Plano de Testes - API ServeRest - Challenge 1

1. Apresentação

Este documento apresenta o planejamento de testes para a API ServeRest (https://compassuol.serverest.dev/), visando garantir a qualidade das funcionalidades e o cumprimento das regras de negócio descritas nas User Stories.

2. Objetivo

Assegurar que a aplicação API ServeRest (https://compassuol.serverest.dev/)esteja funcionando conforme o esperado, explorando suas funcionalidades com base em heurísticas de qualidade, Users Stories. O foco está em descobrir falhas ocultas, comportamentos inesperados e inconsistências inclusive na documentação do Swagger.

3. Escopo dos Testes

Funcionalidades a serem testadas:

- CRUD de **Usuários**
- **Login** e autenticação
- CRUD de **Produtos**
- CRUD de **Carrinho**

4. Análise

Link para o Swagger Online: ServeRest

US 001 – [API] Usuários

Funcionalidades:

- CRUD de usuários (**/usuarios** e **/usuarios/{id}**)
- Campos obrigatórios: **nome, email, password, administrador**
- Restrições:
  - E-mail único (não permitir duplicidade com **POST** ou **PUT**)
  - E-mails de provedores **gmail** e **hotmail** não são permitidos
  - E-mails devem ter formato válido
  - Senha entre 5 e 10 caracteres

- Não é possível excluir usuários que possuam carrinhos
- **PUT** com ID inexistente cria novo usuário

- Possíveis respostas relevantes:
  - **201 Created**: cadastro com sucesso
  - **400 Bad Request**: e-mail duplicado ou usuário inexistente
  - **200 OK**: listagem, busca ou exclusão bem-sucedida

Riscos identificados:

- Falhas na validação de e-mail e senha permitindo dados inválidos
- Possibilidade de contornar a regra de exclusão de usuários com carrinho
- Falhas no tratamento de PUT criando duplicidades

Funcionalidades a serem exploradas:

- CRUD de Usuários
- Login e autenticação
- CRUD de Produtos
- CRUD de Carrinho

US 002 – [API] Login

Funcionalidades:

- Autenticação via **POST /login**
- Geração de token Bearer válido por 10 minutos
- Necessário para acessar rotas protegidas (produtos, carrinhos)
- Restrições:
  - Usuários não cadastrados não autenticam
  - Senha incorreta retorna **401 Unauthorized**
- Possíveis respostas relevantes:
  - **200 OK**: token gerado
  - **401 Unauthorized**: credenciais inválidas

Riscos identificados:

- Token não expirar corretamente
- Aceitar login com credenciais incorretas
- Token inválido não bloqueando acesso a rotas privadas

US 003 – [API] Produtos

Funcionalidades:

- CRUD de produtos **(/produtos e /produtos/{id})**
- Apenas administradores podem criar/editar/excluir
- Restrições:
  - Nome único (não permitir duplicidade com **POST** ou **PUT**)
  - Não excluir produtos vinculados a carrinhos
  - **PUT** com ID inexistente cria novo produto
- Possíveis respostas relevantes:
  - **201 Created**: produto criado
  - **400 Bad Request**: nome duplicado
  - **401 Unauthorized**: token inválido/ausente
  - **403 Forbidden**: acesso restrito a administradores

Riscos identificados:

- Permitir criação de produtos duplicados
- Permitir exclusão de produto em carrinho
- Falhas na verificação de permissão de administrador

Análise Geral do Swagger

- A API segue o padrão REST, com respostas em JSON e status codes claros.
- Endpoints possuem parâmetros bem definidos, aceitando **query params** e **path params**.
- Segurança baseada em token Bearer, com rotas protegidas especificadas via `security: ApiKeyAuth`.

Riscos Gerais

1. **Validação de entrada insuficiente** — entrada de dados fora dos padrões especificados.
2. **Falhas de autenticação/autorização** — acesso a rotas restritas sem token válido.
3. **Integridade de dados** — duplicidade de registros e inconsistência em exclusões condicionais.
4. **Expiração de sessão** — tokens não expirarem no tempo devido.

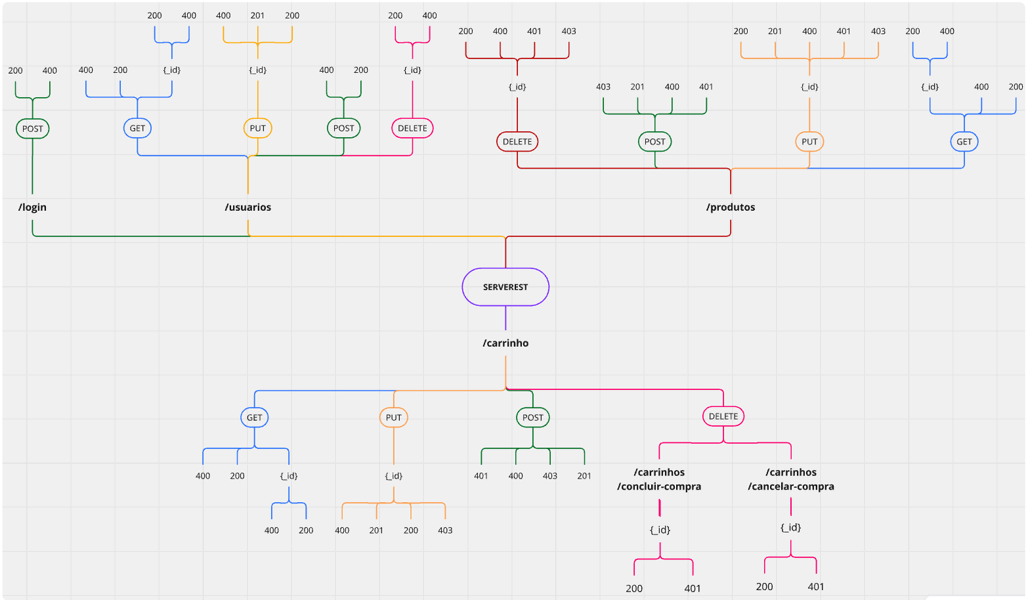
5. Técnicas Aplicadas

- **Caixa-Preta (Funcional)**: Verifica se os endpoints da API retornam as respostas esperadas, considerando apenas a entrada e saída, sem análise do código interno.

- **Análise de Valor-Limite:** Testa valores nos limites mínimo e máximo definidos para um campo, além de valores fora dos limites.
  - **Exemplo:**
    - Campo `senha` (4 a 12 caracteres):
      - 4 caracteres → **Inválido**
      - 6 caracteres → **Válido**
      - 11caracteres → **Inválido**
      - 10 caracteres → **Válido**
- **Testes End-to-end:** Rotina principal da aplicação.
- **Testes de Contrato (Swagger/OpenAPI):** Garante que as respostas da API estão de acordo com o contrato definido (Swagger/OpenAPI), prevenindo falhas de integração.

5. Mapa Mental da Aplicação

pb-compass/Documentos/MapasMentaisServeRest/ServeRest-API.jpg at main · MarceloSwap/pb-compass



Mapa mental - Status Code, Api ServeRest (https://compassuol.serverest.dev/)

7. Cenário de Teste Planejados

ID	Descrição do Cenário	Pré-condições	Dados de Entrada	Resultado Esperado	Prioridade
CT001	Criar usuário	—	Nome, e-mail	**201 Created** e ID gerado	Alta

	com dados válidos		válido, senha 6–10 caracteres, administrador=true/false		
CT002	Tentar criar usuário com e-mail já cadastrado	Usuário existente com e-mail específico	Mesmo e-mail	**400 Bad Request** com mensagem “Este email já está sendo usado”	Alta
CT003	Criar usuário com e-mail inválido	—	E-mail sem “@” ou domínio inválido	**400 Bad Request**	Média
CT004	Criar usuário com provedor `gmail` ou `hotmail`	—	E-mail `teste@gmail.com`	**400 Bad Request**	Média
CT005	Criar usuário com senha menor que 5 caracteres	—	Senha com 4 caracteres	**400 Bad Request**	Média

CT006	Criar usuário com senha maior que 10 caracteres	—	Senha com 11 caracteres	**400 Bad Request**	Média
CT007	Atualizar usuário com ID inexistente via PUT	—	ID inexistente e dados válidos	**201 Created** e novo ID	Alta
CT008	Excluir usuário com carrinho vinculado	Usuário autenticado com carrinho	ID do usuário	**400 Bad Request** com mensagem “Não é permitido excluir usuário com carrinho cadastrado”	Alta
CT009	Login com credenciais válidas	Usuário existente	E-mail e senha corretos	**200 OK** com token Bearer válido por 10 min	Alta
CT010	Login com senha incorreta	Usuário existente	E-mail correto, senha errada	**401 Unauthorized**	Alta
CT011	Login com usuário inexistente	—	E-mail não cadastrado	**401 Unauthorized**	Alta

CT012	Acessar rota protegida sem token	—	—	**401 Unauthorized**	Alta
CT013	Criar produto válido como administrador	Usuário autenticado e admin=true	Nome único, preço, descrição, quantidade	**201 Created** e ID gerado	Alta
CT014	Criar produto com nome duplicado	Produto existente	Mesmo nome	**400 Bad Request**	Alta
CT015	Criar produto sem token	—	Dados válidos	**401 Unauthorized**	Alta
CT016	Criar produto com usuário não administrador	Usuário autenticado admin=false	Dados válidos	**403 Forbidden**	Alta
CT017	Excluir produto vinculado a carrinho	Produto em carrinho	ID do produto	**400 Bad Request** com mensagem de bloqueio	Alta
CT018	Atualizar produto com ID inexistente via PUT	—	ID inexistente e dados válidos	**201 Created**	Média

8. Matriz de Risco

Risco	Probabilidade	Impacto	Nível	Observações
Criar usuário com e-mail duplicado não bloqueado	Alta	Alta	Crítico	Pode gerar inconsistência no banco de dados e duplicidade de contas
Falha na validação de formato de e-mail	Média	Alta	Alto	Permite cadastro com dados inválidos
Login aceitando credenciais inválidas	Baixa	Alta	Alto	Quebra de segurança e acesso indevido
Token não expirar corretamente	Média	Alta	Alto	Risco de sessões indefinidas e vulnerabilidade de segurança
Criar produto duplicado não bloqueado	Alta	Alta	Crítico	Pode impactar integridade e experiência do usuário
Exclusão de produto vinculado a carrinho	Alta	Alta	Crítico	Pode gerar carrinhos com itens inexistentes
Acesso a rotas restritas sem autenticação	Média	Alta	Alto	Vazamento ou alteração indevida de dados

Exclusão de usuário com carrinho vinculado	Alta	Alta	Crítico	Impacta relacionamento de dados e estoque

9. Cobertura de Testes

A cobertura será medida considerando:

- **Percentual de endpoints testados** (com base no Swagger)
- **Percentual de regras de negócio validadas** (com base nas User Stories)
- **Usando o critério Path Coverage (input)**

Abordagem:

1. Mapear todos os endpoints e métodos HTTP no Swagger.
2. Associar cada cenário de teste planejado a um endpoint e regra de negócio.
3. Registrar na execução:
   - Cenários executados com sucesso
   - Cenários falhos com evidências

Métrica Alvo:

- **Cobertura mínima de 80%** dos endpoints documentados.
- Cobrir **100% das regras de negócio críticas** (US 001, US 002, US 003).

Resultados obtidos:

100% endpoints testados (com base no Swagger)
90% das regras de negócio críticas (US 001, US 002, US 003).
100% para o caminho feliz baseano Path Coverage (input)

10. Documento com mapeamento de issues e melhorias

ID	Endpoint	Cenário	Resposta Atual	Problema	Severidade
ISS-01	DELETE /carrinhos/concluir-compra	Concluir compra com sucesso	200 OK + "Registro	Mensagem não reflete a ação de compra concluída.	Baixa

			excluído com sucesso "		
ISS-02	DELETE /carrinhos /cancelar-compra	Cancelar carrinho	200 OK + "Registro excluído com sucesso "	Mensagem deveria indicar claramente que os produtos foram devolvidos ao estoque.	Alta
ISS-03	PUT /produtos/ {id}	Criar produto inexistente via PUT	201 Created	Documentação não deixa claro que PUT pode criar novo produto.	Média
ISS-04	DELETE /produtos/ {id}	Produto associado a carrinho	400 Bad Request	Mensagem genérica. Não explica que produto não pode ser excluído porque está em carrinho.	Alta
ISS-05	POST /usuarios	Criar usuário com email Gmail ou Hotmail	201 Created (usuário cadastrado com sucesso)	Regra de negócio deveria bloquear cadastros com Gmail e Hotmail, mas API aceita normalmente.	Alta
ISS-06	POST /usuarios	Criar usuário com senha < 5	201 Created (usuário cadastr	Regra de negócio deveria rejeitar senha inválida, mas API aceita.	Alta

		ou > 10 caracteres	ado com sucesso)		

10. Documento com mapeamento de issues e melhorias

ID	Endpoint	Sugestão	Benefício
MELH-01	DELETE /carrinhos/concluir-compra	Alterar mensagem para: "Compra concluída com sucesso"	Alinhamento semântico com regra de negócio.
MELH-02	DELETE /carrinhos/cancelar-compra	Alterar mensagem para: "Compra cancelada. Produtos retornaram ao estoque"	Transparência para o cliente que consome a API.
MELH-03	PUT /produtos/{id}	Documentar claramente que pode criar um novo produto caso o ID não exista.	Evita confusão e falhas no uso da API.
MELH-05	Swagger Docs	Atualizar documentação de /carrinhos/{id} para explicitar que apenas GET existe.	Evita dúvidas e testes desnecessários.
ISS-06	POST /usuarios	Criar usuário com email Gmail	201 Created (usuário

		ou Hotmail	cadastrado com sucesso)
ISS-07	POST /usuarios	Criar usuário com senha < 5 ou > 10 caracteres	201 Created (usuário cadastrado com sucesso)

11. Testes Candidatos a Automação

Os seguintes cenários foram selecionados para automação no Postman por serem críticos, repetitivos ou de fácil validação automatizada:

US 001 – Usuários

- CT001 – Criar usuário com dados válidos
- CT002 – Criar usuário com e-mail já cadastrado
- CT004 – Criar usuário com provedor gmail/hotmail
- CT008 – Excluir usuário com carrinho vinculado

US 002 – Login

- CT009 – Login com credenciais válidas
- CT010 – Login com senha incorreta
- CT012 – Acessar rota protegida sem token

US 003 – Produtos

- CT013 – Criar produto válido como administrador
- CT014 – Criar produto com nome duplicado
- CT017 – Excluir produto vinculado a carrinho
- CT016 – Criar produto com usuário não administrador

Critérios de escolha:

- Alta criticidade para o negócio
- Repetibilidade nos ciclos de teste
- Validação clara via API (status code, corpo da resposta)

Alguns resultados:
ServeRest