Challenge 02 - Plano de Testes (ServeRest)

Plano de Testes

> **Software:** API Serverest
>
> **QA Responsável:** Karen Késsia Herrmann

---

1. Apresentação

"Este plano de testes visa validar as funcionalidades da API ServeRest — cadastro de usuários, login e produtos — conforme as regras de negócio definidas nas User Stories (US001 a US003) e documentação do Swagger.”

2. Objetivo

Garantir que a API ServeRest esteja conforme os critérios definidos, segura, resistente a dados inválidos e com comportamento adequado em todos os cenários.

3. Escopo

Escopo incluso:

- Módulo de **Usuários** (US001).
- Módulo de **Login** (US002).
- Módulo de **Produtos** (US003).
- Módulo de **Carrinhos**
- Verificação de regras de negócio, como e-mails válidos, senhas, duplicidade, token, etc.

Escopo excluído:

- Testes de Front-End
- Integração com API’s Externas
- Testes de carga, performance e segurança avançada.

Análise de Testes

Análise foi feita a partir de:

- Swagger oficial da API: ServeRest
- User Stories fornecidas (US001 a US003).

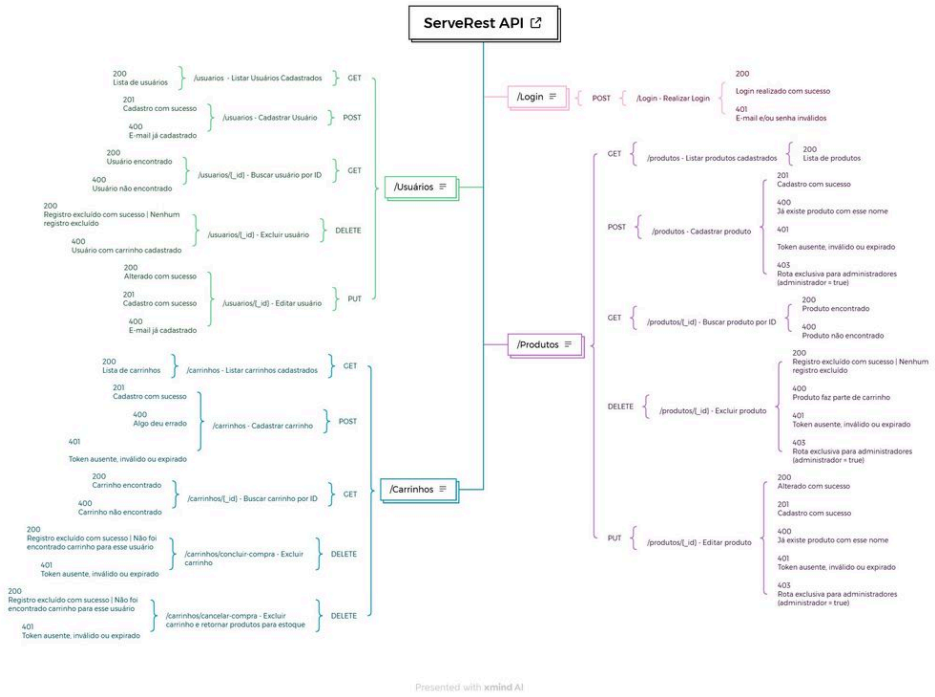
Técnicas Aplicadas

- Análise de Partições de Equivalência
- Valor Limite
- CRUD

- Heurísticas como análise do Valor Limite, Teste Baseado em Risco, Exploratórios Orientados a Regras e análise por partições de equivalência.
- Teste Baseado em Riscos
- Testes Exploratórios
- Teste Baseado em Regras de Negócio

---

4. Mapa Mental da Aplicação



5. Users Storys e Critérios de Aceitação

US & CA

User Stories

US 001 - [API] Usuários

Sendo um vendedor de uma loja

Gostaria de poder me cadastrar no Marketplace do ServeRest

Para poder realizar as vendas dos meus produtos

DoR

- Banco de dados e infraestrutura para desenvolvimento disponibilizados;
- Ambiente de testes disponibilizado.

DoD

- CRUD de cadastro de vendedores (usuários) implementado (CRIAR, ATUALIZAR, LISTAR E DELETAR);
- Análise de testes cobrindo todos verbos;

- Matriz de rastreabilidade atualizada;
- Automação de testes baseado na análise realizada;

Acceptance Criteria

- Os vendedores (usuários) deverão possuir os campos NOME, E-MAIL, PASSWORD e ADMINISTRADOR;
- Não deverá ser possível fazer ações e chamadas para usuários inexistentes;
- Não deve ser possível criar um usuário com e-mail já utilizado;
- Caso não seja encontrado usuário com o ID informado no PUT, um novo usuário deverá ser criado;
- Não deve ser possível cadastrar usuário com e-mail já utilizado utilizando PUT;
- Os testes executados deverão conter evidências;
- Não deverá ser possível cadastrar usuários com e-mails de provedor gmail e hotmail;
- Os e-mails devem seguir um padrão válido de e-mail para o cadastro;
- As senhas devem possuír no mínimo 5 caracteres e no máximo 10 caracteres;
- A cobertura de testes deve se basear no Swagger e ir além, cobrindo cenários alternativos.

US 002: [API] Login

Sendo um vendedor de uma loja com cadastro já realizado

Gostaria de poder me autenticar no Marketplace da ServeRest

Para poder cadastrar, editar, atualizar e excluir meus produtos

DoR

- Banco de dados e infraestrutura para desenvolvimento disponibilizados;
- API de cadastro de usuários implementada;
- Ambiente de testes disponibilizado.

DoD

- Autenticação com geração de token Bearer implementada;
- Análise de testes cobrindo a rota de login;
- Matriz de rastreabilidade atualizada;
- Automação de testes baseado na análise realizada;

Acceptance Criteria

- Usuários não cadastrados não deverão conseguir autenticar;
- Usuários com senha inválida não deverão conseguir autenticar;
- No caso de não autenticação, deverá ser retornado um status code 401 (Unauthorized);
- Usuários existentes e com a senha correta deverão ser autenticados;
- A autenticação deverá gerar um token Bearer;
- A duração da validade do token deverá ser de 10 minutos;
- Os testes executados deverão conter evidências;
- A cobertura de testes deve se basear no Swagger e ir além, cobrindo cenários alternativos.

US 003: [API] Produtos

Sendo um vendedor de uma loja com cadastro já realizado

Gostaria de poder me autenticar e cadastrar produtos no Marketplace do ServeRest

Para poder cadastrar, editar, atualizar e excluir meus produtos

DoR

- Banco de dados e infraestrutura para desenvolvimento disponibilizados;
- API de cadastro de usuários implementada;
- API de autenticação implementada;
- Ambiente de testes disponibilizado.

DoD

- CRUD de cadastro de Produtos implementado (CRIAR, ATUALIZAR, LISTAR E DELETAR);
- Análise de testes cobrindo a rota de produtos;
- Matriz de rastreabilidade atualizada;
- Automação de testes baseado na análise realizada;

Acceptance Criteria

- Usuários não autenticados não devem conseguir realizar ações na rota de Produtos;
- Não deve ser possível realizar o cadastro de produtos com nomes já utilizados;
- Não deve ser possível excluir produtos que estão dentro de carrinhos (dependência API Carrinhos);
- Caso não exista produto com o o ID informado na hora do UPDATE, um novo produto deverá ser criado;
- Produtos criados através do PUT não poderão ter nomes previamente cadastrados;
- Os testes executados deverão conter evidências;
- A cobertura de testes deve se basear no Swagger e ir além, cobrindo cenários alternativos.

6. Cenários de Teste Planejados

ID	Cenário	Endpoint	Resultado Esperado	Estratégia
TC001	Criar usuário com dados válidos	*POST /usuarios*	201 Created	Fluxo principal
TC002	Criar usuário com domínio de e-mail inválido (Gmail/Hotmail)	*POST /usuarios*	400	Regra de unicidade
TC003	Criar usuário com e-mail duplicado	*POST /usuarios*	400	Validação de formato
TC004	Atualizar usuário com ID inexistente	*PUT /usuarios/{id}*	201 Created	Upsert

TC005	Consultar usuário inexistente	*GET /usuarios/{id}*	400	Consulta negativa
TC006	Deletar usuário válido	*DELETE /usuarios/{id}*	200 OK	Cobertura CRUD
TC007	Criar usuário com senha no limite	*POST /usuarios*	201 Created	Valor-limite válido
TC008	Criar usuário com senha inferior a 5 caracteres	*POST /usuarios*	400	Valor-limite inferior
TC009	Criar usuário com senha superior a 10 caracteres	*POST /usuarios*	400	Valor-limite superior
TC010	Atualizar usuário com e-mail duplicado	*PUT /usuários/{id}*	400	Regra de unicidade
TC011	Deletar usuário inexistente	*DELETE /usuários/{id}*	400	Exclusão negativa
TC012	Login com credenciais válidas	*POST /login*	200 OK	Fluxo principal
TC013	Login com senha inválida	*POST /login*	401	Autenticação negativa
TC014	Login com usuário não cadastrado	*POST /login*	401	Autenticação negativa
TC015	Acessar rota protegida com token expirado	*/login*	401	Teste de sessão
TC016	Criar produto com dados válidos	*POST /produtos*	201 Created	Fluxo principal
TC017	Criar produto com nome duplicado	*POST /produtos*	400	Regra de unicidade
TC018	Deletar produto vinculado a carrinho	*DELETE /produtos/{id}*	400	Regra de negócio
TC019	Atualizar produto com ID inexistente	*DELETE /produtos/{id*	201 Created	Upsert
TC020	Acessar produtos sem autenticação	*POST /produtos*	401	Autenticação negativa

TC021	Atualizar produto com nome duplicado	*PUT /produtos/{id}*	400	Regra de unicidade
TC022	Criar produto como usuário não administrador	*POST /produtos*	403	Regra de permissão
TC023	Criar produto com preço inválido	*POST /produtos*	400	Valor-limite inválido
TC024	Fluxo integrado usuário-produto-carrinho	*POST /usuarios, POST /login, POST /produtos*	200, 201 Created	Teste de integração
TC025	Criar carrinho com produtos válidos	*POST /carrinhos*	201 Created	Fluxo principal
TC026	Cancelar carrinho e reabastecer estoque	*DELETE /carrinhos/cancelar-compra*	200 OK	Teste de integração
TC027	Impedir criação de carrinho duplicado	*POST /carrinhos*	400	Teste negativo

Total de testes planejados: **27 cenários**

7. Matriz de Risco

A matriz de risco a seguir foi construída a partir dos cenários de teste aplicados à API de usuários e produtos. Ela identifica possíveis falhas críticas como **cadastro duplicado**, **acesso não autorizado**, e **operações inválidas**, avaliando o **impacto** que cada uma pode causar no sistema e a **probabilidade** de ocorrência com base na natureza da funcionalidade. Os riscos classificados como **críticos** concentram-se, principalmente, em violações de regras de negócio e falhas de segurança, sendo prioritários para testes rigorosos.

Risco	TC	Descrição	Probabilidade	Impacto	Prioridade
R001	TC002	Aceitar domínios de e-mail inválidos	Médio	Alto	Alto
R002	TC003	Permitir cadastro de e-mails duplicados	Alto	Alto	Crítico

R004	TC010	Atualizar usuário com e-mail já cadastrado	Médio	Alto	Alto
R006	TC013	Permitir login com senha inválida	Alto	Médio	Alto
R014	TC012	Login com credenciais válidas	Baixo	Alto	Médio
R015	TC015	Acessar rota protegida com token expirado	Médio	Alto	Alto
R008	TC017	Criar produto com nome duplicado	Alto	Alto	Crítico
R011	TC022	Usuário comum conseguir criar produto	Crítico	Médio	Crítico
R016	TC016	Criar produto com dados válidos	Baixo	Alto	Médio

8. Cobertura de Testes

> Foram planejados **27 cenários de teste** no total, todos **validados manualmente via Postman**, garantindo **100% de cobertura funcional** em relação aos fluxos principais e riscos identificados na matriz.
>
> Destes, **14 cenários foram selecionados para automação com Robot Framework**, com base em critérios de **criticidade, repetitividade e impacto nas regras de negócio**. A automação tem como foco promover eficiência e confiabilidade contínua sobre funcionalidades sensíveis da API ServeRest.

Total de Cenários Planejados	Cenários Automatizados (Robot)	Cenários Testados Manualmente (Postman)	Cobertura Funcional (%)
27	**14**	**27**	**100%**

9. Testes Candidatos à Automação

/Usuários

TC001 - Criar usuário com dados válidos

TC002 - Criar usuário com domínio de e-mail inválido

TC003 - Criar usuário com e-mail duplicado

TC004 - Atualizar usuário com ID inexistente

TC005 - Consultar usuário inexistente

/Login

TC012 - Login com credenciais válidas

TC013 - Login com senha inválida

TC014 - Login com usuário não cadastrado

TC015 - Acessar rota protegida com token expirado

/Produtos

TC016 - Criar produto com dados válidos

TC017 - Criar produto com nome duplicado

TC018 - Deletar produto vinculado a carrinho

TC019 - Atualizar produto com ID inexistente

TC020 - Acessar produtos sem autenticação

> Os cenários automatizados priorizam **fluxos principais**, **validações críticas** e **testes com alta frequência de execução**, cobrindo integralmente as User Stories **US 001**, **US 002** e **US 003**.
>
> Inicialmente, os testes foram executados no **Postman.** Entretanto, está sendo realizada uma transição gradual para o **Robot Framework**, que passará a **centralizar os testes automatizados da API**.
>
> Para isso, será utilizada a biblioteca **RequestsLibrary**, que permite a realização de **chamadas HTTP**, **validação de status**, **análise de payloads**, **autenticação** e **manipulação de headers** — proporcionando maior **flexibilidade**, **legibilidade** e **controle** sobre os testes.

10. Entregáveis

Relatório Analítico de Testes > ServeRest