



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:

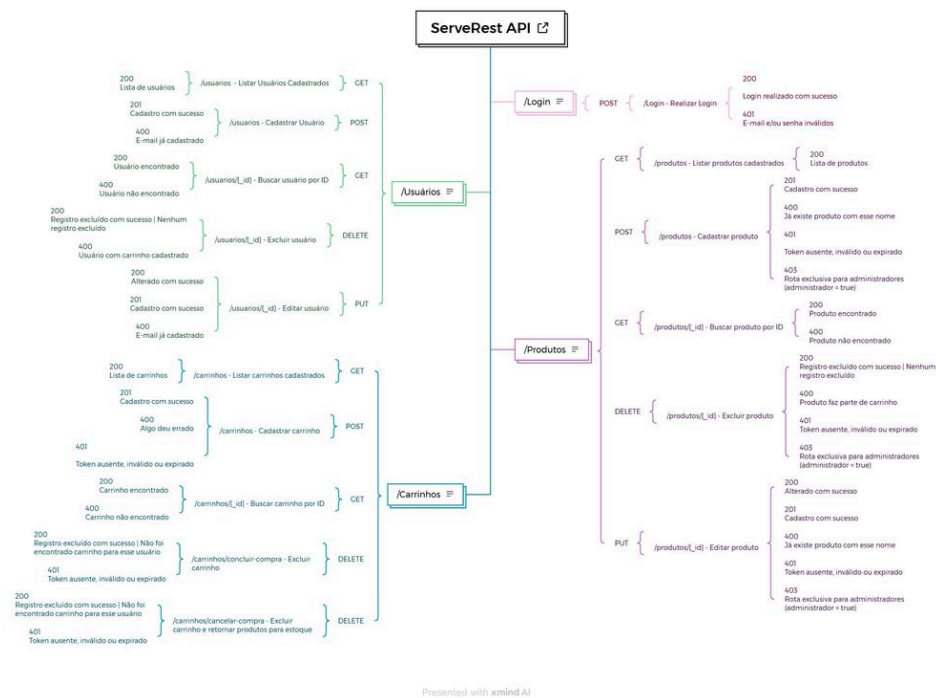
- Swagger oficial da API:  [ServeRest](#)
- User Stories fornecidas (US001 a US003).

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. 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	<i>DELETE /usuarios/{id}</i>	200 OK	Cobertura CRUD
TC007	Criar usuário com senha no limite	<i>POST /usuarios</i>	201 Created	Valor-limite válido
TC008	Criar usuário com senha inferior a 5 caracteres	<i>POST /usuarios</i>	400	Valor-limite inferior
TC009	Criar usuário com senha superior a 10 caracteres	<i>POST /usuarios</i>	400	Valor-limite superior
TC010	Atualizar usuário com e-mail duplicado	<i>PUT /usuários/{id}</i>	400	Regra de unicidade
TC011	Deletar usuário inexistente	<i>DELETE /usuários/{id}</i>	400	Exclusão negativa
TC012	Login com credenciais válidas	<i>POST /login</i>	200 OK	Fluxo principal
TC013	Login com senha inválida	<i>POST /login</i>	401	Autenticação negativa
TC014	Login com usuário não cadastrado	<i>POST /login</i>	401	Autenticação negativa
TC015	Acessar rota protegida com token expirado	<i>/login</i>	401	Teste de sessão
TC016	Criar produto com dados válidos	<i>POST /produtos</i>	201 Created	Fluxo principal
TC017	Criar produto com nome duplicado	<i>POST /produtos</i>	400	Regra de unicidade
TC018	Deletar produto vinculado a carrinho	<i>DELETE /produtos/{id}</i>	400	Regra de negócio
TC019	Atualizar produto com ID inexistente	<i>DELETE /produtos/{id}</i>	201 Created	Upsert
TC020	Acessar produtos sem autenticação	<i>POST /produtos</i>	401	Autenticação negativa
TC021	Atualizar produto com nome duplicado	<i>PUT /produtos/{id}</i>	400	Regra de unicidade

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

Total de testes planejados: **27 cenários**

6. 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

7. 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%

8. Testes Candidatos à Automação [🔗](#)

/Usuários

TC001 - Criar usuário com dados válidos

/Login

TC012 - Login com credenciais válidas

TC013 - Login com senha inválida

/Produtos

TC016 - Criar produto com dados válidos

TC002 - Criar usuário com domínio de e-mail inválido/GMAIL

TC002 - Criar usuário com domínio de e-mail inválido/HOTMAIL

TC003 - Criar usuário com e-mail duplicado

TC005 - Consultar usuário inexistente

TC008 - Criar usuário com senha inferior a 5 caracteres

TC009 - Criar usuário com senha superior a 10 caracteres

TC010- Atualizar usuário com e-mail duplicado

TC014 - Login com usuário não cadastrado

TC017 - Criar produto com nome duplicado

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. Além disso, foram adicionados cenários **extras** como forma de completude, garantindo uma cobertura mais ampla e robusta da API.

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 aplicação.

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.

9. Entregáveis [↗](#)

 [Relatório Analítico de Testes > ServeRest](#)