


# Agemilson Abreu

## 1. Apresentação

Este documento detalha o plano de testes para a API REST ServeRest (  **ServeRest** ). O objetivo é guiar as atividades de teste para assegurar a qualidade, estabilidade e conformidade da aplicação com as regras de negócio especificadas. O foco principal será a validação dos endpoints de Usuários, Login e Produtos, conforme detalhado nas User Stories, garantindo que todas as funcionalidades e critérios de aceitação sejam atendidos.

## 2. Objetivo

O objetivo principal deste plano é verificar sistematicamente e validar a funcionalidade, segurança e confiabilidade da API ServeRest. Garantir o funcionamento das operações principais para que o usuário tenha uma experiência ideal, além de assegurar que todas as regras de negócio e critérios de aceite descritos nas User Stories estejam de acordo com o esperado. Por fim, criar uma base de testes automatizada (via Postman) para facilitar a execução de testes futuros.

## 3. Escopo

- Testes manuais e automatizados (Postman Tests) nas rotas:
  - `/usuarios`
  - `/login`
  - `/produtos`
- Verificação de requisitos funcionais e regras de negócio descritos nas US 001, US 002 e US 003.
- Cenários positivos, negativos e alternativos.

## 4. Análise

A análise dos testes foi baseada na documentação Swagger da API e nas User Stories fornecidas.

### • US 001: [API] Usuários

- **Impacto:** Endpoint `/usuarios` .
- **Análise:** Esta US define o CRUD de usuários e regras de negócio essenciais como a unicidade de e-mail, validação do formato de e-mail (inclusive restrição de provedores como gmail/hotmail), e complexidade da senha. Um ponto crítico é a regra de que um `PUT` para um ID inexistente deve criar um novo usuário, o que difere do comportamento REST padrão e precisa de atenção especial.

### • US 002: [API] Login

- **Impacto:** Endpoint `/login` .
- **Análise:** O foco é a autenticação. Os testes devem cobrir cenários de sucesso e falha. A validade do token de 10 minutos é um requisito que pode ser complexo de validar em um teste automatizado curto, mas deve ser considerado.

### • US 003: [API] Produtos

- **Impacto:** Endpoint `/produtos` .
- **Análise:** Define o CRUD de produtos, com a principal restrição de que todas as rotas (exceto listagem `GET` ) exigem autenticação. As regras de negócio incluem a unicidade do nome do produto, a criação de produto via

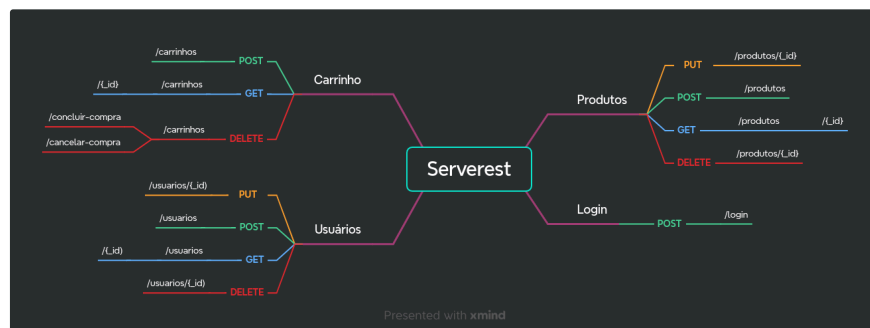
**PUT** para ID inexistente e a importante regra de integração de não permitir a exclusão de um produto associado a um carrinho.

## 5. Técnicas Aplicadas

- **Particionamento de Equivalência:** Agrupar dados de entrada em classes válidas e inválidas.
  - *Exemplo:* Para o campo **email**, as partições são: e-mails válidos (ex: **teste@dominio.com**), e-mails inválidos (ex: **teste.com**, **teste@**), e-mails de provedores restritos (ex: **user@gmail.com**).
- **Análise de Valor Limite:** Testar os limites dos campos de entrada.
  - *Exemplo:* Para o campo **password** (mínimo 5, máximo 10 caracteres), testar com 4, valor entre 5 e 10, e 11 caracteres.
- **Teste Exploratório:** Navegação livre pela API via Postman para descobrir defeitos não previstos nos cenários formais.
- **Teste Baseado em Requisitos:** Criação de cenários de teste diretamente a partir dos Critérios de Aceitação das User Stories para garantir 100% de cobertura dos requisitos.
- **Null, Zero, Vazio:** Validando se campos aceitam inputs de vazios ou valores inexistentes que não deveriam ser aceitos.

## 6. Mapa Mental da Aplicação

Foi construído um mapa mental detalhando todos os endpoints e os verbos de cada um.



## 7. Cenários de Teste Planejados

A seguir, uma amostra dos cenários planejados. A lista completa estará na collection do Postman.

ID	Feature	Descrição do Cenário	Esperado	Prioridade
CT-USR-01	Usuários	Criar um novo usuário com dados válidos	Sucesso	P1
CT-USR-02	Usuários	Tentar criar um usuário com um e-mail já existente	Falha	P1

CT-USR-03	Usuários	Tentar criar um usuário com e-mail de provedor restrito [Gmail, Hotmail]	Falha	P2
CT-USR-04	Usuários	Tentar criar um usuário com senha fora do limite de caracteres (entre 5 e 10 caracteres)	Falha	P2
CT-USR-05	Usuários	Tentar criar um usuário com campos inválidos [Email, Nome, Senha]	Falha	P1
CT-USR-06	Usuários	Atualizar um usuário existente (PUT) com um ID válido	Sucesso	P2
CT-USR-07	Usuários	Criar um novo usuário ao tentar atualizar (PUT) com um ID inexistente.	Sucesso	P3
CT-USR-08	Usuários	Criar um novo usuário usando um email existente	Falha	P2
CT-LOG-01	Login	Realizar login com credenciais válidas e	Sucesso	P1

		receber token.		
CT-LOG-02	Login	Tentar realizar login com senha incorreta.	Falha	P1
CT-LOG-03	Login	Tentar realizar login com um e-mail não cadastrado.	Falha	P1
CT-PRO-01	Produtos	Cadastrar um novo produto com token de autenticação válido.	Sucesso	P1
CT-PRO-02	Produtos	Tentar cadastrar um produto sem token de autenticação .	Falha	P1
CT-PRO-03	Produtos	Tentar cadastrar um produto com nome duplicado.	Falha	P2
CT-PRO-04	Produtos	Criar um novo produto com ID inexistente (PUT)	Sucesso	P2
CT-PRO-05	Produtos	Tentar criar um produto com nome duplicado (PUT)	Falha	P2
CT-PRO-06	Produtos	Tentar excluir um	Falha	P2

		produto que está em um carrinho.		
CT-PRO-07	Produtos	Excluir um produto que não está em um carrinho.	Sucesso	P2

## 8. Priorização da Execução dos Cenários de Teste

A priorização seguirá o critério de criticidade para o negócio, focando primeiro nos fluxos que viabilizam a operação da plataforma.

- Prioridade 1 (Crítica - Must Have): Caminhos felizes essenciais e falhas críticas.
  - Login com sucesso e falha.
  - Cadastro de usuário com sucesso.
  - Cadastro de produto com sucesso (autenticado).
  - Listagem de usuários e produtos.
- Prioridade 2 (Alta - Should Have): Validações de regras de negócio importantes e fluxos de atualização/deleção.
  - Validações de campos (e-mail duplicado, nome de produto duplicado, senha).
  - Atualização de usuários e produtos.
  - Deleção de usuários e produtos (cenário simples).
  - Regra de não exclusão de produto em carrinho.
- Prioridade 3 (Média - Could Have): Casos de borda e regras de negócio menos comuns.
  - Regra de criação de novo recurso via **PUT** com ID inexistente.
  - Validações de todos os campos de todos os endpoints.
  - Testes com múltiplos filtros em endpoints **GET**.

## 9. Matriz de Risco

Risco	Probabilidade	Impacto	Nível do Risco	Mitigação
Falha na Autenticação o impede o uso de 90% da API.	Média	Alto	Crítico	Priorizar testes de <b>login</b> (P1). Automação de smoke test para o endpoint de login.

Violação de Regra de Negócio (ex: e-mail/produto único) causa inconsistência de dados.	Média	Médio	Alto	Cenários de teste específicos para duplicidade (P2), com dados pré-cadastrados.
Acesso Não Autorizado a rotas protegidas.	Baixa	Alto	Alto	Testes rigorosos em todos os endpoints que exigem token, verificando acesso com e sem token.
Inconsistência de Dados pela regra de PUT criar recurso se ID não existe.	Média	Médio	Médio	Cenários dedicados para PUT em IDs existentes e inexistentes (P3).
Não conformidade com o Contrato (Swagger) quebra integração de clientes.	Baixa	Médio	Médio	Utilizar a collection do Postman para validar os schemas de resposta contra a documentação.

## 10. Cobertura de Testes

A cobertura será medida de duas formas:

1. Cobertura de Requisitos: Será criada uma matriz de rastreabilidade ligando cada Critério de Aceitação das User Stories a um ou mais Cenários de Teste (IDs CT-XXX-XXX). O objetivo é atingir 100% de cobertura dos requisitos documentados.
2. Cobertura de Endpoints (API Coverage): Garantir que todos os verbos ( GET , POST , PUT , DELETE ) de todos os endpoints em escopo ( /usuarios , /login , /produtos ) sejam cobertos por pelo menos um teste de

sucesso e um de falha principal.

## 11. Testes Candidatos a Automação

A automação será focada nos testes já planejados para garantir a estabilidade da aplicação. A estratégia é dividida em:

1. **Smoke Test (Verificação do Fluxo Principal)** Consiste em um único fluxo de ponta a ponta que valida se as operações mais críticas da API estão funcionando. As requisições que serão executadas em sequência:
  - **POST /usuarios** (Criação de usuário - CT-USR-01)
  - **POST /login** (Autenticação do usuário - CT-LOG-01)
  - **POST /produtos** (Criação de produto autenticado - CT-PRO-01)
  - **DELETE /produtos/{id}** (Exclusão do produto - CT-PRO-07)
  - **DELETE /usuarios/{id}** (Limpeza do usuário criado)
2. **Testes de Regressão (Validação das Regras de Negócio)** Esta suíte agrupa os principais cenários de sucesso e falha que para validar as regras de negócio mais importantes:
  - **Validações de Usuário:** Inclui os testes de e-mail duplicado ( CT-USR-02 ), provedor restrito ( CT-USR-03 ) e os limites de senha ( CT-USR-04 ).
  - **Validações de Produto:** Cobre a tentativa de cadastro com nome duplicado ( CT-PRO-03 ) e o acesso a rotas sem token ( CT-PRO-02 ).
  - **Regras de PUT :** Testa os comportamentos específicos do **PUT** , como a criação de um novo produto com ID inexistente e a falha ao tentar atualizar com um nome duplicado.
  - **Teste de Dependência:** Valida o bloqueio de exclusão de um produto que está em um carrinho ( CT-PRO-06 ).