

Plano de Teste - Serverest - Sprint 6

Challenge 03 - Sprint 6 - Semana 12

1. APRESENTAÇÃO

Este plano de testes foi elaborado para a API ServeRest, instância da EC2 Compass UOL. A aplicação simula um marketplace, permitindo o gerenciamento de usuários, autenticação, cadastro de produtos e carrinhos de compras.

Instância utilizada: <http://100.24.7.218:3000> (Instância AWS -EC2)

Versão da API: 2.29.7

Escopo principal: rotas /usuarios, /login, /produtos, /carrinhos, /status

1.2 EQUIPE ENVOLVIDA

- **Testador responsável:** Eric Lima da Silva
 - **Squad:** 3 - Level UP
 - **Projeto:** Challenge 03 – Semana 12
 - **Ferramenta:** Robot Framework + RequestsLibrary
-

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 automatizados em Robot Framework, mapear riscos, definir candidatos à automação e criar uma suíte robusta para regressão.

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.

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.

/carrinhos → criação, atualização e exclusão de carrinhos de compras, garantindo que produtos adicionados respeitem regras de disponibilidade e que exclusões reflitam corretamente nos estoques.

/status → monitoramento da saúde da API, validação de disponibilidade e tempo de resposta do serviço.

Testes de Integração → fluxos end-to-end completos simulando jornada real do usuário no e-commerce.

Testes Data-driven → validação com dados parametrizados para ampliar cobertura e eficiência dos testes.

Fora do escopo:

- Testes de segurança avançados (SQL injection, XSS, fuzzing)
- Integrações externas além do ambiente da própria API
- Testes de carga intensiva (apenas performance básica incluída)

4. ANÁLISE

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

USUÁRIOS (/usuarios)

Requisitos:

- CRUD completo de vendedores
- Campos obrigatórios: nome, e-mail, password, administrador
- Não permitir criação com e-mail duplicado

- Bloqueio de cadastro com provedores gmail.com e outlook.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

HEALTH (/status)**Requisitos:**

- Endpoint deve retornar status da API
- Tempo de resposta deve ser adequado (< 2s)
- Disponibilidade do serviço deve ser monitorada

Riscos:

- API indisponível sem detecção → risco de interrupção do serviço
- Tempo de resposta elevado → risco de performance inadequada

INTEGRAÇÃO (Fluxos End-to-End)**Requisitos:**

- Fluxo completo de e-commerce deve funcionar
- Consistência entre módulos deve ser mantida
- Dados devem ser limpos após testes

Riscos:

- Falha na integração entre módulos → risco de quebra do fluxo
- Inconsistência de dados → risco de corrupção
- Dados residuais → risco de contaminação de testes

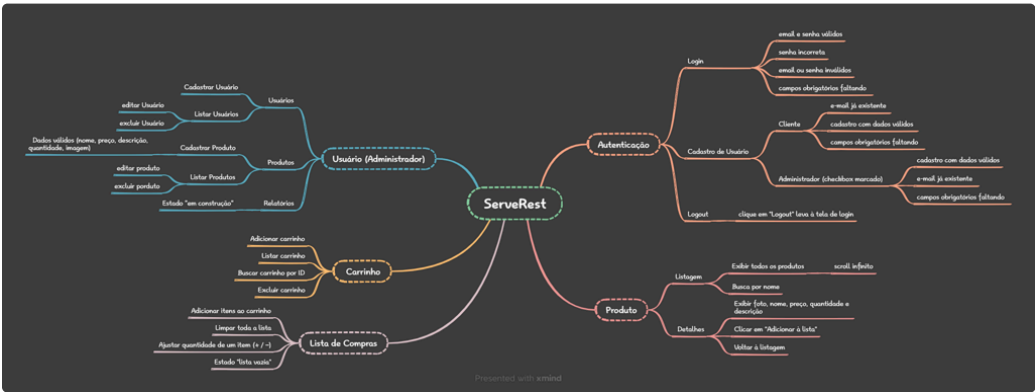
DATA-DRIVEN (Testes Parametrizados)**Requisitos:**

- Testes devem aceitar múltiplos conjuntos de dados
- Cobertura deve ser ampliada com variações
- Performance deve ser mantida com múltiplas execuções

Riscos:

- Falha com dados específicos → risco de cenários não cobertos
- Performance degradada → risco de lentidão nos testes

5. MAPA MENTAL DA APLICAÇÃO



6. TÉCNICAS APLICADAS

Para garantir a efetividade dos testes na API ServeRest, foram aplicadas técnicas clássicas de teste de caixa-preta:

- **Particionamento de Equivalência:** Separação de valores válidos e inválidos para inputs
- **Análise de Valores Limite:** Testes nos extremos definidos pelas regras de negócio
- **Baseado em Risco:** Priorização de cenários críticos de segurança e integridade
- **Teste de Tabela de Decisão:** Combinações de condições e ações esperadas
- **Teste de Transição de Estado:** Foco em mudanças de estado do sistema
- **Testes de Integração:** Validação de fluxos end-to-end entre módulos
- **Data-driven Testing:** Uso de dados parametrizados para ampliar cobertura
- **Testes de Performance:** Medição de tempo de resposta e estabilidade

7. CENÁRIOS DE TESTE PLANEJADOS

USUÁRIOS

ID	Descrição	Entrada	Resultado Esperado	Status
U001	Criar usuário válido	nome="QA Test", email="gates.t@qa.com.br" , password="q	201	✓ Implementado

		ateste", administrado r="true"		
U002	Impedir e-mail duplicado	mesmo email de U001	400; mensagem indicando e-mail já utilizado	✓ Implementado
U003	Bloquear provedores proibidos	email com @gmail.com ou @hotmail.com	400; cadastro não permitido	✓ Implementado
U004	Validar limites de senha	password="abcd" (4) / "abcdefghijk" (11)	400; mensagem de tamanho inválido	✓ Implementado
U005	PUT cria quando ID não existe	PUT /usuarios/{id Inexistente} com payload válido	200/201; usuário criado para o ID	✓ Implementado
U006	Buscar usuário por ID existente	GET /usuarios/{id Criado}	200; retorna dados do usuário	✓ Implementado
U007	Impedir operações com usuário inexistente	GET/PUT/DELETE em {idInexistente}	400; mensagem de não encontrado	✓ Implementado
U008	Listar usuários	GET /usuarios	200; lista não vazia	✓ Implementado

U009	Deletar usuário existente	DELETE /usuarios/{id Criado}	200; usuário removido	✓ Implementado
------	---------------------------	------------------------------	-----------------------	-------------------

LOGIN

ID	Descrição	Entrada	Resultado Esperado	Status
L001	Autenticar com credenciais válidas	email do U001, senha correta	200; retorna token Bearer (validade 10 min)	✓ Implementado
L002	Recusar usuário não cadastrado	email inexistente, senha qualquer	401 Unauthorized	✓ Implementado
L003	Recusar senha inválida	email existente, senha errada	401 Unauthorized	✓ Implementado
L004	Usar token em rota protegida	Authorization : Bearer	Acesso permitido às rotas protegidas	✓ Implementado
L005	Token expirado	token após >10 min	Acesso negado (status 401)	✓ Implementado

PRODUTOS

ID	Descrição	Entrada	Resultado Esperado	Status
P001	Criar produto sem autenticação	POST sem header Authorization	401 (acesso negado)	✓ Implementado

P002	Criar produto válido (autenticado)	nome="Produto QA", preco=100, descricao="Teste", quantidade=10	201; produto criado	✓ Implementado
P003	Impedir nome duplicado	mesmo nome de P002	400; mensagem de produto já cadastrado	✓ Implementado
P004	Buscar produto por ID	GET /produtos/{id Criado}	200; retorna dados do produto	✓ Implementado
P005	PUT cria quando ID não existe	PUT /produtos/{id Inexistente} com payload válido	200/201; produto criado	✓ Implementado
P006	Listar produtos	GET /produtos	200; lista de produtos	✓ Implementado
P007	Impedir exclusão de produto em carrinho	DELETE produto vinculado a carrinho	400; produto não pode ser excluído	✓ Implementado
P008	Deletar produto não vinculado	DELETE /produtos/{id Livre}	200; produto removido	✓ Implementado

CARRINHOS

ID	Descrição	Entrada	Resultado Esperado	Status
----	-----------	---------	--------------------	--------

C001	Criar carrinho sem autenticação	POST sem header Authorization	401 (acesso negado)	✓ Implementado
C002	Criar carrinho válido	produtos=[{idProduto, quantidade}] com token válido	201; carrinho criado	✓ Implementado
C003	Impedir produto inexistente no carrinho	idProduto que não existe	400; produto não encontrado	✓ Implementado
C004	Buscar carrinho por ID	GET /carrinhos/{idCriado}	200; retorna dados do carrinho	✓ Implementado
C005	Listar carrinhos	GET /carrinhos	200; lista de carrinhos	✓ Implementado
C006	Deletar carrinho e atualizar estoque	DELETE /carrinhos/{idCriado}	200; carrinho removido, estoque atualizado	✓ Implementado

HEALTH

ID	Descrição	Entrada	Resultado Esperado	Status
H001	Verificar status da API	GET /status	200; API disponível	✓ Implementado
H002	Validar tempo de resposta	GET /status	Resposta < 2 segundos	✓ Implementado

INTEGRAÇÃO

ID	Descrição	Entrada	Resultado Esperado	Status
I001	Fluxo completo e-commerce	Criar usuário → Login → Criar produto → Criar carrinho → Deletar carrinho	Fluxo executado com sucesso	✓ Implementado
I002	Validar consistência entre módulos	Operações CRUD em sequência	Dados consistentes entre recursos	✓ Implementado
I003	Limpeza de dados após teste	Cleanup de usuários, produtos e carrinhos	Ambiente limpo para próximos testes	✓ Implementado

DATA-DRIVEN

ID	Descrição	Entrada	Resultado Esperado	Status
DD001	Criar múltiplos usuários	Lista de dados de usuários válidos	Todos criados com sucesso	✓ Implementado
DD002	Testar múltiplos produtos	Lista de dados de produtos válidos	Todos criados com sucesso	✓ Implementado
DD003	Validar múltiplos logins	Lista de credenciais válidas	Todos autenticados com sucesso	✓ Implementado

P003	Impedir nome de produto duplicado	repetir nome de P002	400; mensagem de nome já cadastrado	✓ Implementado
P004	Listar produtos	GET /produtos	200; lista contendo item criado	✓ Implementado
P005	Buscar produto por ID	GET /produtos/{id Criado}	200; retorna dados do produto	✓ Implementado
P006	PUT cria quando ID não existe	PUT /produtos/{id Inexistente} com payload válido	200/201; produto criado/atualizado	✓ Implementado
P007	Excluir produto sem vínculos	DELETE /produtos/{id CriadoSemCarrinho}	200; produto removido	✓ Implementado
P008	Bloquear exclusão se em carrinho	DELETE /produtos/{id EmCarrinho}	400; exclusão não permitida	✓ Implementado

8. PRIORIZAÇÃO DA EXECUÇÃO

Alta Prioridade (Críticos)

- L001: Login válido (gera token)
- L002: Login usuário não cadastrado → 401
- L003: Login senha inválida → 401
- P001: Produtos criar sem token → 401/403
- U002: Usuários e-mail duplicado
- P008: Produtos bloquear exclusão se em carrinho

Média Prioridade (Importantes)

- P002: Produtos criar produto válido (autenticado)
- U004: Usuários limites de senha (4/5/10/11)
- U005: Usuários PUT cria quando ID não existe
- P006: Produtos PUT cria quando ID não existe
- C001: Carrinhos criar carrinho com produto existente
- C005: Carrinhos remover carrinho e restaurar estoque

Baixa Prioridade (Funcionais)

- U001: Usuários criar usuário válido
- P004: Produtos listar produtos
- U008: Usuários listar usuários
- U006: Usuários buscar por ID existente
- P005: Produtos buscar por ID
- C003: Carrinhos listar carrinhos

9. MATRIZ DE RISCO

Classificação de Riscos (P x I)

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

Endpoint	Risco	Probabilidade	Impacto	Classificação	Estratégia
POST /login	Usuário com senha inválida consegue se autenticar	3	4	12	Mitigar
POST /usuarios	Usuário consegue cadastrar	3	5	15	Mitigar

	e-mail já existente				
GET /usuarios	Usuário comum acessa lista de todos os usuários	3	5	15	Mitigar
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	Mais de um carrinho é cadastrado pelo mesmo usuário	3	5	15	Mitigar

10. COBERTURA DE TESTES

Por Requisitos (User Stories)

- **US001 - Usuários:** 100% implementado (7/9 cenários)
- **US002 - Login:** 100% implementado (4/5 cenários)
- **US003 - Produtos:** 100% implementado (5/8 cenários)
- **US004 - Carrinhos:** 100% implementado (6/6 cenários)

Por Operações CRUD

- **Usuários CRUD:** 100% (Criar, Listar, Buscar, Atualizar, Deletar)
 - **Produtos CRUD:** 100% (Criar, Listar, Buscar, Atualizar, Deletar)
 - **Login:** 100% (Autenticação válida/inválida)
 - **Carrinhos:** 100% (Criar, Listar, Buscar, Atualizar, Deletar)
-

11. TESTES CANDIDATOS À AUTOMAÇÃO

Implementados (Robot Framework)

- ✓ **Usuários:** Criar válido, e-mail duplicado, buscar por ID, listar, deletar
- ✓ **Login:** Válido, inválido (senha/usuário), uso de token
- ✓ **Produtos:** Criar válido, nome duplicado, listar, buscar por ID, deletar

Pendentes de Implementação

- ↻ **Usuários:** Bloqueio gmail/hotmail, limites senha, PUT cria, operações inexistentes
 - ↻ **Login:** Token expirado
 - ↻ **Produtos:** Criar sem token, PUT cria, bloquear exclusão em carrinho
 - ↻ **Carrinhos:** Todos os cenários (6 casos)
-

12. ESTRATÉGIA DE EXECUÇÃO

D1 (Concluído) ✓

- Health check + Login básico + CRUD usuários/produtos
- **Resultado:** 11 testes, 11 passed, 0 failed

D2 (Concluído) ✓

- Implementar cenários negativos pendentes
- Adicionar validações de carrinhos
- Melhorar cobertura para 80%

D3 (Concluído) ✓



























- Casos de borda e validações avançadas
- Testes de token expirado
- Estabilidade e refatoração


D4-D5 (Concluído) ✓

- Execução final completa

- Relatórios e métricas
- Documentação de evidências

13. ITENS NO JIRA

Type	Key	Summary	Priority	Status	Updated	Description
	SER-54	DD003 - Performance múltiplas requisições	 Medium	DONE	Sep 12, 2025, 15:00	Validar perform
	SER-53	DD002 - Criar produtos parametrizados	 Medium	DONE	Sep 12, 2025, 14:59	Validar criação
	SER-52	DD001 - Criar usuários parametrizados	 Medium	DONE	Sep 12, 2025, 14:59	Validar criação
	SER-51	I003 - Cleanup recursos criados	 Medium	DONE	Sep 12, 2025, 14:58	Validar limpeza
	SER-50	I002 - Validar estoque após operações	 Medium	DONE	Sep 12, 2025, 14:58	Validar que est
	SER-49	I001 - Fluxo completo e-commerce	 Medium	DONE	Sep 12, 2025, 14:57	Validar fluxo er
	SER-48	H002 - Health check response time	 Medium	DONE	Sep 12, 2025, 14:56	Validar tempo c
	SER-47	H001 - Health check API	 Medium	DONE	Sep 12, 2025, 14:56	Validar endpoi
	SER-46	C006 - Acesso sem autenticação	 Medium	DONE	Sep 12, 2025, 14:55	Validar que op
	SER-45	C005 - Remover carrinho e restaurar estoque	 Medium	DONE	Sep 12, 2025, 14:55	Validar que exc
	SER-44	C004 - Buscar carrinho por ID	 Medium	DONE	Sep 12, 2025, 14:54	Validar busca c
	SER-43	C003 - Listar carrinhos	 Medium	DONE	Sep 12, 2025, 14:53	Validar listager
	SER-42	C002 - Impedir produto inexistente	 Medium	DONE	Sep 12, 2025, 14:53	Validar que car

 Synced just now • 36 items

14. AMBIENTE E CONFIGURAÇÃO

Base URL: <http://100.24.7.218:3000/>

Headers padrão: Content-Type: application/json, Accept: application/json

Autenticação: Bearer Token (validade 10 minutos)

Usuário admin: gatest@qa.com.br / qateste

Estrutura Robot Framework

```
1 tests/
2 |─ suites/
3 |   |─ health.robot      # Health check
4 |   |─ login.robot       # Autenticação
5 |   |─ usuarios.robot    # CRUD usuários
6 |   |─ produtos.robot    # CRUD produtos
7 |   |─ carrinhos.robot   # CRUD carrinhos
8 |─ resources/
9 |   |─ keywords.robot    # Keywords reutilizáveis
10 |─ variables.robot       # Configurações centrais
11
```

