Plano de Testes da API ServeRest

Data: 30/09/2025

Autora: Andressa Von Ahnt

Este documento descreve a estratégia, o escopo e os cenários de teste para a API ServeRest, com o objetivo de garantir sua qualidade, funcionalidade e estabilidade, com foco nas regras de negócio e nas APIs REST.

1. Apresentação

Este plano de testes foi elaborado para validar a API ServeRest, com base na documentação do Swagger e nas User Stories fornecidas. O foco está em garantir que as APIs de Usuários, Login, Produtos e Carrinhos funcionem corretamente e atendam às regras de negócio. Ao final, serão comparados os cenários planejados com os de fato executados, documentando o que foi possível verificar.

2. Objetivo

O principal objetivo deste plano é garantir que a API ServeRest atenda aos requisitos definidos nas User Stories, identificando e reportando quaisquer falhas ou divergências entre o comportamento esperado e o real da aplicação, com o objetivo de contribuir para a qualidade do produto.

3. Escopo

Funcionalidades em Escopo

- **US 001 - Login:** Autenticação de usuários e geração de token.
- **US 002 - Usuários:** CRUD de usuários (Criação, Leitura, Atualização e Exclusão) e validações de dados.
- **US 003 - Produtos:** CRUD de produtos e validações de regras de negócio específicas, com controle de acesso via token de autenticação.
- **US 004 - Carrinhos:** CRUD de Carrinhos de Compras e validações de regras de negócio, com controle de acesso via token de autenticação.

Funcionalidades Fora de Escopo

- Testes de performance e carga.
- Testes de infraestrutura ou banco de dados.

4. Análise e Cenários de Teste Planejados

A análise foi baseada na documentação do Swagger e nas User Stories, cobrindo cenários positivos (comportamento esperado), negativos (comportamentos alternativos e de exceção) e de segurança.

- **Módulo: Login (US 001)**

ID	Método	Endpoint	Cenário de Teste	Pré-Condição	Tipo de Teste	Comportamento Esperado	Mensagem Esperada
L-001	POST	/login	**Login Válido** (E-mail e senha corretos)	Usuário válido cadastrado	Positivo	200 OK	"Login realizado com sucesso" Retornar `authorization`
L-002	POST	/login	**Login Inválido** (E-mail não cadastrado)	Nenhuma	Negativo / Funcional	401 Unauthorized	"Email não cadastrado"
L-003	POST	/login	**Login Inválido** (Senha incorreta)	Usuário cadastrado	Negativo / Funcional	401 Unauthorized	"Senha incorreta"

L-004	POST	/login	**Login Inválido** (Campos obrigatórios ausentes)	Nenhuma	Negativo	400 Bad Request	“Campos obrigatórios não foram preenchidos”

- **Módulo: Usuários (US 002)**

ID	Método	Endpoint	Cenário de Teste	Tipo de Teste	Comportamento Esperado	Mensagem Esperada
U-001	GET	/usuarios	Listar todos os usuários	Positivo	200 OK	Retorna array de usuários
U-002	GET	/usuarios/{_id}	Buscar usuário por ID existente	Positivo	200 OK	Retornar dados do usuário
U-003	GET	/usuarios/{_id}	Buscar usuário por ID inexistente	Negativo	400 Bad Request	"Usuário não encontrado"
U-004	POST	/usuarios	Cadastrar usuário válido	Positivo	201 Created	"Cadastro realizado com sucesso"
U-005	POST	/usuarios	Cadastrar com E-mail já utilizado	Negativo	400 Bad Request	"Este email já está sendo usado"

U-006	POST	/usuarios	Cadastrar com E-mail inválido (formato)	Negativo	400 Bad Request	"email deve ser um email válido"
U-007	POST	/usuarios	Cadastrar com E-mail de provedor restrito (Ex: @gmail.com)	Negativo / Regra	400 Bad Request	"Email de provedor restrito"
U-008	POST	/usuarios	Cadastrar com Senha Mínima Válida (5 caracteres)	Positivo / Borda	201 Created	"Cadastro realizado com sucesso"
U-009	POST	/usuarios	Cadastrar com Senha Mínima Inválida (4 caracteres)	Negativo / Borda	400 Bad Request	Erro de validação indicando limite
U-010	POST	/usuarios	Cadastrar com Senha Máxima Válida (10 caracteres)	Positivo / Borda	201 Created	"Cadastro realizado com sucesso"

U-011	POST	/usuarios	Cadastrar com Senha Máxima Inválida (11 caracteres)	Negativo / Borda	400 Bad Request	Erro de validação indicando limite
U-012	PUT	/usuarios/{_id}	Atualizar usuário existente	Positivo	200 OK	"Registro alterado com sucesso"
U-013	PUT	/usuarios/{_id}	Atualizar usuário com E-mail já usado por outro	Negativo	400 Bad Request	"Este email já está sendo usado"
U-014	PUT	/usuarios/{_id}	Criar novo usuário com PUT em ID inexistente (UPSERT)	Alternativo	201 Created	"Cadastro realizado com sucesso"
U-015	PUT	/usuarios/{_id}	Tentar criar novo usuário (UPSERT) com ID inexistente e e-mail já utilizado	Negativo	400 Bad Request	"Este email já está sendo usado"
U-016	DELETE	/usuarios/{_id}	Excluir usuário existente	Positivo	200 OK	"Registro excluído com sucesso"

			(sem carrinho)			
U-017	DELETE	/usuarios/{_id}	Excluir usuário inexistente	Negativo	200 OK ou 404 Not Found	Não causar erro crítico
U-018	DELETE	/usuarios/{_id}	Excluir usuário com carrinho associado	Negativo / Regra	400 Bad Request	Mensagem indicando que não pode excluir usuário com carrinho

- **Módulo: Produtos (US 003)**

ID	Método	Endpoint	Cenário de Teste	Tipo de Teste	Comportamento Esperado	Mensagem Esperada
P-001	GET	/produtos	Listar todos os produtos	Positivo	200 OK	Retorna array de produtos
P-002	POST	/produtos	Cadastrar produto válido (com token)	Positivo	200 OK	"Cadastro realizado com sucesso"
P-003	POST	/produtos	Tentar cadastrar sem Token de autenticação	Negativo / Segurança	401 Unauthorized	"Token de acesso ausente..."

P-004	POST	/produtos	Cadastrar produto com Nome já utilizado	Negativo / Regra	400 Bad Request	"Já existe produto com esse nome"
P-005	PUT	/produtos/{_id}	Atualizar produto existente (com token)	Positivo	200 OK	"Registro alterado com sucesso"
P-006	PUT	/produtos/{_id}	Atualizar produto com Nome já utilizado por outro	Negativo / Regra	400 Bad Request	"Já existe produto com esse nome"
P-007	PUT	/produtos/{_id}	Criar novo produto com PUT em ID inexistente (com token)	Alternativo	201 Created	"Cadastro realizado com sucesso"
P-008	PUT	/produtos/{_id}	Tentar atualizar/criar sem Token de autenticação	Negativo / Segurança	401 Unauthorized	"Token de acesso ausente..."
P-009	DELETE	/produtos/{_id}	Excluir produto existente (com token)	Positivo	200 OK	"Registro excluído com sucesso"
P-010	DELETE	/produtos/{_id}	Excluir produto	Negativo	400 Bad Request	Mensagem de erro (ex:

			inexistente (com token)			Produto não encontrado)
P-011	DELETE	/produtos/{_id}	Tentar excluir sem Token de autenticação	Negativo / Segurança	401 Unauthorized	"Token de acesso ausente..."
P-012	DELETE	/produtos/{_id}	Excluir produto que está em um carrinho	Negativo / Regra	400 Bad Request	"Não é possível excluir produto que faça parte de carrinho"

- **Módulo: Carrinhos (US 004)**

ID	Método	Endpoint	Cenário de Teste	Pré-Condição	Comportamento Esperado	Mensagem Esperada
C-001	GET	/carrinhos	Listar todos os carrinhos	Carrinhos existentes	200 OK	Retornar array de carrinhos
C-002	POST	/carrinhos	Criar carrinho válido	Token e ID de produto existente	201 Created	"Cadastro realizado com sucesso"
C-003	POST	/carrinhos	Tentar criar carrinho sem Token	Nenhuma	401 Unauthorized	"Token de acesso ausente..."

C-004	POST	/carrinhos	Tentar criar carrinho com ID de produto inexistente	Token válido	400 Bad Request	Mensagem de erro indicando ID inválido
C-005	DELETE	/carrinhos/concluir-compra	Concluir uma compra (esvaziar o carrinho)	Carrinho ativo e com itens, Token válido	200 OK	Mensagem de sucesso (ex: Compra concluída)
C-006	DELETE	/carrinhos/cancelar-compra	Cancelar uma compra (esvaziar o carrinho)	Carrinho ativo e com itens, Token válido	200 OK	Mensagem de sucesso (ex: Compra cancelada)
C-007	DELETE	/carrinhos/cancelar-compra	Tentar cancelar compra sem Token	Nenhuma	401 Unauthorized	"Token de acesso ausente..."

5. Técnicas Aplicadas

Teste Funcional

O foco principal é validar se o comportamento da API está em total conformidade com as User Stories e a documentação técnica. Isso envolve a verificação de que todas as operações de CRUD (Criação, Leitura, Atualização e Exclusão) funcionam conforme o esperado em todos os endpoints, como /usuarios, /produtos e /carrinhos.

Além disso, o teste funcional abrange a qualidade da resposta, confirmando que o status code retornado (ex: 200 OK, 201 Created) e o payload (corpo da resposta) são precisos e úteis.

Teste de Regressão

A reexecução de testes existentes é vital para garantir que as correções de bugs reportados não introduzam falhas em funcionalidades previamente estáveis. Esta técnica será aplicada ao Login e Cadastro, garantindo que as alterações no código para mitigar os issues não tenham quebrado a estabilidade da aplicação.

Teste de Segurança

Foco na validação dos mecanismos de acesso, que são de Prioridade Alta. Os testes de segurança garantem que as rotas restritas (POST, PUT, DELETE em /produtos e /carrinhos) são acessíveis somente com um token de autenticação válido. A execução verifica, por exemplo, o acesso a rotas protegidas sem o token, esperando o retorno de 401 Unauthorized.

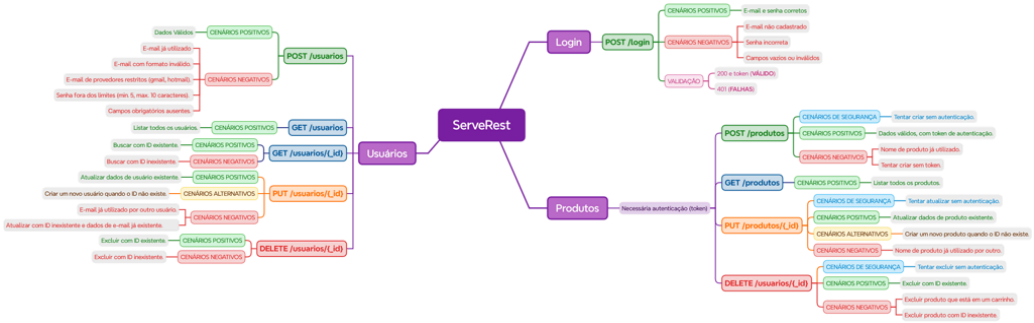
Além disso, deve-se testar regras críticas de negócio, como a impossibilidade de deletar um usuário ou produto que possua um carrinho associado.

Técnicas de Design de Testes

- **Particionamento por Equivalência:** Deve-se utilizar esta técnica para criar classes de dados que representam grandes conjuntos de testes, como testar uma senha ou um nome de produto válido em oposição a testar um nome de produto já existente ou um e-mail com domínio restrito. Isso reduz a redundância de testes ao cobrir todos os comportamentos esperados e inesperados.
- **Análise de Valor Limite:** Foco nos valores nos limites e próximos aos limites de um campo, como as senhas. Por exemplo, deve-se testar senhas com 5 e 10 caracteres (os limites válidos) e 4 e 11 caracteres (os limites inválidos).

6. Mapa Mental da Aplicação

O mapa mental abaixo ilustra a estrutura da API, seus endpoints e os cenários de teste planejados, incluindo as dependências entre as funcionalidades.



7. Priorização da Execução dos Cenários de Teste

1. **Prioridade Crítica:** Estes cenários são a base para o uso da aplicação e para a execução de todos os outros testes. Uma falha aqui impede o progresso total.
   - **Login e Geração de Token (L-001):** Essencial, pois o token de autenticação é o pré-requisito para testar todos os endpoints de Produto e Carrinho.
   - **Cadastro de Usuário Válido (U-004)**: A funcionalidade principal do módulo Usuários. É a fundação para os fluxos de ponta a ponta e para os testes de regressão automatizados.
   - **Acesso Negado (P-003, P-008, P-011, C-003, C-007):** Testes de Segurança que tentam acessar rotas protegidas (Produtos e Carrinhos) sem o token de autorização. A falha aqui representa um risco Crítico (Violação de Acesso).
2. **Prioridade Alta:** Cenários que validam regras de negócio essenciais e os limites de dados que foram foco de bugs (issues). São vitais para a integridade dos dados.
   - **Validação de Limites de Senha (Borda) (U-008, U-009, U-010, U-011):** Foco na validação das correções do Issue 1, testando os limites válidos (5 e 10 caracteres) e os inválidos (4 e 11 caracteres).
   - **Validação de E-mail Único e Restrito (U-005, U-007):** Garantir que e-mails já utilizados e e-mails de provedores restritos (@gmail, @hotmail) sejam rejeitados (Mitiga Issue 2).
   - **Criação de Produto Válido (P-002):** O fluxo principal do módulo Produtos, fundamental para os testes de Carrinhos.
   - **Criação de Carrinho Válido (C-002):** O ponto de entrada para o novo módulo Carrinhos e necessário para os testes de exclusão de usuário/produto.
   - **Conclusão de Compra (C-005):** Crítico para o negócio. Valida a finalização da transação, atualização de estoque e fluxo de pagamento.
3. **Prioridade Média:** Cenários que envolvem modificação e exclusão de dados já existentes, além da validação das mensagens de erro (Melhoria de Qualidade).
   - **Atualização (U-012, P-005):** Testes de `PUT` para Usuários e Produtos existentes.
   - **Cenários Alternativos (U-014, P-007):** Testar a criação de um novo recurso utilizando o método `PUT` em um ID inexistente.

- **Validação de Mensagens de Erro (L-002, L-003):** Execução dos cenários de Login Inválido para verificar se a **Melhoria 2** (mensagens de erro específicas como "Email não cadastrado" ou "Senha incorreta") foi implementada.

4. **Prioridade Baixa:** Testes de leitura e cenários negativos de menor impacto.
   - **Leitura/Listagem (U-001, U-002, P-001, C-001):** Testes de `GET` para listar todos e buscar por ID.
   - **Exclusão Negativa/Inexistente (U-017, P-010):** Tentar excluir IDs que não existem.
   - **Cenários de Finalização de Carrinho (C-005, C-006):** Cancelar Compra.

8. Matriz de Risco

Risco Potencial	Probabilidade	Impacto	Estratégia de Mitigação
Falha na validação de Senha ou E-mail	Média	Alto	Testes de Borda/Limite (5, 10, 4, 11 caracteres) e Testes de Domínio Restrito (gmail/hotmail) automatizados.
Falha na autenticação	Média	Crítico	Testes de regressão automatizados para login e logout.
Violação da regra de e-mail único	Média	Alto	Validar e-mails existentes antes de cada criação e atualização. Testar com massa de dados variada.
Criação de produto com nome já existente	Média	Alto	Implementar validação de nome de produto na requisição.
Acesso a rotas protegidas sem autenticação	Alta	Crítico	Testar todas as rotas de produtos sem o token de autorização.
Falha na Conclusão de Compra	Média	Crítico	Automação imediata do fluxo de criação de carrinho, adição de produto e

			conclusão de compra, para garantir integridade financeira e de estoque.
Exclusão de Usuário/Produto com carrinho ativo	Alta	Crítico	Validar a regra de negócio que impede a exclusão de qualquer recurso (Usuário ou Produto) que esteja ligado a um carrinho ativo.
Token expirado ou inválido	Média	Alto	Validar o comportamento da API com tokens inválidos e expirados.
Mensagens de Erro Genéricas (Login)	Média	Baixo	Incluir o requisito de Padronização de Mensagens de Erro nos casos de teste, esperando mensagens específicas.

9. Cobertura de Testes

A cobertura de testes para a API ServeRest é planejada para ser exaustiva, garantindo que todas as funcionalidades essenciais, regras de negócio e aspectos de segurança sejam verificados antes da entrega. A estratégia se baseia na garantia de que todas as regras de negócio das User Stories e funcionalidades do Swagger estão devidamente verificadas.

A cobertura planejada será detalhada nos seguintes níveis:

- **Cobertura Funcional (Caminhos da API)**
  - Garantindo que todos os caminhos (endpoints) da API serão percorridos. A validação será feita para todos os verbos HTTP (GET, POST, PUT, DELETE) em cada endpoint, assegurando que as funcionalidades de CRUD (Criação, Leitura, Atualização e Exclusão) operam conforme o esperado.
  - **Total de Endpoints em Escopo:** 8 (*/login*, */usuarios*, */usuarios/{_id}*, */produtos,* */produtos/{_id}*, */carrinhos*, */carrinhos/concluir-compra* e */carrinhos/cancelar-compra*).
  - **Cálculo da Cobertura de Caminhos da API REST (Planejada)**: Com a execução dos testes planejados, todos os 8 endpoints serão verificados. Portanto, a cobertura de caminhos planejada é de 100%.

- **Cobertura de Regras de Negócio**
  - Esta cobertura garante a integridade dos dados e o alinhamento com os requisitos do cliente (User Stories), com foco especial nos pontos de risco identificados:
    - **Validação de Dados:** Cobertura de cenários de Borda/Limite (ex: senhas de 4, 5, 10 e 11 caracteres).
    - **Restrições de Domínio:** Verificação da restrição de e-mails de provedores específicos (ex: gmail, hotmail).
    - **Unicidade**: Confirmação da validação de unicidade de e-mails de usuários e nomes de produtos.
    - **Integridade de Dados:** Validação das regras de negócio que impedem a exclusão de usuários ou produtos que possuam um carrinho ativo.
- **Cobertura de Segurança**
  - O foco é garantir o controle de acesso e a autorização na API.
    - **Autorização por Token:** Os testes garantem que as rotas de produtos e carrinhos são acessíveis somente com um token de autenticação válido e que o acesso sem autorização é negado.
    - Fluxo de Autenticação: Cobertura do ciclo de vida do token, incluindo sua geração (POST /login) e os cenários de token inválido ou ausente.
- **Cobertura de Qualidade**
  - **Padronização de Erros:** Cobertura dos cenários de falha para garantir que a API retorne mensagens específicas (ex: "Email não cadastrado" ou "Senha incorreta").

10. Testes Candidatos à Automação

A automação será focada nos cenários de **Prioridade Crítica (P1)** e **Alta (P2)** para garantir a regressão rápida e a mitigação de risco imediata, utilizando o Robot Framework.

1. **Fluxos de Estabilidade e Segurança (P1)**
   - **Login e Token:** Automatizar a geração e validação do *token* de autenticação, que é o pré-requisito para todos os testes de Produtos e Carrinhos.
   - **Acesso Negado (Segurança):** Verificar se a API retorna 401 Unauthorized ao tentar acessar rotas protegidas (`/produtos` , `/carrinhos`) sem enviar o token.
   - **CRUD de Usuário e Produto Válido:** Automatizar o ciclo completo de criação, leitura e exclusão para manter a base de dados limpa e estável.
2. **Validação de Regras de Negócio e Issues (P2)**

- **Validação de Borda e Unicidade:** Cobrir os testes de limites de senha (4,5,10,11 caracteres), além de validar a unicidade de e-mail e nome de produto.
- **Integridade Crítica (Bloqueio de Exclusão):** Automatizar a falha esperada ao tentar excluir um Usuário ou Produto que possui um carrinho ativo, garantindo a integridade dos dados.
- **Fluxo de Compra (Ponta a Ponta):** Automatizar a sequência Login → Criar Carrinho → Concluir Compra, validando a transação mais crítica do negócio.

3. **Qualidade de Resposta**
   - **Validação de Mensagens de Erro:** Automatizar a checagem do *corpo* da resposta em cenários negativos (ex: Login Inválido), garantindo que a API retorne mensagens específicas e claras ("Email não cadastrado", "Senha incorreta").