Eric Lima

PLANO DE TESTE - SERVEREST

1. APRESENTAÇÃO

Este plano de testes foi elaborado para a API **ServeRest**, instância da **Compass UOL**.. A aplicação simula um marketplace, permitindo o gerenciamento de usuários, autenticação, cadastro de produtos e carrinhos de compras.

- **Instância utilizada:** https://compassuol.serverest.dev/
- **Versão da API:** 2.29.7
  Escopo principal: rotas /usuarios, /login, /produtos e /carrinhos.

1.2 EQUIPE ENVOLVIDA

- **Testador responsável:** Eric Lima da Silva
- **Squad:** 2 - Beaver
- **Projeto:** Challenge 02 – Semana 8

2. OBJETIVO

O objetivo deste plano é **validar a conformidade funcional da API ServeRest**, assegurando que suas rotas principais — **/usuarios, /login, /produtos e /carrinhos** — atendam corretamente às regras de negócio definidas. A meta é garantir que as operações de CRUD, autenticação e gerenciamento de carrinhos funcionem conforme esperado, identificando falhas, inconsistências e oportunidades de melhoria.

Além disso, busca-se **fornecer evidências de execução dos testes**, mapear riscos, definir candidatos à automação e a criação de uma coleção Postman.

3. ESCOPO

O escopo deste plano contempla as principais rotas da API **ServeRest**, cobrindo tanto **funcionalidades essenciais** quanto **validações de regras de negócio**. Os testes foram planejados para avaliar **operações de CRUD**, **autenticação/autorização** e **interações entre recursos**, assegurando confiabilidade do sistema.

Rotas incluídas:

- **/usuarios** → validação do CRUD de vendedores (criação, atualização, listagem e exclusão), incluindo regras de unicidade de e-mail, bloqueio de provedores Gmail/Hotmail, e restrições de senha (mín. 5 e máx. 10 caracteres).
- **/login** → autenticação de usuários cadastrados, geração e expiração do token Bearer, validação de credenciais inválidas e cenários de falha (401 Unauthorized).
- **/produtos** → CRUD de produtos, com verificação de unicidade de nome, necessidade de autenticação, e impossibilidade de exclusão de itens vinculados a carrinhos.
- **/carrinho** → criação, atualização e exclusão de carrinhos de compras, garantindo que produtos adicionados respeitem regras de disponibilidade e que exclusões reflitam corretamente nos estoques.

Fora do escopo:

- Testes de performance e carga (ex.: tempo de resposta sob alto volume de requisições).
- Testes de segurança avançados (SQL injection, XSS, fuzzing).
- Integrações externas além do ambiente da própria API.

3.1 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.

4. ANÁLISE

A análise foi realizada a partir das **User Stories** fornecidas e da documentação Swagger da API ServeRest, contemplando os critérios de aceite e possíveis riscos associados ao uso do

sistema. O objetivo desta etapa é identificar as **regras de negócio críticas**, **pontos vulneráveis** e **cenários de falha potenciais** para priorizar os testes.

USUÁRIOS (/USUARIOS)

- **Requisitos:**
  - CRUD completo de vendedores (criar, listar, atualizar e deletar).
  - Campos obrigatórios: **nome, e-mail, password, administrador**.
  - Não permitir criação com e-mail duplicado.
  - Bloqueio de cadastro com provedores **gmail.com** e **hotmail**.com .
  - Senha deve possuir entre **5 e 10 caracteres**.
  - PUT em ID inexistente deve criar um novo usuário.
- **Riscos:**
  - Permitir cadastro com e-mail inválido → risco de inconsistência.
  - Falha na validação de senha → risco de brecha de segurança.
  - Duplicidade de e-mails → risco de integridade no banco de dados.

LOGIN (/LOGIN)

- **Requisitos:**
  Apenas usuários cadastrados podem autenticar.
  - Autenticação deve gerar **token Bearer válido por 10 minutos**.
  - Login inválido deve retornar **401 Unauthorized**.
- **Riscos:**
  Aceitar login com senha incorreta → risco de acesso indevido.
  - Token sem expiração correta → risco de sessão insegura.
  - Erros genéricos em vez de mensagens adequadas → dificultam análise de falha.

PRODUTOS (/PRODUTOS)

- **Requisitos:**
  - CRUD de produtos restrito a usuários autenticados.
  - Nome de produto deve ser único.
  - Produtos vinculados a **carrinhos** não podem ser excluídos.
  - PUT em ID inexistente deve criar novo produto.
- **Riscos:**

- Falta de validação no cadastro → duplicidade ou dados inconsistentes.
  Exclusão indevida de produto ainda em uso → risco de quebra de fluxo de carrinho.
- Cadastro sem autenticação → risco de acesso não autorizado.

CARRINHOS (/CARRINHOS)

- **Requisitos:**
  - CRUD de carrinhos vinculado a usuários autenticados.
  - Deve validar existência e estoque dos produtos adicionados.
  - Exclusão do carrinho deve atualizar corretamente a disponibilidade dos produtos.
- **Riscos:**
  Permitir adicionar produto inexistente → risco de erro em vendas.
  - Estoque não atualizado após exclusão → inconsistência de inventário.
    Manipulação de carrinho sem token válido → risco de fraude.

5. TÉCNICAS APLICADAS

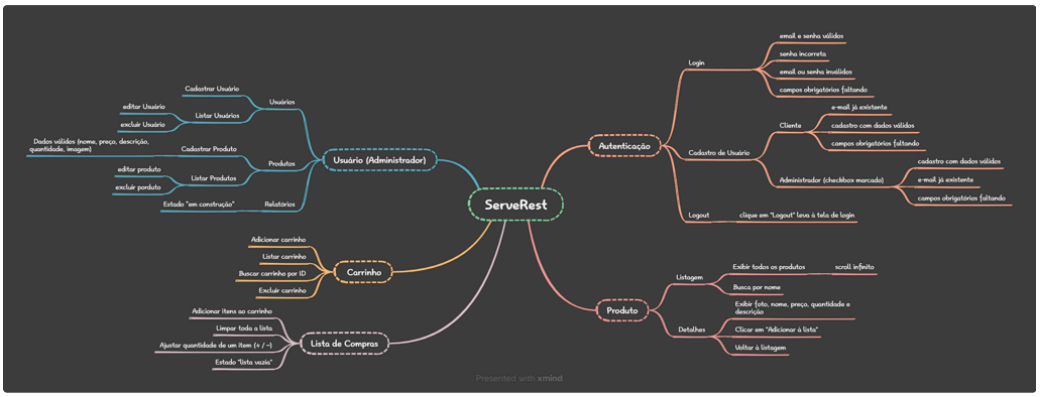
Para garantir a efetividade dos testes na API **ServeRest (v2.29.7)**, foram aplicadas técnicas clássicas de **teste de caixa-preta**, priorizando simplicidade e cobertura de regras de negócio:

- **Particionamento de Equivalência**
  - Separação de valores válidos e inválidos para inputs como e-mail, senha e nomes de produtos.
  - Ex.: e-mails válidos vs. inválidos; senhas com menos de 5, entre 5 e 10 e acima de 10 caracteres.
- **Análise de Valores Limite**
  - Testes nos extremos definidos pelas regras de negócio.
  - Ex.: senha com 4 caracteres (inválida), senha com 5 (válida), senha com 10 (válida), senha com 11 (inválida).
- **Baseado em Risco**
  - Priorização de cenários críticos que afetam diretamente segurança e integridade de dados.
    Ex.: login com credenciais inválidas, exclusão de produtos em carrinho, criação de usuários duplicados.
- **Teste de Tabela de Decisão**
  - Representação de combinações de condições e ações esperadas.

- Ex.: autenticação → (usuário cadastrado/não cadastrado) × (senha válida/inválida).

- **Teste de Transição de Estado**
  - Foco em mudanças de estado do sistema.
  - Ex.: produto adicionado a um carrinho, carrinho excluído → estoque atualizado.

6. MAPA MENTAL DA APLICAÇÃO



7. CENÁRIOS DE TESTE PLANEJADOS

USUÁRIOS

ID	Descrição	Entrada	Resultado Esperado
U001	Criar usuário válido	nome="Teste QA", email="teste.qa@example.com", password="senha123", administrador="true"	201
U002	Impedir e-mail duplicado	mesmo email de U001	400; mensagem indicando **e-mail já utilizado**
U003	Bloquear provedores	email com **@gmail.com** ou	400; cadastro **não permitido**

	proibidos	**@hotmail.com**	
U004	Validar limites de senha	password="abcd" (4) / "abcdefghijk" (11)	400; mensagem de **tamanho inválido**
U005	PUT cria quando ID não existe	PUT /usuarios/{idInexistente} com payload válido	200/201/; usuário **criado** para o ID
U006	Buscar usuário por ID existente	GET /usuarios/{idCriado}	200; retorna dados do usuário
U007	Impedir operações com usuário inexistente	GET/PUT/DELETE em {idInexistente}	400; mensagem de **não encontrado**
U008	Listar usuários	GET /usuarios	200; lista não vazia (após U001)
U009	Deletar usuário existente	DELETE /usuarios/{idCriado}	200; usuário **removido**

LOGIN

ID	**Descrição**	**Entrada**	**Resultado Esperado**
L001	Autenticar com credenciais válidas	email do U001, senha correta	200; retorna **token Bearer** (validade 10 min)

L002	Recusar usuário não cadastrado	email inexistente, senha qualquer	**401 Unauthorized**
L003	Recusar senha inválida	email existente, senha errada	**401 Unauthorized**
L004	Usar token em rota protegida	Authorization: Bearer <token válido>	Acesso permitido às rotas protegidas
L005	Token expirado	token após >10 min	Acesso negado (status 401 conforme implementação)

PRODUTOS

ID	**Descrição**	**Entrada**	**Resultado Esperado**
P001	Criar produto sem autenticação	POST sem header Authorization	**401** (acesso negado)
P002	Criar produto válido (autenticado)	nome="Teclado QA", preco=100, descricao="...", quantidade=5	201; corpo contém **_id**
P003	Impedir nome de produto duplicado	repetir nome de P002	400; mensagem de **nome já cadastrado**
P004	Listar produtos	GET /produtos	200; lista contendo item criado

P005	Buscar produto por ID	GET /produtos/{idCriado}	200; retorna dados do produto
P006	PUT cria quando ID não existe	PUT /produtos/{idInexistente} com payload válido	200/201; produto **criado/atualizado**
P007	Excluir produto sem vínculos	DELETE /produtos/{idCriadoSemCarrinho}	200; produto removido
P008	Bloquear exclusão se em carrinho	DELETE /produtos/{idEmCarrinho}	400; exclusão **não permitida** (dependência de Carrinhos).

CARRINHO

ID	**Descrição**	**Entrada**	**Resultado Esperado**
C001	Criar carrinho com produto existente	produtoId={idProduto}, quantidade=1 (token válido)	201; carrinho criado com item
C002	Impedir produto inexistente	produtoId={idInexistente}	400; validação falha
C003	Listar carrinhos	GET /carrinhos	200; lista contendo carrinho criado
C004	Buscar carrinho por ID	GET /carrinhos/{idCa	200; retorna itens e totais

		rrinho}	
C005	Remover carrinho e restaurar estoque	DELETE /carrinhos/{idCarrinho}	200; estoque dos produtos **restaurado**
C006	Acesso sem autenticação	operações sem token	**401**; acesso negado

8. PRIORIZAÇÃO DA EXECUÇÃO DOS CENÁRIOS DE TESTE

ID	Funcionalidade	Cenário	Prioridade
L002	Login	Usuário não cadastrado → 401	**Alta**
L003	Login	Senha inválida → 401	**Alta**
P001	Produtos	Criar sem token → 401/403	**Alta**
U002	Usuários	E-mail duplicado	**Alta**
U003	Usuários	Domínios bloqueados (gmail/hotmail)	**Alta**
P008	Produtos	Bloquear exclusão se em carrinho	**Alta**
L001	Login	Login válido (gera token)	**Alta**

P002	Produtos	Criar produto válido (autenticado)	**Alta**
U004	Usuários	Limites de senha (4/5/10/11)	**Média**
U005	Usuários	PUT cria quando ID não existe	**Média**
P006	Produtos	PUT cria quando ID não existe	**Média**
C001	Carrinhos	Criar carrinho com produto existente	**Média**
C005	Carrinhos	Remover carrinho e restaurar estoque	**Média**
U001	Usuários	Criar usuário válido	**Média**
P004	Produtos	Listar produtos	**Baixa**
U008	Usuários	Listar usuários	**Baixa**
U006	Usuários	Buscar por ID existente	**Baixa**
P005	Produtos	Buscar por ID	**Baixa**
C003	Carrinhos	Listar carrinhos	**Baixa**

9. MATRIZ DE RISCO

A matriz de risco apresentada a seguir foi elaborada com base nas funcionalidades críticas da API **ServeRest**, considerando os cenários planejados nos testes. Para cada risco identificado, foram atribuídos valores de **Probabilidade (1 a 5)** e **Impacto (1 a 5)**:

- **Probabilidade (P):** indica a chance do risco ocorrer durante a utilização do sistema.
- **Impacto (I):** reflete o nível de gravidade caso o risco se concretize, variando de falhas menores até comprometimento da segurança ou integridade dos dados.
- **Classificação (P x I):** resultado da multiplicação dos dois fatores, utilizado para priorizar ações.

Escala de Probabilidade e Impacto

Para a classificação dos riscos da API **ServeRest**, foram definidos os seguintes critérios numéricos:

Probabilidade (P)

Valor	Descrição
1	Raro – chance mínima de ocorrer
2	Pouco provável – pode acontecer em casos isolados
3	Moderado – ocorre ocasionalmente
4	Frequente – ocorre com certa regularidade
5	Quase certo – ocorre com alta frequência

Impacto (I)

Valor	Descrição
1	Baixo – efeito mínimo, não compromete a aplicação
2	Limitado – gera inconsistências simples ou retrabalho
3	Médio – pode comprometer dados ou experiência do usuário

4	Alto – compromete regras de negócio importantes
5	Crítico – causa falhas graves de segurança ou perda de dados

Classificação Final

A **classificação (P x I)** resulta da multiplicação entre os dois fatores.

- **1–5:** Baixo risco
- **6–10:** Médio risco
- **11–15:** Alto risco

Endpoint	Nome do Risco	Probabilidade	Impacto	Classificação (P x I)	Estratégia
POST /login	Usuário com senha inválida consegue se autenticar	3	4	12	Mitigar
POST /login	Token Bearer não é gerado após autenticação válida	4	2	8	Mitigar
POST /usuarios	Usuário consegue se cadastrar sem	2	4	8	Mitigar

	preencher todos os campos obrigatórios				
POST /usuarios	Usuário consegue cadastrar e-mail já existente	3	5	15	Mitigar
GET /usuarios	Usuário comum acessa lista de todos os usuários cadastrados	3	5	15	Mitigar
DELETE /usuarios/{id}	Usuário com carrinho ativo é excluído do sistema	2	3	6	Mitigar
POST /produtos	Usuário não autenticado cadastra produto	2	4	8	Mitigar
POST /produtos	Produto é cadastrado com	3	3	9	Mitigar

	nome já existente				
PUT /produtos/{id}	Usuário não administrador consegue editar produto	3	5	15	Mitigar
DELETE /produtos/{id}	Produto dentro de carrinho é excluído	3	3	9	Mitigar
POST /carrinhos	Usuário não autenticado cadastra carrinho	2	4	8	Mitigar
POST /carrinhos	Carrinho é cadastrado com produto duplicado	5	2	10	Mitigar
POST /carrinhos	Mais de um carrinho é cadastrado pelo mesmo usuário	3	5	15	Mitigar
POST /carrinhos	Carrinho é cadastrado com	4	3	12	Mitigar

	produto com quantidade insuficiente				
GET /carrinhos/{id}	Usuário consegue buscar carrinho de outro usuário	2	5	10	Mitigar

10. COBERTURA DE TESTES

Abordagem: a cobertura foi mensurada por **requisitos (User Stories e critérios de aceite)**, **operações CRUD**, **autenticação/autorização** e **regras de negócio**.

1) Cobertura por requisitos (User Stories)

- **US001 – Usuários:** 100% dos critérios de aceite cobertos nos cenários (e-mail único, bloqueio gmail/hotmail, senha 5–10, operações em inexistentes e **PUT cria**).
- **US002 – Login:** 100% dos critérios de aceite cobertos (401 para falhas, geração de **token Bearer**, validade de **10 min**).
- **US003 – Produtos:** 100% dos critérios de aceite cobertos (autenticação obrigatória, nome único, **PUT cria**, bloqueio de exclusão quando em carrinho).

Observação: **/carrinhos** foi coberta como **rota de apoio** para validar a regra “produto não pode ser excluído se estiver em carrinho”. Essa rota não possui User Story no anexo; portanto, o escopo foi **parcial** e orientado pela dependência de Produtos.

2) Cobertura por operações (CRUD / Fluxos)

Categoria	Universo	Coberto	Cobertura
Usuários – CRUD	Criar, Listar, Buscar, Atualizar (PUT), Deletar	Todos	**100%**

Produtos – CRUD	Criar, Listar, Buscar, Atualizar (PUT), Deletar	Todos	**100%**
Login	Autenticação (válida/INVÁLIDA), expiração	Todos	**100%**
Carrinhos	Criar, Listar, Buscar, Deletar (consistência estoque)	Parcial (apoio)	**Cobertura essencial**

3) Cobertura de regras de negócio (recorte)

- **Validações de Usuários:** e-mail **único e válido**, **bloqueio gmail/hotmail**, senha **5–10**, operações com **ID inexistente**, **PUT cria**.
- **Autorização:** rotas de **Produtos/Carrinhos** exigem token; falhas retornam **401/403**.
- **Produtos: nome único**, **PUT cria**, **não excluir** quando em **carrinho**.

11. TESTES CANDIDATOS À AUTOMAÇÃO

A automação deve focar em cenários **críticos e repetitivos**, cobrindo fluxos principais e validações de regras de negócio. Os testes foram mapeados para execução via **Postman/Newman** e podem futuramente ser integrados em pipelines de CI/CD.

Usuários (/usuarios)

- Criar usuário válido (U001).
- Impedir e-mail duplicado (U002).
- Validar bloqueio de provedores proibidos (U003).
- Testar limites de senha (U004).

Login (/login)

- Login válido → geração de token (L001).
- Login inválido → senha incorreta (L003).
- Login inválido → usuário inexistente (L002).
- Token expirado (L005) *(alto valor para regressão de segurança)*.

Produtos (/produtos)

- Criar produto válido com token (P002).
- Impedir criação sem token (P001).
- Impedir nome duplicado (P003).
- Excluir produto em carrinho (P008).

Carrinhos (/carrinhos)

- Criar carrinho com produto existente (C001).
- Remover carrinho e restaurar estoque (C005).

Esses cenários foram escolhidos por:

1. **Alta criticidade** (autenticação, autorização, integridade de dados).
2. **Alta repetição/regressão** (CRUD de usuários/produtos, login).
3. **Viabilidade de automação no Postman** (verificação de status codes, corpo da resposta e variáveis de ambiente).