LUCAS - APRESENTAÇÃO

1 - Apresentação:

O objetivo é apresentar o plano de testes elaborado para a API ServeRest, com foco em ***garantir a qualidade da aplicação por meio de análise das regras de negócio*** e da validação das funcionalidades REST. A API simula um marketplace de produtos, onde vendedores podem se cadastrar, autenticar e gerenciar seus produtos.

A analise considero os endpoints documentados no Swagger da aplicação, bem como as User Stories fornecidas, que definem critérios de aceitação essenciais para validar o comportamento esperado. As técnicas aplicadas neste plano visam identificar falhas, inconsistências e oportunidades de melhoria, garantindo uma entrega mais robusta e confiável.

2 - Objetivo:

Assegurar que os principais fluxos de uso da API ServeRest funcionem conforme as especificações fornecidas, com foco em suas funcionalidades:

- Cadastro, autenticação e gestão de usuários (vendedores);
- Autenticação via login e geração de token;
- Cadastro, edição, listagem e exclusão de produtos.

Esse plano visa validar os caminhos felizes (Fluxos esperados) quanto os alternativos (erros, dados inválidos, comportamentos inesperados), garantindo que:

- As regras de negócio sejam atendidas;
- A comunicação da API siga os padrões REST;
- O retorno de dados e status HTTP estejam adequados aos cenários testados;
- As falhas sejam documentadas como issues com sugestões de melhoria.

Alem disso, o plano prevê testes manuais via Postman, evidências de execução e sugestões de automação para testes de regressão.

3 - Escopo:

Este plano de testes ***contempla a análise e validação dos principais endpoints*** expostos pela API ServeRest, com base no Swagger oficial e nas User Stories fornecidas. O escopo foi deficino conforme os critérios de aceitação, funcionalidades críticas e regras de negócio associadas as rotas abaixo:
FUNCIONALIDADES INCLUÍDAS NO ESCPOPO:

- / usuarios
  - Cadasro, atualização, listagem e remoção de usuários (CRUD)
  - Restrições de e-mail, senha e validação de campos obrigatórios
- / login
  - Autenticação com geração de token Bearer
  - Validação de login com dados inválidos, senhas erradas ou usuários inexistentes
- / produtos
  - Cadastro, edição, listagem e exclusão de produtos (CRUD)
  - Restrições de nomes duplicados e ações sem autenticação
  - Validação de dependência com carrinhos (para exclusão de produtos)

FORA DO ESCOPO:

- Teste de performance e carga
- Teste de segurança profunda
- Integração com sistemas externos ou banco de dados diretamente

FERRAMENTAS E RECURSOS UTILIZADOS:

- Swagger oficial da API: https://compassuol.serverest.dev/
- Postman para criação e execução dos testes
- Jira para registro de issues e melhorias
- Documentação em Confluence para apresentação do plano de testes

4 - Análise:

A analise a seguir foi ***realizada com base no Swagger da API ServeRest e nas User Stories fornecidas.*** Foram identificados os principais requisitos fincionais e regras de negócio que devem ser validadas durante os testes:

US001 - Cadastro de Usuários (/usuarios)
Regras de negócio identificadas:

- Um usuario deve conter os campos: nome, email, password, administrador
- O e-mail deve ser único
- Não pode aceitar e-mails com domínios gmail.com ou hotmail.com
- A senha deve ter entre 5 e 10 caracteres
- Caso o PUT seja usado com um ID inexistente, deve criar um novo usuário
- PUT não pode sobrescrever um usuário com e-mail já utilizado
- Ações sobre usuários inexistentes devem ser tratadas com erro aproriado

- Todos os testes devem conter evidências de execução

US002 - Login (/login)
Regras de negócio identificadas:

- Login só deve funcionar para usuários com senha correta
- Senha incorreta ou usuário inexistente deve retornar *401 Unauthorized*
- Login valido deve gerar token Bearer
- Token deve ter validade de 10 minutes
- Não deve haver retorno de dados sensíveis
- É essencial validar também comportamentos alternativos (ex: campos vazios)

US003 - Produtos (/produtos)
Regras de negócio identificadas:

- Ações do CRUD só devem ser possíveis para usuarios autenticados
- Não pode cadastrar produtos com nomes duplicados
- Produtos em carrinhos não podem ser excluidos
- Se o PUT for usado com um ID inexistente, deve criar novo produto
- PUT não pode criar produto com nome já existente
- Todos os testes devem conter evidências
- A cobertura deve ir além do Swagger, incluir cass inválidos, campos faltantes, erros esperados

PONTOS CHAVES IDENTIFICADOS:

- Existem validações específicas de dados (e-mail, senha, nomes únicos)
- Há cenários de ateticação obrigatória para segurança
- A dependência entre APIs (Produtos → Carrinhos) exige atenção especial

Esta analise nos dá base para desenhar cenários de teste realistas e alinhados ás regras de negócio.

5 - Técnicas a serem aplicadas:

PARTICIONAMENTO DE EQUIVALÊNCIA:

- Utilizado para validar faixas válidas e inválidas de entrada.
- Exemplo: testar senhas com menos de 5 e mais de 10 caracteres, e-mails com domínios proibidos (invalidos) e válidos (aceitos).

ANÁLISE DO VALOR LIMITE:

- Focada em testar os limites extremos das regras, como:
  - Senhas com exatamente 5 e 10 caracteres (limites validos)
  - Senhas com 4 e 11 caracteres (limites inválidos)

TABELA DE DECISÃO:

- Aplicada para verificar combinações de regras de negócio, como:
  - Usuário já existente + e-mail repetido + PUT = erro
  - Produto com nomerepetido + sem autenticação = erro

TESTE BASEADO EM REQUISITOS:

- Todos os testes foram mapeados com base direta nos critérios das User Stories, garantindo alinhamento com os critérios de aceitação e o DoD.

TESTE DE API COM FOCO EM COMPORTAMENTO (Black Box)

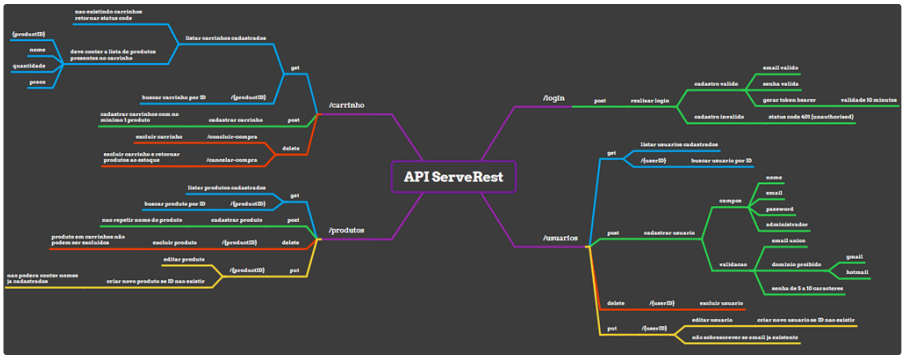
- As validações foram feitas sem acessar a lógica interna da aplicação, analisando apenas:
  - Requisições enviadas
  - Respostas HTTP (status, corpo headers)
  - Regas de negócio esperadas

TESTE EXPLORATÓRIO (Manual no Postman)

- Foram realizados testes livres, sem script definido, buscando identificar:
  - Comportamentos inesperados
  - Mensagens de erro genéricas ou inconsistentes
  - Reações da API e inputs extremos ou vazios

Essas técnicas ajudam a garantir uma cobertura equilibrada, focando tanto em cenários esperandos quanto em situações de falha ou uso incorreto da API.

6 - Mapa Mental:



7 - Cenários de teste planejados:

/USUARIOS

1. Criar usuário com dados válidos:
   Objetivo: Verificar se é possível cadastrar um novo usuário com todos os campos obrigatórios validos
   Entrada: nome, email valido, senha entre 5 e 10 caracteres, administrador (true ou false)
   Resultado esperado: Status 201 Created, usuário cadastrado com sucesso.
2. Criar usuario com email já existente
   Objetivo: Validar que não é permitido criar dois usuários com o mesmo email.
   Resultado esperado: Status 400 Bad Request, com mensagem de erro informando email duplicado.
3. Criar usuário com domínio de email proibido (gmail/hotmail)
   Objetivo: Garantir rejeição do sistema aos dominios
   Resultado esperado: Status 400 Bad Request, com mensagem específica para dominios invalidos
4. Criar um usuário com email invalido (sem “@” e caracteres invalidos, etc.)
   Resultado esperado: Status 400, e mensagem indicando email inválido
5. Criar usuário com senha fora dos limites permitidos (<5 ou >10 caracteres)
   Resultado esperado: Rejeição da requisição com mensagem de erro
6. Listar todos os usuários cadastrados
   Objetivo: Verificar se a roda GET /usuarios retorna uma lista valida
   Resultado esperado: Status 200 OK, com array de usuários
7. Buscar usuário por ID existente
   Objetivo: Validar busca individual por ID
   Resultado esperado: Status 200 OK, e dados corretos do usuário
8. Buscar por ID inexistente
   Resultado esperado: Satus 400 Bad Request ou 404 not Found, com mensagem de usuário não encontrado
9. Editar usuário existente com dados válidos
   Objetivo: Confirmar a utilização dos dados
   Resultado esperado: Status 200 OK, e campos atualizados
10. Editar usuário com email já utilizado
    Objetivo: Evitar duplicidade ao editar usuário
    Resultado esperado: Status 400, com mensagem de email já cadastrado
11. Editar usuário com ID inexistente
    Objetivo: verificar como o sistema se comporta
    Resultado esperado: Status 404 not Found, com mensagem de usuario nao identificado
12. Deletar usuário existente
    Resultado esperado: Status 200 OK, e com confirmação de exclusão
13. Tentar deletar usuário inexistente
    Resultado esperado: Status 400 Bad Request ou 404 not Found

Cenário ID	Prioridade	Justificativa	impacto	Risco
C-U001	Alta	Fluxo principal de cadastro	Alto	Moderado
C-U002	Alta	Impede duplicação de dados críticos	Alto	Baixo
C-U003	Média	Regras de negócio específicas	Moderado	Baixo
C-U004	Média	Validação básica crítica	Moderado	Baixo
C-U005	Média	Segurança e validação	Baixo	Moderado
C-U006	Média	Verificação de listagem geral	Moderado	Baixo
C-U007	Baixa	Verifica retorno individual esperado	Moderado	Baixo

C-U008	Baixa	Comportamento esperado com dados inválidos	Baixo	Baixo
C-U009	Média	Verifica se edições são possíveis	Alto	Baixo
C-U010	Alta	Evita duplicidade de e-mails	Moderado	Baixo
C-U011	Baixo	Fluxo negativo importante a validar	Baixo	Baixo
C-U012	Média	Operação de remoção	Moderado	Moderado
C-U013	Baixo	Validação negativa	Baixo	Baixo

/LOGIN

1. Login com credenciais válidas
   Objetivo: Verificar se um usuário com email e senha corretos consegue se autenticar com sucesso.
   Resultado esperado: Status 200 OK + token Bearer + mensagem "Login realizado com sucesso".
2. Login com e-mail não cadastrado
   Objetivo: Validar se o sistema bloqueia tentativas de login com e-mails não existentes.
   Resultado esperado: Status 401 Unauthorized, com mensagem "Email e/ou senha inválidos".
3. Login com senha incorreta
   Objetivo: Garantir que a autenticação falha quando a senha está errada.
   Resultado esperado: Status 401 Unauthorized, com mensagem "Email e/ou senha inválidos".
4. Login com campos obrigatórios ausentes
   Objetivo: Verificar como o sistema lida com payloads incompletos ou vazios.
   Resultado esperado: Status 400 Bad Request ou mensagem de erro sobre campos obrigatórios.
5. Verificação da estrutura e validade do token
   Objetivo: Confirmar que o token Bearer gerado expira após 10 minutos.
   Resultado esperado: Status 401 Unauthorized, com mensagem “Token expirado”

Cenário ID	Prioridade	Justificativa	impacto	Risco
C-L001	Alta	Fluxo principal	Alto	Alto
C-L002	Alta	Teste negativo essencial	Alto	Alto
C-L003	Alta	Teste negativo essencial	Alto	Alto
C-L004	Alta	Validação de campos obrigatórios	Alto	Alto
C-L005	Alta	Segurança e sessão	Alto	Alto

/PRODUTOS

1. Cadastro de produto com dados válidos e usuário autenticado
   Objetivo: Verificar se um usuário autenticado consegue cadastrar um produto com nome único, preço, descrição e quantidade válidos.
   Resultado esperado: Status 201 Created, com mensagem de sucesso e dados do produto criado.
2. Cadastro de produto com nome já utilizado
   Objetivo: Garantir que não seja possível cadastrar dois produtos com o mesmo nome.
   Resultado esperado: Status 400 Bad Request, com mensagem "Já existe produto com esse nome".
3. Cadastro de produto com usuário não autenticado
   Objetivo: Verificar se a API bloqueia o cadastro de produtos quando o usuário não envia um token válido.
   Resultado esperado: Status 401 Unauthorized, com mensagem de erro.
4. Edição de produto existente com dados válidos
   Objetivo: Verificar se um usuário autenticado pode editar os dados de um produto já existente.
   Resultado esperado: Status 200 OK, com mensagem de sucesso e dados atualizados.
5. Edição de produto inexistente (ID inválido)
   Objetivo: Confirmar que, se o produto com o ID informado não existir, um novo produto seja criado.
   Resultado esperado: Status 201 Created, com mensagem de criação.
6. Edição de produto com nome duplicado (nome já utilizado por outro produto)
   Objetivo: Garantir que não seja possível editar um produto para um nome já usado por outro.
   Resultado esperado: Status 400 Bad Request, com mensagem "Já existe produto com esse nome".
7. Exclusão de produto que está em carrinho
   Objetivo: Confirmar que produtos vinculados a carrinhos não podem ser excluídos.
   Resultado esperado: Status 400 Bad Request, com mensagem "Produto não pode ser excluído pois está em um carrinho".
8. Exclusão de produto que não está em carrinho
   Objetivo: Validar se um produto não relacionado a carrinhos pode ser removido com sucesso.

Resultado esperado: Status 200 OK, com mensagem "Produto excluído com sucesso".

9. Listagem de produtos
   Objetivo: Verificar se a API retorna a lista de produtos cadastrados.
   Resultado esperado: Status 200 OK, com array de produtos.
10. Listagem de produto por ID válido
    Objetivo: Validar o retorno correto de um produto específico ao consultar por ID.
    Resultado esperado: Status 200 OK, com dados do produto.
11. Listagem de produto por ID inexistente
    Objetivo: Garantir resposta adequada quando o ID buscado não existir.
    Resultado esperado: Status 404 Not Found, com mensagem "Produto não encontrado".

Cenário ID	Prioridade	Justificativa	impacto	Risco
C-P001	Alta	Fluxo principal	Alto	Alto
C-P002	Média	Validação de duplicidade	Alto	Moderado
C-P003	Alta	Segurança e controle de acesso	Alto	Alto
C-P004	Média	Operação comum de alteração	Moderado	Moderado
C-P005	Média	Validação negativa (inclusive confusa)	Baixo	Baixo
C-P006	Alta	Previne erros críticos	Baixo	Moderado
C-P007	Alta	Validação de regra de negócio	Alto	Alto
C-P008	Média	Remoção padrão	Moderado	Baixo
C-P009	Média	Fluxo de consulta comum	Moderado	Baixo
C-P010	Média	Consulta individual esperada	Moderado	Baixo
C-P011	Média	Resposta adequada em caso de erro	Baixo	Baixo

/CARRINHOS

1. Listar todos os carrinhos cadastrados
   Objetivo: Validar o retorno de todos os carrinhos existentes.
   Resultado esperado: Status 200 OK, lista de carrinhos com detalhes (produtos, quantidade, preço, etc).
2. Listar carrinhos quando não houver nenhum cadastrado
   Objetivo: Verificar o comportamento do sistema quando não há carrinhos registrados.
   Resultado esperado: Status 200 OK, corpo com array vazio ou mensagem apropriada.
3. Buscar carrinho por ID existente
   Objetivo: Garantir que o carrinho específico é retornado corretamente via ID.
   Resultado esperado: Status 200 OK, com dados corretos do carrinho.
4. Buscar carrinho com ID inexistente
   Objetivo: Validar a resposta ao buscar um carrinho inexistente.
   Resultado esperado: Status 404 Not Found, com mensagem de erro “Carrinho não encontrado”.
5. Criar carrinho com pelo menos um produto válido
   Objetivo: Garantir que o carrinho é criado corretamente com um ou mais produtos.
   Resultado esperado: Status 201 Created + carrinho com lista de produtos e totais.
6. Criar carrinho com lista vazia de produtos
   Objetivo: Testar a restrição de criação com payload inválido (sem produtos).
   Resultado esperado: Status 400 Bad Request, com mensagem “É necessário ao menos 1 produto”.
7. Concluir compra (DELETE /carrinhos/concluir-compra)
   Objetivo: Verificar se a compra finaliza corretamente e o carrinho é removido.
   Resultado esperado: Status 200 OK, com mensagem “Compra concluída com sucesso”
8. Cancelar compra (DELETE /carrinhos/cancelar-compra)
   Objetivo: Verificar se a compra é cancelada e o estoque dos produtos é restaurado.
   Resultado esperado: Status 200 OK, com mensagem “Compra cancelada com sucesso” + produtos de volta ao estoque.

Cenário ID	Prioridade	Justificativa	impacto	Risco
C-C001	Média	Consulta geral de carrinhos	Baixo	Baixo
C-C002	Média	Teste negativo importante	Baixo	Baixo
C-C003	Média	Fluxo esperado de consulta	Moderado	Moderado
C-C004	Média	Validação negativa	Baixo	Baixo

C-C005	Média	Fluxo principal	Baixo	Baixo
C-C006	Baixa	Regras de validação de payload	Baixo	Baixo
C-C007	Alta	Regra de negócio importante	Moderado	Moderado
C-C008	Alta	Reposição de estoque, lógica crítica	Moderado	Moderado

8 - Cobertura Minima:

- **Cobertura mínima de alto impacto**, em ordem de prioridade real:

Bloco	Testes mínimos para cobrir o fluxo principal	Justificativa
🟩 Cadastro (/usuarios)	Criar usuário com dados válidos	Essencial para iniciar o uso do sistema
🟩 Login (/login)	Login com credenciais válidas	Necessário para autenticação nas demais rotas
🟩 Produto (/produtos)	Cadastro de produto válido + Listagem	Sem produto, não há compra
🟩 Carrinho (/carrinhos)	Criar carrinho com produto válido	Início do processo de compra
🟩 Conclusão da compra	Concluir compra com sucesso	Final do fluxo principal e objetivo da API

Tabela de Execução de Testes Manuais

Nº	Rota	Cenário	Dados de Entrada	Resultado Esperado	Resultado Obtido	Status (Pass/Fail)
1	POST /usuarios	Criar usuário com dados válidos	nome, email válido, senha, admin	201 Created + dados do usuário	201 Created + ID gerado	✅ Pass
2	POST /login	Login com credenciais válidas	email + senha	200 OK + token	200 OK + token gerado	✅ Pass
3	POST /produtos	Cadastro de produto	nome, preço, descrição, qtd	201 Created + ID produto	201 Created + produto salvo	✅ Pass
4	POST /carrinhos	Criar carrinho com produto válido	ID produto, qtd	201 Created + resumo da compra	201 Created + ID gerado	✅ Pass
5	DELETE /carrinhos/concluir-compra	Concluir compra	token + carrinho criado	200 OK + mensagem sucesso	-	-

---

....
