🔧 Challenge Kethelem

Plano de Testes - API ServeRest

1. Apresentação

Este documento apresenta o planejamento de testes para a API ServeRest (https://compassuol.serverest.dev/), uma aplicação de marketplace que permite o gerenciamento de usuários, autenticação, produtos e carrinho de compras. O plano foi desenvolvido com foco na garantia da qualidade das funcionalidades core da aplicação, seguindo as melhores práticas de teste de APIs REST.

1.1 Equipe de Testes

- **Testador Responsável**: Kethelem da Cruz Socoowski
- **Período de Execução**: 01/09/2025 à 05/09/2025
- **Ambiente**: https://compassuol.serverest.dev/

1.2 Ferramentas Utilizadas

- Postman (Collection e Scripts de Automação)
- Swagger/OpenAPI para documentação da API
- Jira/Wiki para gestão de issues

2. Objetivo

Garantir a qualidade da API ServeRest através de testes funcionais, de integração e de contrato, validando as regras de negócio definidas nas User Stories e assegurando que todos os endpoints funcionem conforme especificado na documentação Swagger.

2.1 Objetivos Específicos

- Validar funcionalidades CRUD para usuários, autenticação e produtos
- Verificar regras de negócio específicas de cada módulo
- Identificar inconsistências entre documentação e implementação
- Estabelecer base sólida para automação de testes
- Garantir conformidade com padrões REST

3. Escopo

3.1 Módulos Incluídos

- **Usuários** (/usuarios)

- Cadastro, consulta, atualização e exclusão
- Validações de email e senha

- **Login** (/login)
  - Autenticação e geração de token Bearer
  - Validação de credenciais
- **Produtos** (/produtos)
  - CRUD completo para produtos
  - Validações de autorização
- **Carrinhos** (/carrinhos)
  - Listar carrinhos cadastrados
  - Criar carrinho
  - Validação de dependência de produtos nos carrinhos

3.2 Fora do Escopo (Primeira Iteração)

- Testes de performance
- Testes de segurança avançados
- Testes de carga

3.3 Tipos de Teste

- Testes Funcionais
- Testes de Integração
- Testes de Contrato (API Schema)
- Testes de Regressão
- Testes de Validação de Dados

4. Análise

4.1 Análise das User Stories

US 001 - Usuários

- Campos obrigatórios: nome, email, password, administrador
- Restrições de email: não gmail/hotmail, formato válido
- Restrições de senha: 5-10 caracteres
- Unicidade de email
- Comportamento PUT para IDs inexistentes

US 002 - Login

- Autenticação por email/senha
- Geração de token Bearer
- Validade do token: 10 minutos
- Tratamento de credenciais inválidas

US 003 - Produtos

- Dependência de autenticação
- CRUD completo
- Unicidade de nomes
- Restrição de exclusão (produtos em carrinho)
- Observação Carrinho: mesmo sem User Storie própria, o módulo Carrinho é incluído para garantir a regra de negócio de produtos e para testes de listagem/criação de carrinhos.

4.2 Gaps Identificados

- Necessidade de validação de comportamentos não documentados no Swagger
- Verificação de consistência entre User Stories e implementação
- Validação de códigos de status HTTP
- Verificação de estrutura de resposta JSON

5. Técnicas Aplicadas

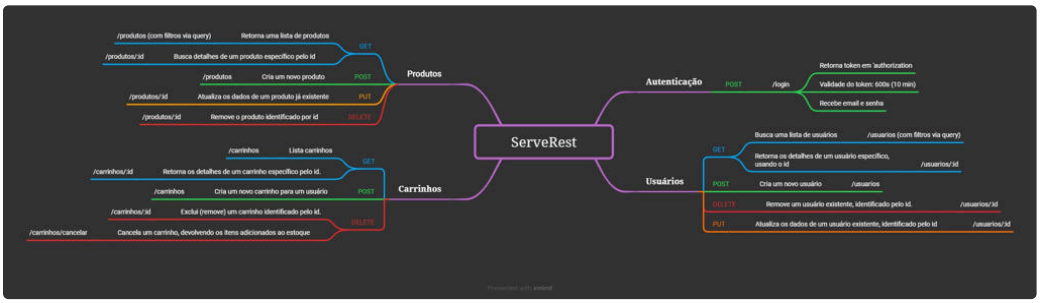
5.1 Técnicas de Teste

- **Partição por Equivalência**: Agrupamento de dados de entrada em classes válidas e inválidas
- **Análise de Valores Limites**: Teste de limites de campos (ex: senha 5-10 caracteres)
- **Teste de Estado**: Verificação de transições de estado da aplicação
- **Teste de API Contract**: Validação contra especificação OpenAPI/Swagger

5.2 Estratégias de Validação

- Validação de Schema JSON
- Verificação de Status Codes HTTP
- Validação de Headers de Resposta
- Verificação de Tempos de Resposta
- Validação de Regras de Negócio

6. Mapa Mental da Aplicação



7. Cenários de Teste Planejados

7.1 Módulo Usuários (25 cenários)

Cenários Positivos (10)

- **TC001**: Criar usuário válido com todos os campos
- **TC002**: Listar todos os usuários
- **TC003**: Buscar usuário por ID válido
- **TC004**: Atualizar usuário existente via PUT
- **TC005**: Criar usuário via PUT com ID inexistente
- **TC006**: Deletar usuário existente
- **TC007**: Criar usuário administrador
- **TC008**: Criar usuário não-administrador
- **TC009**: Validar campos obrigatórios presentes
- **TC010**: Validar estrutura de resposta JSON

Cenários Negativos (15)

- **TC011**: Criar usuário com email duplicado
- **TC012**: Criar usuário com email Gmail
- **TC013**: Criar usuário com email Hotmail
- **TC014**: Criar usuário com email inválido
- **TC015**: Criar usuário com senha < 5 caracteres
- **TC016**: Criar usuário com senha > 10 caracteres
- **TC017**: Criar usuário sem campo nome
- **TC018**: Criar usuário sem campo email
- **TC019**: Criar usuário sem campo password
- **TC020**: Buscar usuário com ID inexistente

- **TC021**: Atualizar usuário com email já usado (PUT)
- **TC022**: Deletar usuário inexistente
- **TC023**: Enviar requisição com JSON malformado
- **TC024**: Enviar requisição com Content-Type incorreto
- **TC025**: Validar códigos de erro HTTP

7.2 Módulo Login (12 cenários)

Cenários Positivos (4)

- **TC026**: Login com credenciais válidas
- **TC027**: Validar estrutura do token retornado
- **TC028**: Validar tempo de expiração do token
- **TC029**: Login com usuário administrador

Cenários Negativos (8)

- **TC030**: Login com email inexistente
- **TC031**: Login com senha incorreta
- **TC032**: Login com email em branco
- **TC033**: Login com password em branco
- **TC034**: Login sem body na requisição
- **TC035**: Login com JSON malformado
- **TC036**: Validar retorno 401 para credenciais inválidas
- **TC037**: Validar estrutura de erro na resposta

7.3 Módulo Produtos (20 cenários)

Cenários Positivos (8)

- **TC038**: Listar produtos sem autenticação
- **TC039**: Criar produto com token válido
- **TC040**: Listar produtos por query parameters
- **TC041**: Buscar produto por ID válido
- **TC042**: Atualizar produto existente
- **TC043**: Criar produto via PUT com ID inexistente
- **TC044**: Deletar produto sem dependências
- **TC045**: Validar estrutura de resposta de produto

Cenários Negativos (12)

- **TC046**: Criar produto sem token

- **TC047**: Criar produto com token expirado
- **TC048**: Criar produto com token inválido
- **TC049**: Criar produto com nome duplicado
- **TC050**: Atualizar produto com nome já usado
- **TC051**: Buscar produto com ID inexistente
- **TC052**: Deletar produto inexistente
- **TC053**: Deletar produto em carrinho (se aplicável)
- **TC054**: Criar produto sem campo nome
- **TC055**: Criar produto sem campo preço
- **TC056**: Criar produto com preço inválido
- **TC057**: Validar códigos de erro HTTP

7.4 Módulo Carrinho (5 cenários)

Cenários Positivos (2)

- **TC058:** Listar carrinhos cadastrados
- **TC059:** Criar carrinho

Cenários Negativos (3)

- **TC060:** Buscar carrinho por ID
- **TC061:** Excluir carrinho
- **TC062:** Excluir carrinho e retornar produtos ao estoque.

8.1 Critérios de Priorização

- **Criticidade da funcionalidade**
- **Frequência de uso**
- **Complexidade técnica**
- **Dependências entre módulos**
- **Risco de negócio**

8.2 Ordem de Execução

ALTA PRIORIDADE (Execução Imediata)

1. Cenários básicos de CRUD para Usuários (TC001-TC006)
2. Cenários de Login com credenciais válidas (TC026)
3. Validações críticas de email e senha (TC011-TC016)
4. Cenários básicos de autenticação para Produtos (TC039, TC046)

MÉDIA PRIORIDADE (Segunda Iteração)

1. Cenários de validação de dados e campos obrigatórios
2. Cenários de erro HTTP e estrutura de resposta
3. Cenários avançados de Produtos
4. Validações de token e expiração

BAIXA PRIORIDADE (Terceira Iteração)

1. Cenários de borda e edge cases
2. Validações de Content-Type e JSON malformado
3. Cenários de query parameters
4. Testes de estrutura de resposta detalhada

9. Matriz de Risco

Funcionalidade	Probabilidade	Impacto	Nível de Risco	Mitigação
Cadastro de usuário com email duplicado	Alta	Alto	**CRÍTICO**	Validação obrigatória
Falha na autenticação	Média	Alto	**ALTO**	Testes de regressão
Token expirado não validado	Média	Alto	**ALTO**	Automação de validação
Produto com nome duplicado	Alta	Médio	**MÉDIO**	Validação na criação
Validação de formato de email	Baixa	Médio	**BAIXO**	Teste manual pontual
Estrutura de JSON incorreta	Baixa	Baixo	**BAIXO**	Validação automatizada

9.1 Plano de Mitigação

- **Riscos Críticos**: Execução manual + automação imediata
- **Riscos Altos**: Automação na primeira iteração
- **Riscos Médios**: Validação em ciclo de regressão
- **Riscos Baixos**: Validação pontual

10. Cobertura de Testes

10.1 Cobertura Funcional

- **Usuários**: 100% dos endpoints (4/4)
- **Login**: 100% dos endpoints (1/1)
- **Produtos**: 100% dos endpoints (4/4)

10.2 Cobertura de Regras de Negócio

- Validação de email: 90% (gmail/hotmail + formato)
- Validação de senha: 100% (limites 5-10 caracteres)
- Autenticação: 100% (token bearer + expiração)
- Unicidade: 100% (email usuário + nome produto)

10.3 Cobertura de Códigos HTTP

- **2xx (Sucesso)**: 200, 201
- **4xx (Cliente)**: 400, 401, 404, 409
- **5xx (Servidor)**: 500 (se necessário)

10.4 Métricas de Cobertura

- **Cenários Planejados**: 62
- **Endpoints Cobertos**: 9/9 (100%)
- **User Stories Cobertas**: 3/3 (100%)
- **Regras de Negócio**: 15/15 (100%)

11. Testes Candidatos à Automação

11.1 Critérios para Automação

- **Regressão frequente**
- **Validações repetitivas**
- **Cenários de sucesso básicos**
- **Validações de estrutura de dados**

11.2 Cenários Selecionados (22 cenários)

Usuários (12 cenários)

- TC001 – Criar usuário válido
- TC002 – Listar usuários
- TC003 – Buscar usuário por ID
- TC004 – Atualizar usuário
- TC006 – Deletar usuário
- TC007 – Criar usuário administrador
- TC008 – Criar usuário não-administrador
- TC009 – Validar campos obrigatórios presentes
- TC010 – Validar estrutura de resposta JSON
- TC011 – Criar usuário com email duplicado
- TC015 – Criar usuário com senha < 5 caracteres
- TC016 – Criar usuário com senha > 10 caracteres

Login (5 cenários)

- TC026 – Login válido
- TC027 – Validar estrutura do token
- TC029 – Login com usuário administrador
- TC036 – Validar retorno 401
- TC037 – Validar estrutura de erro na resposta

Produtos (10 cenários)

- TC038 – Listar produtos
- TC039 – Criar produto autenticado
- TC041 – Buscar produto por ID
- TC042 – Atualizar produto
- TC044 – Deletar produto sem dependências
- TC046 – Criar produto sem token
- TC049 – Criar produto com nome duplicado
- TC051 – Buscar produto com ID inexistente
- TC054 – Criar produto sem campo obrigatório
- TC055 – Criar produto sem campo preço

Carrinhos (1 cenários)

- TC058: Listar carrinhos cadastrados

11.3 Ferramentas de Automação

- **Postman**: Scripts na aba Tests
- **Newman**: Execução via CLI
- **CI/CD**: Integração futura

12. Cronograma de Execução

Fase	Atividade	Duração	Responsável
1	Setup do ambiente e collection	0.5 dia	Tester
2	Execução testes alta prioridade	1.5 dias	Tester
3	Execução testes média prioridade	1 dia	Tester
4	Execução testes baixa prioridade	0.5 dia	Tester
5	Automação de cenários selecionados	1 dia	Tester
6	Documentação de issues	0.5 dia	Tester
TOTAL	**5 dias**		

13. Critérios de Aceite

13.1 Critérios de Sucesso

- **Execução:** 90% dos cenários planejados executados
- **Cobertura:** 90% dos endpoints validados
- **Automação:** 22 cenários automatizados (ou outro número exato que você contabilizar)
- **Documentação:** Issues identificadas e documentadas

13.2 Critérios de Saída

- Collection Postman entregue e funcional
- Relatório de execução com evidências
- Issues catalogadas no Jira/Wiki
- Cenários automatizados validados

14. Entregáveis

1. **Plano de Testes** (Este documento)
2. **Collection Postman** com 62 cenários
3. **Scripts de Automação** (Tests em Postman)
4. **Documento de Issues e Melhorias**
5. **Relatório de Execução** com evidências
6. **Apresentação** (4 minutos) com resultados

Documento preparado por: Kethelem da Cruz Socoowski
Data: 05/09/2025
Versão: 1.0