido (ex: `/produtos`)	401 Unauthorized após 10 min, indicando expiração do token	SIM
CT-023	Login com token reutilizado após logout	Token válido emitido e logout realizado (se aplicável)	Reutilizar token antigo após logout ou tempo excedido	401 Unauthorized	NÃO
CT-024	Login com e-mail com letras maiúsculas	Usuário cadastrado com letras minúsculas	POST `/login` com `"JOAO@VALIDO.COM"`	401 Unauthorized ou 200 OK dependendo da case-sensitivity	NÃO

CT-025	Cadastro de produto com dados válidos	Usuário autenticado (token válido)	POST `/produtos` com payload: `{ "nome": "Camiseta Tech", "preco": 99, "descricao": "Camiseta com estampa tecnológica", "quantidade": 10 }`	201 Created + "Cadastro realizado com sucesso" + ID retornado	SIM
CT-026	Cadastro sem autenticação	Nenhum token fornecido ou token inválido	POST `/produtos` com payload válido, sem header Authorization	401 Unauthorized + mensagem de autenticação obrigatória	SIM
CT-027	Cadastro com nome duplicado	Produto "Camiseta Tech" já cadastrado	POST `/produtos` com nome já utilizado	400 Bad Request + "Já existe produto com esse nome"	SIM
CT-028	Edição com dados válidos	Produto cadastrado + usuário autenticado	PUT `/produtos/{id}` com alterações válidas no payload	200 OK + "Registro alterado com sucesso"	SIM
CT-029	PUT com ID inexistente (criação implícita)	Nenhum produto com ID `xyz123`	PUT `/produtos/xyz123` com payload válido	200 ou 201 + mensagem coerente indicando criação	SIM
CT-030	PUT com nome duplicado	Dois produtos com nomes diferentes	Atualizar produto alterando seu nome para o nome de outro produto existente	400 Bad Request + "Já existe produto com esse nome"	SIM
CT-031	Deletar produto sem vínculo	Produto não está em carrinho	DELETE `/produtos/{id}` com token válido	200 OK + "Registro excluído com sucesso"	SIM
CT-032	Deletar produto vinculado a carrinho	Produto em um carrinho ativo	DELETE `/produtos/{id}`	400 ou 403 + "Não é permitido excluir produto vinculado a um carrinho"	SIM
CT-033	Listar todos os produtos	Existem produtos cadastrados	GET `/produtos`	200 OK + lista de produtos	NÃO
CT-034	Criar produto com campo obrigatório ausente	—	POST `/produtos` sem o campo `"quantidade"`	400 Bad Request + mensagem de campo obrigatório ausente	SIM
CT-035	Criar produto com quantidade negativa	—	POST `/produtos` com `"quantidade": -5`	400 Bad Request + mensagem de validação coerente	SIM
CT-036	Criar produto com preço zero	—	POST `/produtos` com `"preco": 0`	⚠️ Não determinado	SIM (Em análise)
CT-037	Criar produto com nome em branco	—	POST `/produtos` com `"nome": ""`	400 Bad Request + mensagem de nome obrigatório	SIM
CT-038	Listar produtos com token inválido	Token inválido no header	GET `/produtos` com token inválido	401 Unauthorized + mensagem coerente	NÃO
CT-039	PUT com campos extras inesperados	Produto existente	PUT `/produtos/{id}` com campos irrelevantes no payload (ex: `"categoria": "techwear"`)	⚠️ Não determinado	SIM (Em análise)
CT-040	Criar carrinho com produto válido e quantidade adequada	Usuário autenticado + produto cadastrado	POST `/carrinhos` com payload válido, ex: `{ "produtos": [{ "idProduto": "abc123", "quantidade": 2}`	201 Created + "Cadastro realizado com sucesso" + ID do carrinho	SIM
CT-041	Criar carrinho com quantidade zero	Usuário autenticado + produto cadastrado	POST `/carrinhos` com `"quantidade": 0`	400 Bad Request + mensagem coerente sobre quantidade inválida	SIM
CT-042	Criar carrinho com quantidade acima do estoque	Produto com estoque limitado (ex: 5 unidades)	POST `/carrinhos` com `"quantidade": 9999`	400 Bad Request + mensagem de estoque insuficiente	SIM

CT-043	Criar carrinho com produto inexistente	Usuário autenticado	POST `/carrinhos` com `idProduto` inválido	400 ou 404 + mensagem de erro coerente	SIM
CT-044	Buscar carrinho por ID válido	Carrinho criado com sucesso	GET `/carrinhos/{id}`	200 OK + dados do carrinho com produtos inseridos	SIM
CT-045	Buscar carrinho por ID inexistente	ID inválido ou carrinho excluído	GET `/carrinhos/{id-invalido}`	404 Not Found + mensagem: "Carrinho não encontrado"	NÃO
CT-046	Deletar carrinho existente	Carrinho criado previamente com produtos	DELETE `/carrinhos/{id}`	200 OK + "Registro excluído com sucesso"	NÃO
CT-047	Cancelar compra e devolver ao estoque	Carrinho criado e estoque já reduzido	DELETE `/carrinhos/cancelar-compra/{id}`	200 OK + "Registro excluído com sucesso. Estoque dos produtos restaurado"	SIM
CT-048	Criar carrinho sem autenticação	—	POST `/carrinhos` sem header Authorization	401 Unauthorized + "Token de acesso ausente ou inválido"	SIM
CT-049	Criar carrinho com payload malformado	—	POST `/carrinhos` com JSON inválido ou campos faltantes	400 Bad Request + mensagem de erro de estrutura	SIM
CT-050	Deletar carrinho inexistente	ID inválido	DELETE `/carrinhos/{id-invalido}`	404 Not Found + mensagem: "Carrinho não encontrado"	NÃO

Observação: **CT-036** e **CT-039** dependem de clareza nas regras de negócio (possível automação condicional após resolução).

Matriz de Risco

ID	Cenário	Impacto	Probabilidade	Risco	Ação de Mitigação
CT-008	PUT criando usuário com ID inexistente (ambíguo)	Crítico	Média	Alto	Issue no Jira para definição de regra de negócio antes dos testes
CT-015	Login com credenciais válidas	Alto	Baixa	Crítico	Monitorar execução automática no CI; testes de fumaça diários
CT-016	Login com usuário não cadastrado	Médio	Média	Médio	Automatizar verificação de erro `401` em cada build
CT-022	Validade do token (10 min)	Crítico	Baixa	Alto	Script no Postman simulando expiração; documentação clara do processo
CT-036	Criar produto com preço zero (indefinido)	Alto	Média	Crítico	Issue no Jira; bloquear testes até retorno do PO
CT-039	PUT com campos extras inesperados	Alto	Média	Crítico	Issue no Jira; padronizar payloads antes da automação
CT-040	Criar carrinho com produto válido	Alto	Baixa	Crítico	Automatizar fluxo E2E no Postman; fixture de produto pré-criado
CT-042	Criar carrinho com quantidade acima do estoque	Médio	Média	Médio	Automatizar teste de estoque insuficiente; mock de ambiente
CT-047	Cancelar compra e restaurar estoque	Crítico	Média	Alto	Automatizar verificação de rollback de estoque

CT-050	Deletar carrinho inexistente	Baixo	Baixa	Baixo	Teste manual pontual; não prioritário para automação

---

Rastreabilidade e Visão de Cobertura

Matriz de Rastreabilidade

US	ID Req.	Requisito	Cenários de Teste	Prioridade	Criticidade	Status
US001	**REQ-001**	Usuário deve possuir os campos nome, email, senha, administrador	CT-001, CT-012, CT-014, CT-013	**Alta**	**Crítica**	EXECUTADO
US001	**REQ-002**	Não deve permitir ações em usuários inexistentes	CT-007, CT-010	**Média**	**Alta**	EXECUTADO
US001	**REQ-003**	Não deve permitir cadastro com e-mail duplicado (POST e PUT)	CT-002, CT-009	**Alta**	**Crítica**	EXECUTADO
US001	**REQ-004**	PUT com ID inexistente deve criar novo usuário	CT-008	**Alta**	**Crítica (ambígua)**	BLOQUEADO
US001	**REQ-005**	Bloquear e-mails de domínios gmail/hotmail	CT-004	**Alta**	**Alta**	EXECUTADO
US001	**REQ-006**	Validar formato de e-mail	CT-003	**Alta**	**Alta**	EXECUTADO
US001	**REQ-007**	Senhas devem ter entre 5 e 10 caracteres	CT-005, CT-006	**Alta**	**Alta**	EXECUTADO
US002	**REQ-008**	Usuários não cadastrados não deverão conseguir autenticar	CT-016	**Alta**	**Alta**	NÃO EXECUT...
US002	**REQ-009**	Usuários com senha inválida não deverão conseguir autenticar	CT-017	**Alta**	**Alta**	EXECUTADO
US002	**REQ-010**	Usuários existentes e com senha correta deverão ser autenticados	CT-015	**Alta**	**Alta**	EXECUTADO
US002	**REQ-011**	A autenticação deverá gerar um token Bearer	CT-015	**Alta**	**Alta**	EXECUTADO
US002	**REQ-012**	A duração da validade do token deverá ser de 10 minutos	CT-022	**Alta**	**Crítica**	NÃO EXECUT...
US002	**REQ-013**	Campos obrigatórios (`email` e `password`) devem ser validados	CT-019, CT-020	**Alta**	**Alta**	NÃO EXECUT...
US002	**REQ-014**	Formato inválido de e-mail deve ser rejeitado	CT-018	**Média**	**Média**	EXECUTADO
US003	**REQ-015**	Usuários não autenticados não devem conseguir realizar ações na rota de Produtos	CT-026	**Alta**	**Alta**	EXECUTADO
US003	**REQ-016**	Não deve ser possível cadastrar produto com nome já utilizado	CT-027, CT-030	**Alta**	**Alta**	NÃO EXECUT...
US003	**REQ-017**	Caso não exista produto com o ID informado no PUT, um novo produto deverá ser criado	CT-029	**Média**	**Média**	NÃO EXECUT...
US003	**REQ-018**	Produtos criados através do PUT não poderão ter nomes previamente cadastrados	CT-030	**Alta**	**Alta**	NÃO EXECUT...

US003	**REQ-019**	Produtos com campos inválidos não devem ser aceitos (preço zero, nome em branco, etc.)	CT-034, CT-035, CT-036, CT-037	**Alta**	**Alta**	NÃO EXECUT...
US003	**REQ-020**	Não deve ser possível excluir produtos que estão dentro de carrinhos (dependência Carrinhos)	CT-032	**Alta**	**Crítica**	NÃO EXECUT...
US003	**REQ-021**	Exclusão de produto não vinculado deve retornar sucesso	CT-031	**Alta**	**Alta**	NÃO EXECUT...
US003	**REQ-022**	Listagem de produtos deve retornar todos os itens cadastrados	CT-033	**Média**	**Média**	NÃO EXECUT...
US003	**REQ-023**	PUT com campos adicionais no payload deve ser tratado corretamente	CT-039	**Média**	**Média**	NÃO EXECUT...

Cobertura de Requisitos

- Total de Requisitos: **23**
- Cobertos (com testes executados): **16 (7 restantes)**
- Cobertura Geral: **47,8%**

Requisitos Críticos Testados

- Requisitos com Criticidade Alta ou Crítica: **19**
- Testados (com retorno sucesso): **9**
- Requisitos Críticos Cobertos: **47,3%**

- Total de cenários indicados para automação: 30
- Sugerido uso do Postman + Tests Tab + Newman Runner
