

Plano de testes melhorado - Mário

Plano de Testes: API ServeRest

- Melhoras se pautaram na seção 11, que trata das automações. Os demais detalhes estão no ciclo de testes Jira com o plugin QAlity.

Autor: Mário Queiroz

1. Apresentação

Este documento detalha o planejamento estratégico para os testes da API "ServeRest". O objetivo é guiar a equipe de Qualidade na validação das funcionalidades, regras de negócio e requisitos técnicos definidos para a aplicação. A abordagem visa garantir a estabilidade, segurança e confiabilidade da API.

Este plano servirá como a principal fonte de referência para a criação de cenários, execução de testes (manuais e automatizados) e report de resultados.

O ambiente testado foi: [ServeRest v2.29.7](#).

2. Objetivo

O objetivo principal deste plano de testes é **validar a qualidade e a conformidade da API ServeRest**, garantindo que todas as funcionalidades implementadas, especialmente as descritas nas User Stories (US001, US002, US003), se comportem conforme o esperado.

Objetivos Secundários:

- Identificar e documentar defeitos e inconsistências entre o comportamento da API e os critérios de aceitação.
- Garantir que as regras de negócio (ex: e-mail único, restrição de provedor, validação de senha) sejam rigorosamente aplicadas.
- Validar a segurança de acesso às rotas protegidas por autenticação.

- Criar uma base de testes de regressão automatizados para garantir a estabilidade em futuras alterações.

3. Escopo

3.1. Funcionalidades em Escopo:

Com base nas User Stories e no mapa mental, o escopo deste ciclo de testes inclui:

- **Módulo de Autenticação:**
 - Endpoint /login.
- **Módulo de Usuários:**
 - CRUD completo: GET, POST, PUT, DELETE para o endpoint /usuarios.
- **Módulo de Produtos:**
 - CRUD completo: GET, POST, PUT, DELETE para o endpoint /produtos.
- **Módulo de Carrinhos:**
 - Funcionalidades de listar, cadastrar e excluir carrinhos através dos endpoints /carrinhos, /carrinhos/concluir-compra e /carrinhos/cancelar-compra.

3.2. Funcionalidades Fora do Escopo:

As seguintes funcionalidades não serão o foco deste ciclo de testes:

- Módulo de Relatórios.
- Módulo de Lista de Compras (as funcionalidades de adicionar/remover, pois dependem de Carrinhos e não está mapeada a nível de API).
- Testes não-funcionais, como testes de Carga, Performance e Stress.
- Testes de Interface de Usuário (frontend).

4. Análise de Testes (Estratégia)

A estratégia de testes será baseada em uma análise multifacetada, utilizando os seguintes artefatos:

1. **Documentação Swagger:** Para entender a estrutura técnica dos endpoints, contratos (request/response) e parâmetros.
2. **User Stories (US001, US002, US003):** Para extrair os requisitos funcionais e as regras de negócio (Critérios de Aceitação), que são a principal fonte para os cenários de teste.
3. **Mapa Mental:** Para obter uma visão geral das funcionalidades e suas inter-relações.

A abordagem de testes cobrirá:

- **Testes Funcionais:** Validar se cada função da API executa sua tarefa corretamente (ex: se um POST /usuarios realmente cria um usuário).
- **Testes de Validação (Input Validation):** Verificar como a API lida com dados de entrada inválidos, nulos ou malformatados.
- **Testes de Regras de Negócio:** Foco em validar as restrições específicas (ex: "Não permitir cadastro com e-mail @gmail.com").
- **Testes de Segurança (Autenticação/Autorização):** Garantir que endpoints protegidos não possam ser acessados sem um token válido.

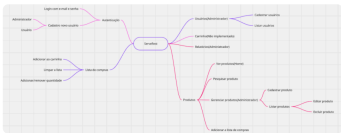
5. Técnicas de Teste Aplicadas

Para garantir uma cobertura abrangente, além do Session-Based Testing Management, as seguintes técnicas serão aplicadas:

- **Particionamento de Equivalência:** Agrupar dados de entrada em classes de equivalência válidas e inválidas (ex: e-mails válidos, e-mails inválidos, e-mails de provedores proibidos).
- **Análise de Valor Limite:** Testar os limites das regras de validação (ex: senhas com 4, 5, 10 e 11 caracteres).
- **Tabela de Decisão:** Utilizada para cenários com múltiplas condições, como a criação de um usuário (ex: é administrador? e-mail é único? senha é válida?).
- **Teste Baseado em Estado:** Validar as transições de estado da aplicação, como o fluxo "Usuário Deslogado → Realiza Login → Acessa Recurso Protegido → Realiza Logout".

6. Mapa Mental da Aplicação


A imagem a seguir, fornecida como base, ilustra a estrutura da aplicação ServeRest e as funcionalidades a serem testadas.



7. Cenários de Teste Planejados

A seguir, uma lista de cenários de alto nível planejados, agrupados por funcionalidade.

US001 - /usuarios

ID	Cenário	Resultado Esperado	Resultado obtido
US001-TC01	[Happy Path] Cadastrar um novo usuário (não administrador) com dados válidos.	Status 201 Created. O usuário é criado no banco de dados.	201 Created
US001-TC02	Cadastrar um usuário com e-mail já existente.	Status 400 Bad Request com a mensagem "Este email já está sendo usado".	400 Bad Request
US001-TC03	Cadastrar um usuário com e-mail em formato inválido (ex: "  t este.com ").	Status 400 Bad Request com mensagem de erro de validação de e-mail.	201 Created

US001-TC04	Cadastrar um usuário com senha fora do limite (ex: 4 caracteres).	Status 400 Bad Request com mensagem de erro sobre o tamanho da senha.	201 Created - Verificação do tamanho da senha falhou.
US001-TC05	Cadastrar um usuário com senha fora do limite (ex: 11 caracteres).	Status 400 Bad Request com mensagem de erro sobre o tamanho da senha.	201 Created - Verificação do tamanho da senha falhou.
US001-TC06	[Happy Path] Listar todos os usuários cadastrados.	Status 200 OK. Retorna uma lista com todos os usuários.	200 OK
US001-TC07	Buscar um usuário por um ID existente.	Status 200 OK. Retorna os dados do usuário específico.	200 OK
US001-TC08	Buscar um usuário por um ID inexistente.	Status 400 Bad Request com a mensagem "Usuário não encontrado".	400 Bad Request
US001-TC09	[Happy Path] Atualizar (PUT) um usuário existente com dados válidos.	Status 200 OK. Os dados do usuário são atualizados.	200 OK.
US001-TC10	Atualizar (PUT) um usuário informando um ID inexistente	Status 201 Created. Um novo usuário é criado com os	201 Created

	(deve criar um novo).	dados fornecidos.	
US001-TC11	Atualizar (PUT) para um e-mail que já está em uso por outro usuário.	Status 400 Bad Request com a mensagem "Este email já está sendo usado".	400 Bad Request
US001-TC12	[Happy Path] Deletar um usuário existente.	Status 200 OK com a mensagem "Registro excluído com sucesso".	200 OK
US001-TC13	Deletar um usuário inexistente	Status 400 "Nenhum registro excluído"	200 OK
US001-TC14	Acessar listagem de usuários sem permissão.	403 Unauthorized	200 OK
US001-TC15	Atualizar para um e-mail de domínio inválido.	400 Bad Request	200 OK
US001-TC16	Atualizar um usuário com senha fora do limite (ex: 4 caracteres).	Status 400 Bad Request com mensagem de erro sobre o tamanho da senha.	200 OK - Verificação do tamanho da senha falhou.
US001-TC17	Atualizar um usuário com senha fora do limite (ex: 11 caracteres).	Status 400 Bad Request com mensagem de erro sobre o tamanho da senha.	200 OK - Verificação do tamanho da senha falhou.

US002 - /login

ID	Cenário	Resultado Esperado	Resultado obtido
US002-TC01	[Happy Path] Realizar login com e-mail e senha corretos.	Status 200 OK. Retorna uma mensagem de sucesso e um Bearer token de autorização.	200 OK
US002-TC02	Tentar realizar login com um e-mail não cadastrado.	Status 401 Unauthorized.	401 Unauthorized
US002-TC03	Tentar realizar login com um e-mail cadastrado, mas senha incorreta.	Status 401 Unauthorized.	401 Unauthorized
US002-TC04	Tentar realizar login com campos de e-mail ou senha vazios.	Status 400 Bad Request com mensagem de erro sobre campos obrigatórios.	400 Bad Request

ID	Cenário	Resultado Esperado	Resultado obtido
US003-TC01	[Happy Path] Cadastrar um novo produto com dados válidos e token de autenticação válido.	Status 201 Created. O produto é criado com sucesso.	201 Created
US003-TC02	Tentar cadastrar um produto sem token de autenticação.	Status 401 Unauthorized com a mensagem "Token de acesso ausente, inválido ou expirado".	401 Unauthorized
US003-TC03	Tentar cadastrar um produto com nome já existente.	Status 400 Bad Request com a mensagem "Já existe produto com esse nome".	400 Bad Request
US003-TC04	[Happy Path] Listar todos os produtos cadastrados.	Status 200 OK. Retorna uma lista de produtos.	200 OK
US003-TC05	Atualizar (PUT) um produto informando um ID inexistente (deve criar um novo).	Status 201 Created. Um novo produto é criado com os dados fornecidos.	201 Created

US003-TC06	Atualizar (PUT) para um nome de produto que já está em uso por outro produto.	Status 400 Bad Request com a mensagem "Já existe produto com esse nome".	400 Bad Request
US003-TC07	[Happy Path] Deletar um produto existente (que não está em um carrinho).	Status 200 OK com a mensagem "Registro excluído com sucesso".	200 OK
US003-TC08	Tentar deletar um produto sem token de autenticação.	Status 401 Unauthorized.	401 Unauthorized
US003-TC09	Tentar deletar um produto que está associado a um carrinho.	Status 400 Bad Request com a mensagem "Não é permitido excluir produto que faz parte de carrinho".	400 Bad request
US003-TC10	Tentar deletar um produto inexistente	400 Bad Request	200 OK

Módulo - /Carrinho

ID	Cenário	Resultado Esperado	Resultado obtido
US004-TC01	Cadastrar um novo carrinho	Status 201 Created. Retorna	200 OK

	com produtos existentes e em quantidade disponível.	mensagem de sucesso e o _id do carrinho.	
US004-TC02	Tentar cadastrar um carrinho sem um token de autenticação válido.	Status 401 Unauthorized com mensagem "Token de acesso ausente, inválido ou expirado".	401 Unauthorized
US004-TC03	Tentar cadastrar um carrinho para um usuário que já possui um carrinho ativo.	Status 400 Bad Request com mensagem "Não é permitido possuir mais de 1 carrinho".	400 Bad Request
US004-TC04	Tentar cadastrar um carrinho adicionando um produto com idProduto inexistente.	Status 400 Bad Request com mensagem "Produto não encontrado".	400 Bad Request
US004-TC05	Tentar cadastrar um carrinho solicitando uma quantidade de produto maior que a disponível em estoque.	Status 400 Bad Request com mensagem "Produto não possui quantidade suficiente".	400 Bad Request
US003-TC06	Listar todos os carrinhos existentes (requer token de administrador).	Status 200 OK. Retorna uma lista com os carrinhos.	200 OK

US004-TC07	Buscar um carrinho específico por seu _id existente.	Status 200 OK. Retorna os detalhes do carrinho solicitado.	200 OK
US004-TC08	Buscar um carrinho por um _id inexistente.	Status 400 Bad Request com mensagem "Carrinho não encontrado".	400 Bad Request
US004-TC09	Concluir uma compra de um carrinho existente.	Status 200 OK com mensagem "Registro excluído com sucesso". O estoque do produto deve ser atualizado.	200 OK
US004-TC10	Cancelar uma compra de um carrinho existente.	Status 200 OK com mensagem "Registro excluído com sucesso". O estoque do produto não deve ser alterado.	200 OK
US004-TC11	Status 400 Bad Request com mensagem "Não foi encontrado carrinho para esse usuário".	Status 400 Bad Request com mensagem "Não foi encontrado carrinho para esse usuário".	400 Bad Request

US004-TC12	Tentar concluir/cancelar uma compra sem um token de autenticação válido.	Status 401 Unauthorized.	401 Unauthorized
------------	--	--------------------------	------------------

8. Priorização da Execução dos Cenários

A execução seguirá uma priorização baseada na criticidade da funcionalidade para o negócio.

• Prioridade Alta (Must Have):

- Cenários de "Happy Path" de todas as funcionalidades (/usuarios, /login, /produtos, /carrinho).
- Testes de autenticação e autorização (US002-TC01, US003-TC02).
- Validações críticas de negócio (e-mail único, nome de produto único).
- "Happy Path" do fluxo de compra (US004-TC01, US004-TC09).

• Prioridade Média (Should Have):

- Cenários de validação de dados (campos vazios, formatos inválidos, limites de senha).
- Fluxos alternativos (ex: PUT criando um novo registro).
- Restrições de negócio específicas (provedores de e-mail).
- Validações de regras de negócio de carrinhos (US004-TC03, US004-TC05).

• Prioridade Baixa (Could Have):

- Cenários de teste para casos de borda menos comuns.
- Testes exploratórios para encontrar falhas não previstas.
- Cenário de cancelamento de compra (US004-TC10).

9. Matriz de Risco

Risco Identificado	Probabilidade	Impacto	Nível de Risco	Estratégia de Mitigação
--------------------	---------------	---------	----------------	-------------------------

Falha no endpoint /login impede o acesso a todas as rotas protegidas.	Média	Alto	Crítico	Priorizar testes exaustivos no fluxo de login (US002).
Falha ao concluir compra gera inconsistência de estoque.	Média	Alto	Crítico	Testar o fluxo de compra e validar o estoque do produto antes e depois da transação.
Permitir cadastro de usuários com e-mails duplicados.	Média	Alto	Alto	Focar nos cenários US001-TC02 e US001-TC13.
Permitir acesso a rotas de produtos sem autenticação.	Baixa	Alto	Médio	Executar testes de segurança (US003-TC02, US003-TC08).
Violação da regra de negócio de provedores de e-mail.	Média	Baixo	Baixo	Garantir a execução dos cenários US001-TC03 e US001-TC04.

Falha ao deletar produto que está em um carrinho.	Média	Médio	Médio	Focar no cenário US003-TC09, simulando a pré-condição.
Permitir compra de produto com estoque insuficiente.	Média	Médio	Médio	Focar no cenário US004-TC05.

10. Cobertura de Testes

A cobertura será medida com base nos seguintes critérios:

- **Cobertura de Requisitos:** Garantir que 100% dos Critérios de Aceitação das User Stories (US001, US002, US003) sejam cobertos por pelo menos um cenário de teste. Perante o definido sobre as User Stories, a cobertura foi de 100%.
- **Path Coverage:** Garantir que a cobertura dos endpoints esteja totalmente abarcada nos testes. De acordo com o mapeado, são 9 caminhos, portando a cobertura atual é 100%.
- **Cobertura de API (Endpoints):** Garantir que todos os verbos (GET, POST, PUT, DELETE) dos endpoints em escopo sejam testados. Portanto, além de englobar as funcionalidades definidas pelas User Stories, os testes também cobriram carrinho. Ante o exposto, a cobertura foi de 100%.
- **Cobertura de Parâmetros:** Garantir que todos os parâmetros de entrada dos corpos de requisição sejam testados com dados válidos e inválidos. 100%.

11. Testes Candidatos a Automação

Testes CRUD (Smoke/Happy Path)

Conjunto mínimo de cenários automatizados para validar a saúde da aplicação após cada deploy, garantindo que o fluxo principal funcione de ponta a ponta. No Robot, foram implementados testes que cobrem:

- **Cadastro de Usuário** (POST `/usuarios`).
- **Login de Usuário** (POST `/login`).
- **Cadastro de Produto** com token de administrador (POST `/produtos`).
- **Exclusão de Produto** (DELETE `/produtos/{id}`).
- **Exclusão de Usuário** (DELETE `/usuarios/{id}`).

Testes de Regressão Funcional

Incluem todos os cenários de *Happy Path* e as principais regras de negócio críticas, garantindo que alterações no código não quebrem funcionalidades existentes. No Robot, foram automatizados:

- **Validação de e-mail único** no cadastro de usuários (erro 400 se duplicado).
- **Validação de nome de produto único** no cadastro de produtos.
- **Restrição de apenas 1 carrinho por usuário** (erro 400 se tentar criar outro).
- **Fluxos completos de atualização e exclusão** de usuários e produtos.

Testes de Contrato

Validações de schema e estrutura de resposta para garantir que o formato dos dados retornados pela API não seja alterado de forma inesperada. No Robot, foram criados testes que:

- Validam campos obrigatórios e tipos de dados nos responses de `/usuarios` e `/produtos` .
- Conferem mensagens e códigos de status esperados para cada operação.

Testes de Validação de Dados

Cenários que exploram entradas inválidas e comportamentos de erro, ideais para automação por serem repetitivos e de fácil reaproveitamento. No Robot, foram implementados testes para:

- **Login** com e-mail não cadastrado, senha incorreta, campos vazios e formato inválido.
- **Cadastro de usuário** com dados faltantes ou inválidos.
- **Cadastro de produto** sem token, com token de usuário comum ou com dados inválidos.

12. Testes automatizados na collection



13. Cronograma

ATIVIDADES	11/08	12/08	13/08	14/08	15/08	18/08
Produção do plano						

de testes						
Criação dos scripts de teste						
Execução dos Testes						
Produção do relatório de issues e melhorias						
Apresentação dos resultados						

14. Report de issues

15. Log

<div>SUITE Tests</div> <div> <div>Full Name: Tests</div> <div>Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests</div> <div>Start / End / Elapsed: 20250912 12:45:50.827 / 20250912 12:45:56.801 / 00:00:05.974</div> <div>Status: 15 tests total, 15 passed, 0 failed, 0 skipped</div> </div>	00:00:05.974	REPORT
<div>SUITE Login</div> <div> <div>Full Name: Tests.Login</div> <div>Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/login</div> <div>Start / End / Elapsed: 20250912 12:45:50.836 / 20250912 12:45:52.377 / 00:00:01.541</div> <div>Status: 5 tests total, 5 passed, 0 failed, 0 skipped</div> </div>	00:00:01.541	
<div>SUITE Login Test</div> <div> <div>Full Name: Tests.Login.Login Test</div> <div>Documentation: Cenários de teste para o endpoint /login, cobrindo a User Story US002</div> <div>Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/login/login_test.robot</div> <div>Start / End / Elapsed: 20250912 12:45:50.836 / 20250912 12:45:52.376 / 00:00:01.540</div> <div>Status: 5 tests total, 5 passed, 0 failed, 0 skipped</div> </div>	00:00:01.540	
<div>SETUP Setup Usuario Comum</div>	00:00:00.517	
<div>TEST US002-TC01: Realizar login com credenciais corretas</div>	00:00:00.189	
<div>TEST US002-TC02: Tentar realizar login com um e-mail não cadastrado</div>	00:00:00.183	
<div>TEST US002-TC03: Tentar realizar login com senha incorreta</div>	00:00:00.192	
<div>TEST US002-TC04: Tentar realizar login com campos vazios</div>	00:00:00.180	
<div>TEST US002-TC05: Tentar realizar login com formato de e-mail inválido</div>	00:00:00.189	
<div>SUITE Produtos</div> <div> <div>Full Name: Tests.Produtos</div> <div>Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/produtos</div> <div>Start / End / Elapsed: 20250912 12:45:52.377 / 20250912 12:45:54.321 / 00:00:01.944</div> <div>Status: 3 tests total, 3 passed, 0 failed, 0 skipped</div> </div>	00:00:01.944	
<div>SUITE Produtos Test</div> <div> <div>Full Name: Tests.Produtos.Produtos Test</div> <div>Documentation: Cenários de teste para o endpoint /produtos, cobrindo a User Story US003.</div> <div>Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/produtos/produtos_test.robot</div> <div>Start / End / Elapsed: 20250912 12:45:52.379 / 20250912 12:45:54.321 / 00:00:01.942</div> <div>Status: 3 tests total, 3 passed, 0 failed, 0 skipped</div> </div>	00:00:01.942	
<div>SETUP Realizar Login Como Administrador</div>	00:00:00.663	
<div>TEST US003-TC01: Cadastrar produto com token de administrador</div>	00:00:00.406	
<div>TEST US003-TC02: Tentar cadastrar produto sem token de autenticação</div>	00:00:00.185	
<div>TEST US003-TC03: Tentar cadastrar produto com nome já existente</div>	00:00:00.583	

- SUITE Usuarios		00:00:02.479
Full Name: Tests Usuarios		
Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/usuarios		
Start / End / Elapsed: 20250912 12:45:54.321 / 20250912 12:45:56.800 / 00:00:02.479		
Status: 7 tests total, 7 passed, 0 failed, 0 skipped		
- SUITE Usuarios Test		00:00:02.475
Full Name: Tests Usuarios Usuarios Test		
Documentation: Cenários de teste para o endpoint /usuarios, cobrindo a User Story US001.		
Source: /Users/mario/Documents/GitHub/Compass/ServeRestRobot/tests/usuarios/usuarios_test.robot		
Start / End / Elapsed: 20250912 12:45:54.323 / 20250912 12:45:56.798 / 00:00:02.475		
Status: 7 tests total, 7 passed, 0 failed, 0 skipped		
+ SETUP	Iniciar Sessao e Criar Usuario Base	00:00:00.507
+ TEARDOWN	Deletar Usuario Base	00:00:00.173
+ TEST	US001-TC01: Cadastrar usuário não administrador com dados válidos	00:00:00.375
+ TEST	US001-TC02: Tentar cadastrar um usuário com e-mail já existente	00:00:00.204
+ TEST	US001-TC06: Listar todos os usuários cadastrados	00:00:00.231
+ TEST	US001-TC09: Atualizar um usuário existente com dados válidos	00:00:00.202
+ TEST	US001-TC10: Atualizar (PUT) um usuário informando um ID inexistente (deve criar um novo)	00:00:00.384
+ TEST	US001-TC12: Deletar um usuário existente	00:00:00.195
+ TEST	US001-TC13: Tentar deletar um usuário inexistente	00:00:00.196