Plano de Teste

1. Apresentação

API: ServeRest 2.29.7

Este plano de testes detalha a estratégia e as atividades de garantia da qualidade para a API REST ServeRest 2.29.7, que simula um e-commerce. O objetivo é validar as regras de negócio e o funcionamento das APIs, assegurando que atendam aos requisitos e mantenham a integridade do sistema

2. Objetivo

Garantir a qualidade da aplicação, validando as regras de negócio e o funcionamento das APIs REST, de modo que atendam aos requisitos especificados e mantenham a integridade.

3. Escopo

Os testes contemplarão as funcionalidades essenciais da aplicação expostas pelas APIs REST, com foco nos fluxos críticos para o negócio.

Recursos a serem Testados:

1. **/login**
    a. POST
2. **/usuários**
    a. GET
    b. POST
    c. PUT
    d. DELETE
3. **/produtos**
    a. GET
    b. POST
    c. PUT
    d. DELETE
4. **/carrinhos**
    a. GET
    b. POST
    c. DELETE

4. Análise (produto, riscos, requisitos)

4.1. Produto

A ServeRest é uma API REST que simula um e-commerce. Ela permite o cadastro e a autenticação de vendedores e clientes, além de operações com produtos e carrinhos

4.2. Foco dos Testes

O foco principal é garantir que as regras de negócio sejam validadas corretamente.

4.3. Análise de Requisitos (User Stories)

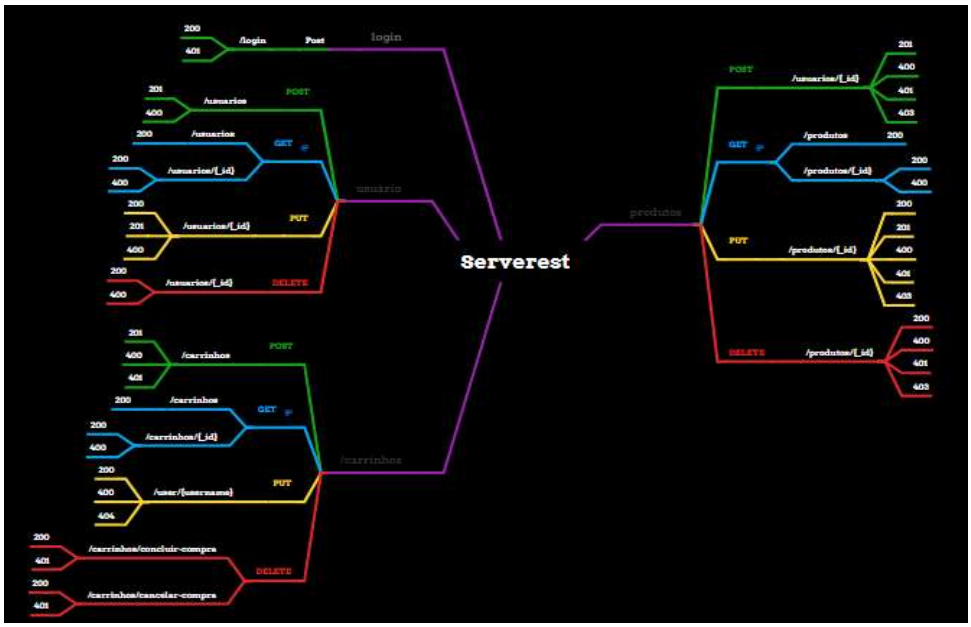
Com base nas User Stories fornecidas, os testes devem cobrir os seguintes requisitos:

- **US 001 - Usuários**: O cadastro de usuários deve conter os campos NOME, E-MAIL, PASSWORD e ADMINISTRADOR. A API não deve permitir o cadastro de e-mails de provedores como Gmail e Hotmail ou e-mails já utilizados. A senha deve ter entre 5 e 10 caracteres.
- **US 002 - Login**: A autenticação de usuários deve gerar um token Bearer. Usuários não cadastrados ou com senha inválida não devem conseguir autenticar, retornando um status code 401.
- **US 003 - Produtos**: Usuários não autenticados não devem conseguir realizar ações. Não deve ser possível cadastrar produtos com nomes já utilizados ou excluir produtos que estão em um carrinho.

5. Técnicas Aplicadas: SBTM, Teste de API via Postman;Teste funcional baseado em User Stories; Teste exploratório com Swagger;

6. Mapa Mental:

Mapa Mental - Serverest



7. Cenários de testes planejados

CENÁRIO	CASO DE TESTE	BDD	
002-Fazer login como um vendedor	CT002.001 - Fazer login com credenciais válidas	**Dado** que um usuário vendedor tenha cadastro já realizado, **quando** for enviada a requisição com o e-mail e a senha corretos, **então** a requisição deve retornar um status code 200 e um token de autenticação.	
	CT002.002 - Fazer login com e-mail inválido	**Dado** que um usuário vendedor queira se autenticar, **quando** for enviada a requisição com um e-mail inválido (não cadastrado), **então** a requisição deve retornar um status code 401 (Unauthorized) e o login não deve ser efetuado.	
	CT002.003 - Fazer login com senha inválida	**Dado** que um usuário vendedor queira se autenticar, **quando** for enviada a requisição com a	

		senha incorreta, **então** a requisição deve retornar um status code 401 (Unauthorized) e o login não deve ser efetuado.	
	CT002.004 - Fazer login com campos ausentes	**Dado** que um usuário vendedor queira se autenticar, **quando** for enviada a requisição faltando o campo `email` ou `password`, **então** a requisição deve retornar um status code 400 e o login não deve ser efetuado.	

8. Priorização

Os testes foram priorizados com base no impacto no negócio.

- **Alta Prioridade:** Cadastro e login de clientes e vendedores, CRUD de produtos e validações de negócio.
- **Média Prioridade:** Edição de usuários.
- **Baixa Prioridade:** Buscar usuário por ID.

9. Matriz de risco

A matriz de risco avalia a probabilidade e o impacto de falhas.

RISCO	PROBABILIDADE (1-5)	IMPACTO (1-5)	FATOR DE RISCO (Prob. x Imp.)
Cadastro aceita e-mail inválido	5	4	20
Carrinho aceita a compra sem ter estoque	4	5	20
Cadastrar um produto com um nome já utilizado	4	5	20
API deleta um produto que está em um carrinho	4	4	16
Quantidade do estoque não ser modificada ao concluir ou cancelar compra	3	5	15
Um cliente atualiza os dados de um produto	3	5	15
API deleta um usuário com carrinho	3	3	9
Falha ao encontrar um carrinho pelo ID	2	3	6

10. Cobertura de testes

A cobertura de testes se baseia no Swagger da aplicação, mas também inclui cenários alternativos para garantir uma validação completa das regras de negócio.

11. Testes candidatos à automação

Os testes de alta prioridade são candidatos ideais para automação. O uso do Postman para executar esses testes de forma automatizada, incluindo validações internas.

12. Mapeamento de Issues

https://cacabugs2025.atlassian.net/jira/software/projects/SVRAPI/boards/133?atlOrigin=eyJpIjoiOTk3YzUzOTFmZWM4NGEzYmIxNzRkZWQ3YjE2NDAyM2MiLCJwIjoiaiJ9