

Plano de Testes: API ServeRest

Testador: Guilherme Hepp da Fonseca

API: Plataforma ServeRest/Swagger do ServeRest

1. Apresentação

Este documento detalha a estratégia e o planejamento de testes para a API REST da plataforma ServeRest. O plano abrange os módulos de **Usuários**, **Autenticação**, **Produtos** e **Carrinhos**, focando em testes funcionais, de integração e de segurança a nível de API.

2. Objetivo

O principal objetivo deste plano de testes é validar a implementação da API ServeRest, garantindo sua funcionalidade, confiabilidade e segurança.

Objetivos Específicos:

- Verificar se todos os endpoints da API se comportam conforme a documentação e os critérios de aceitação das User Stories (US 001, 002, 003).
- Identificar e documentar defeitos, inconsistências e vulnerabilidades de segurança.
- Garantir a correta integração entre os diferentes módulos (ex: um produto em um carrinho não pode ser excluído).
- Validar as regras de negócio, como restrições de acesso (usuários comuns vs. administradores) e unicidade de registros (e-mail de usuário, nome de produto).
- Servir como base para a criação e manutenção de uma suíte de testes automatizados.

3. Escopo

Dentro do Escopo:

- Testes funcionais de todos os endpoints das APIs de [/usuarios](#), [/login](#), [/produtos](#) e [/carrinhos](#).
- Validação de todas as operações CRUD (Create, Read, Update, Delete) para cada módulo.
- Testes de "caminho feliz" (fluxos de sucesso) e "caminho triste" (fluxos de exceção e erro).
- Validação das regras de autenticação e autorização (Bearer Token, rotas exclusivas para administradores).
- Testes de integração entre os módulos (ex: dependência entre Carrinhos e Produtos).

4. Análise

A estratégia de teste foi definida a partir da análise dos seguintes artefatos:

- **User Stories (US 001, 002, 003):** Forneceram o contexto de negócio e a perspectiva do usuário.
- **CrITÉrios de Aceitação:** Detalharam as regras de negócio e os resultados esperados para cada funcionalidade.
- **Mapa Mental da Aplicação (Swagger/Documentação):** Serviu como a documentação técnica principal, detalhando cada endpoint, método, payload esperado e códigos de resposta.

A análise revelou pontos críticos a serem testados, como a necessidade de diferentes perfis de usuário (admin e comum), a validação de unicidade de dados e as dependências lógicas que impedem a exclusão de recursos associados a outros.

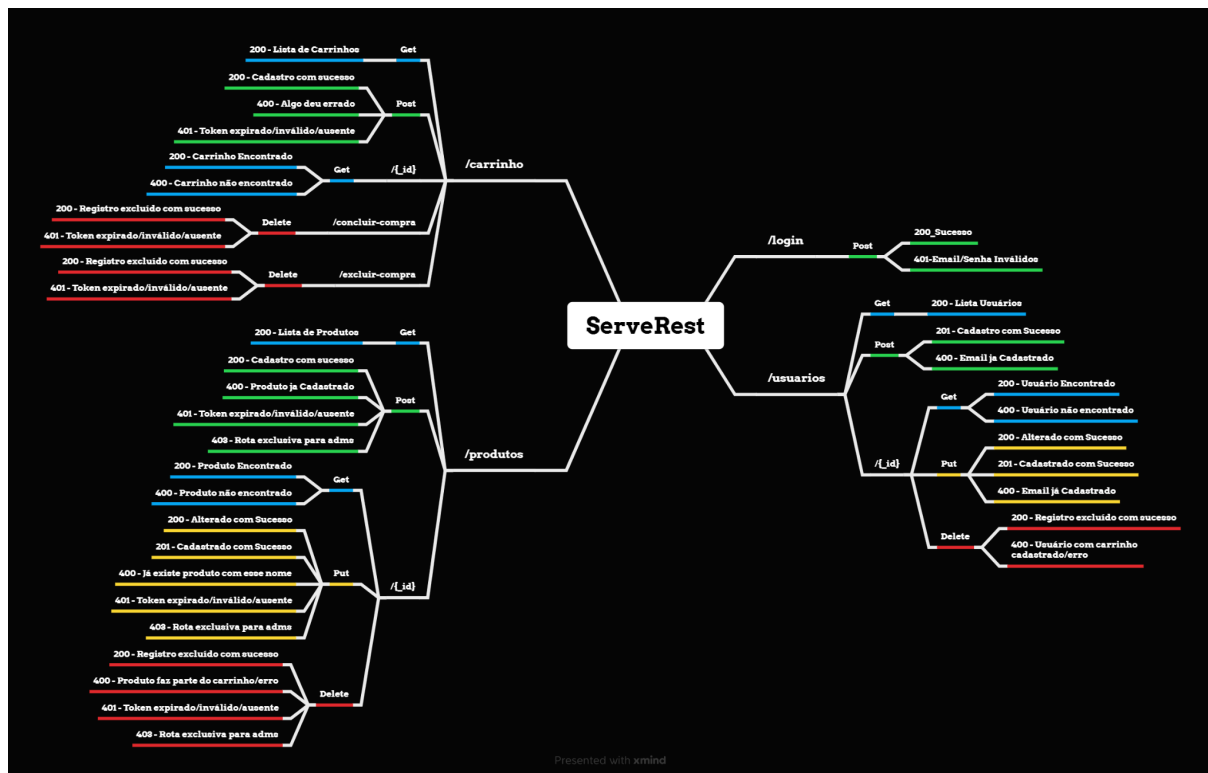
5. Técnicas Aplicadas

Para garantir uma cobertura de testes abrangente, as seguintes técnicas serão aplicadas:

- **Testes Funcionais:** Validação de cada funcionalidade contra seus requisitos.
- **Particionamento de Equivalência:** Agrupamento de dados de entrada em classes de equivalência (ex: e-mails válidos, e-mails inválidos, e-mails de provedores proibidos).
- **Análise de Valor Limite:** Verificação de regras com limites (ex: tamanho mínimo e máximo de senhas).
- **Testes Baseados em Cenários:** Criação de fluxos de ponta a ponta que simulam o uso real da API (ex: admin cria produto -> usuário cria carrinho com o produto -> admin tenta deletar o produto).
- **Testes de Segurança (Básicos):** Verificação de acesso não autorizado a rotas protegidas e análise de respostas para evitar vazamento de dados sensíveis.

6. Mapa Mental da Aplicação

O mapa mental feito no XMind serviu como um guia visual para a estrutura da API. Ele mapeia claramente cada recurso ([/produtos](#), [/carrinhos](#), [/usuario](#), [/login](#)), seus métodos HTTP ([GET](#), [POST](#), [PUT](#), [DELETE](#)) e os códigos de resposta esperados para sucesso e falha. Este mapa foi a base para a criação dos cenários de teste, garantindo que cada rota e cada resultado documentado fossem cobertos.



7. Cenários de Teste Planejados

A seguir, uma lista dos principais cenários de teste planejados:

- **CT-USUÁRIO-01:** Criar com sucesso usuários do tipo "admin" e "comum".
- **CT-USUÁRIO-02:** Falha ao tentar criar usuário com e-mail já existente.
- **CT-USUÁRIO-03:** Editar/Deletar usuário através do ID com seu ID específico.
- **CT-LOGIN-01:** Sucesso no login de admin e comum, com geração de token válido.
- **CT-LOGIN-02:** Falha no login com e-mail inexistente/ senha inválida (retorno 401) ou campo vazio(retorno 400).
- **CT-PRODUTO-01:** Admin cadastra, busca, edita e exclui um produto com sucesso (ciclo CRUD).
- **CT-PRODUTO-02:** Falha ao tentar cadastrar produto com nome duplicado.
- **CT-PRODUTO-03:** Usuário comum falha ao tentar acessar rotas de gerenciamento de produtos (retorno 403).
- **CT-CARRINHO-01:** Usuário comum cria um carrinho, adiciona produtos e conclui a compra com sucesso.
- **CT-USUÁRIO/CARRINHO/PRODUTO-01:** Falha ao tentar excluir um usuário que está associado a um carrinho que contém um produto.

8. Priorização da Execução dos Cenários de Teste

Os cenários serão executados em ordem de prioridade para otimizar a detecção de defeitos críticos.

- **P1 - Crítico:** Fluxos essenciais que, se falharem, impedem o uso básico da aplicação.

- Criar usuário (admin e comum).
- Login (admin e comum).
- Admin cadastrar um novo produto.
- Usuário comum criar um carrinho.
- **P2 - Alto:** Testes das principais funcionalidades e regras de negócio.
 - Ciclo CRUD completo para todos os módulos.
 - Validação de erros comuns (dados duplicados, itens não encontrados).
 - Validação de permissões (admin vs. comum).
- **P3 - Médio:** Testes de cenários alternativos e regras de negócio menos comuns.
 - Validação de todos os campos de cada requisição (ex: formato de e-mail, tamanho de senha).

9. Matriz de Risco

Risco Identificado	Probabilidade	Impacto	Estratégia de Mitigação
Acesso não autorizado a rotas de admin	Média	Alto	Testes exaustivos de permissão em todos os endpoints de produtos, usando tokens de admin e de usuário comum.
Inconsistência de dados (duplicados)	Média	Médio	Cenários de teste focados em violar as regras de unicidade de e-mail e nome de produto.
Exclusão de dados com dependências	Baixa	Alto	Testes de integração específicos para os cenários de exclusão de produto em carrinho e usuário com carrinho.
Vazamento de dados sensíveis em respostas	Alta	Alto	Testes específicos para verificar a ausência de campos como password em endpoints de listagem (GET /usuarios).

10. Cobertura de Testes

O objetivo é atingir 100% de cobertura dos requisitos e endpoints documentados.

- **Cobertura de Requisitos:** Cada Critério de Aceitação de todas as User Stories será mapeado a pelo menos um cenário de teste.
- **Cobertura de Endpoints:** Todos os endpoints e métodos HTTP definidos no mapa mental serão testados.
- **Cobertura de Cenários:** Serão cobertos tanto os fluxos de sucesso ("caminho feliz") quanto os de falha ("caminho triste") para cada funcionalidade.

11. Testes Candidatos a Automação

Dada a natureza da API, **100% dos testes funcionais planejados são candidatos à automação**. A automação será crucial para garantir a regressão contínua da aplicação.

- **Ferramenta Proposta:** Postman, com uso do Collection Runner.
- **Prioridade de Automação:**
 - **Alta (Suíte de Smoke/Regressão):** Os cenários de prioridade P1. Devem rodar a cada novo build ou deploy.
 - **Média (Suíte de Regressão Completa):** Os cenários de prioridade P2. Podem rodar diariamente.
 - **Baixa:** Os cenários de prioridade P3, que podem ser adicionados progressivamente à suíte de regressão.