Plano de Testes para a API ServeRest

---

1. Identificação do Plano de Testes:

- Nome do projeto: API ServeRest
- Responsável: Raique Alfredo Pereira de Ramos
- Data de criação: 06/05/2025

---

2. Objetivo do Plano:

Este plano de testes tem como objetivo garantir que a API ServeRest funcione conforme esperado e atenda aos critérios definidos nas User Stories e na documentação técnica do Swagger. Ele cobre as etapas de planejamento, análise, execução e documentação dos testes, com foco em identificar problemas, assegurar a segurança e usabilidade das funcionalidades, e priorizar cenários que podem ser automatizados para facilitar futuros testes.

---

3. Escopo dos Testes:

- [x] Testes funcionais dos Endpoints: login, usuarios, produtos e carrinhos.
- [x] Validação das regras de negócio.
- [x] Verificação de mensagens e status HTTP para respostas padrão e erros.
- [x] Testes de fluxos completos, como criação de usuários e autenticação
- [x] Cenários alternativos e negativos, como envio de dados inválidos ou duplicados.
- [x] Validação do tempo de expiração do token de autenticação.
- [x] Validação de segurança básica, como testar endpoints sem autenticação.
- [ ] Fora do escopo: Teste de carga e estresse, Integrações externas fora da API ServeRest, Testes de segurança avançados.

---

4. Análise da API:

A API ServeRest possui quatro grupos principais: Usuários, Login, Produtos e Carrinhos. Os testes serão conduzidos com foco nas regras de negócio descritas nas User Stories e no Swagger da API. A análise inclui os seguintes pontos:

- **Usuários:**
  - Testar CRUD completo (criação, atualização, leitura e exclusão) e validações, como restrições de e-mail (não aceitar provedores gmail/hotmail e verificar duplicidade) e senha (5-10 caracteres).
  - Garantir que usuários não existentes retornem respostas adequadas (criação em PUT quando ID não existe).
- **Login:**
  - Verificar autenticação com geração de token.
  - Garantir que tokens inválidos ou expirados retornem 401.
- **Produtos:**
  - Testar CRUD completo e validações, como restrição de nomes duplicados e dependência com carrinhos.
  - Testar fluxo completo de exclusão de produto, verificando impacto em carrinhos.
- **Carrinhos:**
  - Testar cadastro, listagem e exclusão, garantindo que a exclusão retorna produtos ao estoque.

- Testar a busca de carrinho por ID.

- **Dependências Identificadas:**
  - A execução de testes de produtos e carrinhos depende de autenticação bem-sucedida (Login).
  - A exclusão de carrinhos depende de regras específicas associadas a produtos.

Esses testes abordarão cenários críticos, como validação de dados, autenticação e restrições de negócios, além de fluxos completos que integram múltiplos endpoints.

---

5. Técnicas aplicadas:

- Testes de Caixa Preta para verificar entradas e saídas conforme especificações.
- Testes Baseados em Requisitos e Documentação (Swagger e User Stories).
- Particionamento de Equivalência e Análise de Valor Limite para validação de dados.
- Testes de Cenários Alternativos e Negativos para entradas inválidas e situações fora do padrão.
- Testes de Fluxo Completo para validar a integração entre funcionalidades.
- Verificação de Conformidade com o Padrão REST.

---

6. Mapa Mental da aplicação:

---

7. Cenários de Testes Planejados:

Cenários de Testes

Cenários de Testes para US 001 - [API] Usuários

ID: US001-CT01

Título: Criação de Usuário com Todos os Campos Válidos
Pré-condição: Nenhum usuário com os mesmos dados deve estar cadastrado no sistema.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload os campos `nome`, `email`, `password` e `administrador` com valores válidos.

Resultado Esperado:

- Retornar status HTTP 201 (Created).
- Resposta deve conter os dados do usuário criado, exceto a senha.

ID: US001-CT02

Título: Tentar Criar Usuário com E-mail Duplicado
Pré-condição: Um usuário com o mesmo e-mail deve estar previamente cadastrado.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload os campos `nome`, `email`, `password` e `administrador` com o mesmo e-mail já existente.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o e-mail já está em uso.

ID: US001-CT03

Título: Criar Usuário com E-mail Inválido
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload um e-mail em formato inválido, como “usuarioSemEmail”.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o e-mail é inválido.

ID: US001-CT04

Título: Criar Usuário com E-mail de Provedor Restrito
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload um e-mail de provedores como `gmail.com` ou `hotmail.com`.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o provedor do e-mail não é aceito.

ID: US001-CT05

Título: Criar Usuário com Senha Fora dos Limites de Caracteres
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload uma senha com menos de 5 caracteres ou mais de 10 caracteres.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que a senha deve conter entre 5 e 10 caracteres.

ID: US001-CT06

Título: Criar Usuário com Campo “nome” Vazio ou Ausente
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload com o campo `nome` vazio ou sem incluir o campo.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o campo `nome` é obrigatório.

ID: US001-CT07

Título: Criar Usuário sem Campo ADMINISTRADOR
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/usuarios`.
2. Enviar no payload omitindo o campo `administrador`.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o campo administrador é obrigatório.

ID: US001-CT08

Título: Buscar Usuário por ID Inexistente
Pré-condição: Nenhum usuário com o ID informado deve estar cadastrado no sistema.
Passos:

1. Fazer uma requisição GET na rota `/usuarios/{_id}` usando um ID inválido ou inexistente.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o usuário não foi encontrado.

ID: US001-CT09

Título: Atualizar Usuário Existente com Dados Válidos
Pré-condição: Deve existir um usuário previamente cadastrado.
Passos:

1. Fazer uma requisição PUT na rota `/usuarios/{_id}` usando o ID de um usuário existente.
2. Enviar no payload todos os campos obrigatórios com valores válidos.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Mensagem informando que o registro foi atualizado com sucesso.

ID: US001-CT10

Título: Atualizar Usuário com ID Inexistente
Pré-condição: O ID informado na requisição não deve corresponder a nenhum usuário cadastrado.
Passos:

1. Fazer uma requisição PUT na rota `/usuarios/{_id}` usando um ID inexistente.
2. Enviar no payload todos os campos obrigatórios com valores válidos.

Resultado Esperado:

- Retornar status HTTP 201 (Created).
- Criar um novo usuário com os dados fornecidos no payload.

ID: US001-CT11

Título: Tentar Atualizar Usuário com E-mail Duplicado
Pré-condição: Deve existir outro usuário previamente cadastrado com o mesmo e-mail.
Passos:

1. Fazer uma requisição PUT na rota `/usuarios/{_id}`.
2. Enviar no payload um e-mail já cadastrado em outro usuário.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o e-mail já está em uso.

ID: US001-CT12

Título: Atualizar Usuário sem Preencher Todos os Campos Obrigatórios
Pré-condição: Deve existir um usuário previamente cadastrado.
Passos:

1. Fazer uma requisição PUT na rota `/usuarios/{_id}`.
2. Enviar no payload omitindo um ou mais campos obrigatórios.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que campos obrigatórios estão faltando.

Cenários de Testes para US 002 - [API] Login

ID: US002-CT01

Título: Login com Credenciais Válidas
Pré-condição: O e-mail e senha utilizados devem estar previamente cadastrados.
Passos:

1. Fazer uma requisição POST na rota `/login`.

2. Enviar no payload os campos `email` e `password` válidos.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Resposta deve conter o token Bearer válido e mensagem informando que o login foi realizado.

ID: US002-CT02

Título: Tentar Login com E-mail Não Cadastrado
Pré-condição: O e-mail informado no payload não deve existir no sistema.
Passos:

1. Fazer uma requisição POST na rota `/login`.
2. Enviar no payload um `email` inexistente e uma senha qualquer.

Resultado Esperado:

- Retornar status HTTP 401 (Unauthorized).
- Mensagem de erro indicando que as credenciais são inválidas.

ID: US002-CT03

Título: Testar Expiração do Token Após 10 Minutos
Pré-condição: Realizar login válido para gerar o token.
Passos:

1. Fazer uma requisição POST na rota `/login` para obter o token Bearer.
2. Aguardar 10 minutos.
3. Tentar utilizar o token para acessar uma rota protegida.

Resultado Esperado:

- Retornar status HTTP 401 (Unauthorized).
- Mensagem de erro indicando que o token expirou.

ID: US002-CT04

Título: Tentar Login com Senha Inválida
Pré-condição: O e-mail informado no payload deve estar cadastrado, mas a senha deve ser incorreta.
Passos:

1. Fazer uma requisição POST na rota `/login`.
2. Enviar no payload o `email` válido e uma senha inválida.

Resultado Esperado:

- Retornar status HTTP 401 (Unauthorized).
- Mensagem de erro indicando credenciais inválidas.

ID: US002-CT05

Título: Tentar Login Sem Preencher o Campo de E-mail
Pré-condição: Nenhuma.

Passos:

1. Fazer uma requisição POST na rota `/login`.
2. Enviar no payload omitindo as informações do campo `email`.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o campo `email` não pode ficar vazio.

ID: US002-CT06

Título: Tentar Login Sem Preencher o Campo de Senha
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/login`.
2. Enviar no payload omitindo as informações do campo `password`.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o campo `password` não pode ficar vazio.

Cenários de Testes para US 003 - [API] Produtos

ID: US003-CT01

Título: Criar Produto com Todos os Campos Válidos
Pré-condição: O nome do produto informado não deve estar previamente cadastrado.
Passos:

1. Fazer uma requisição POST na rota `/produtos`.
2. Enviar no payload os campos obrigatórios com valores válidos.

Resultado Esperado:

- Retornar status HTTP 201 (Created).
- Resposta deve conter mensagem informando o cadastro bem sucedido e id do produto.

ID: US003-CT02

Título: Tentar Criar Produto com Nome Duplicado
Pré-condição: Deve existir um produto com o mesmo nome já cadastrado.
Passos:

1. Fazer uma requisição POST na rota `/produtos`.
2. Enviar no payload um nome já utilizado em outro produto.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o nome do produto já está em uso.

ID: US003-CT03

Título: Criar Produto com Campos Obrigatórios Ausentes
Pré-condição: Nenhuma.
Passos:

1. Fazer uma requisição POST na rota `/produtos`.
2. Enviar no payload omitindo um ou mais campos obrigatórios.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que campos obrigatórios estão faltando.

ID: US003-CT04

Título: Atualizar Produto Existente com Dados Válidos
Pré-condição: Produto existente previamente cadastrado no sistema.
Passos:

1. Fazer uma requisição PUT na rota `/produtos/{_id}` usando um ID válido.
2. Enviar no payload campos obrigatórios preenchidos com valores válidos.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Produto atualizado com sucesso, refletido na listagem de produtos.

ID: US003-CT05

Título: Tentar Atualizar Produto com ID Inexistente
Pré-condição: O ID informado não deve corresponder a nenhum produto cadastrado.
Passos:

1. Fazer uma requisição PUT na rota `/produtos/{_id}` usando um ID inexistente.
2. Enviar no payload campos obrigatórios preenchidos com valores válidos.

Resultado Esperado:

- Retornar status HTTP 201 (Created).
- Um novo produto deve ser criado com os dados fornecidos.

ID: US003-CT06

Título: Tentar Atualizar Produto com Nome Duplicado
Pré-condição: Outro produto com o mesmo nome já deve existir no sistema.
Passos:

1. Fazer uma requisição PUT na rota `/produtos/{_id}` usando um ID válido.
2. Enviar no corpo da requisição um nome já cadastrado para outro produto.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando duplicidade do nome.

ID: US003-CT07

Título: Excluir Produto Existente que Não Está Associado a Carrinhos
Pré-condição: Produto previamente cadastrado e não vinculado a nenhum carrinho.
Passos:

1. Fazer uma requisição DELETE na rota `/produtos/{_id}` usando um ID válido.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Produto removido com sucesso, não aparecendo mais na listagem de produtos.

ID: US003-CT08

Título: Tentar Excluir Produto que Está Associado a Carrinhos
Pré-condição: Produto previamente cadastrado e vinculado a um carrinho.
Passos:

1. Fazer uma requisição DELETE na rota `/produtos/{_id}` usando um ID válido.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Mensagem de erro indicando que o produto está associado a um carrinho.

ID: US003-CT09

Título: Tentar Realizar Qualquer Ação na Rota de Produtos sem Autenticação
Pré-condição: Nenhuma autenticação realizada no sistema.
Passos:

1. Fazer requisições (GET, POST, PUT, DELETE) para a rota `/produtos` sem enviar token Bearer no cabeçalho.

Resultado Esperado:

- Retornar status HTTP 401 (Unauthorized).
- Mensagem de erro indicando falta de autenticação.

Cenários de Testes para US 004 - [API] Carrinhos

ID: US004-CT01

Título: Cadastrar um carrinho com sucesso.
Pré-condição:

- Existir produtos cadastrados no sistema com estoque disponível.
- O usuário autenticado não possui carrinho ativo.

Passos:

1. Realizar uma requisição POST para a rota `/carrinhos` com um payload contendo os IDs dos produtos e suas quantidades.
2. Ter um token válido no header `Authorization`.

Resultado Esperado:

- O sistema retorna o código de status 201 (Created).

- Retorna mensagem informando que o carrinho foi cadastrado e seu id.

ID: US004-CT02

Título: Tentar cadastrar carrinho com condições inválidas.
Pré-condição: O usuário autenticado não possui carrinhos cadastrado.

Passos:

1. Realizar uma requisição **POST** para a rota `/carrinhos` com um token válido no header Authorization.
2. Enviar no body da requisição um payload contendo condições inválidas, como produtos duplicados, produtos inexistentes, ou produtos sem quantidade suficiente.

Resultado Esperado:

- O sistema retorna o código de status 400 (Bad Request).
- A resposta deve conter uma mensagem apropriada.

ID: US004-CT03

Título: Tentar cadastrar um carrinho sem token.
Pré-condição: Nenhuma.
Passos:

1. Realizar uma requisição POST para a rota `/carrinhos` sem enviar o token no header `Authorization`.

Resultado Esperado:

- O sistema retorna o código de status 401 (Unauthorized).
- A resposta contém a mensagem "Token ausente, inválido ou expirado".

ID: US004-CT04

Título: Tentar buscar carrinho com ID inexistente.
Pré-condição: Nenhum carrinho existe com o ID fornecido.
Passos:

1. Realizar uma requisição GET para a rota `/carrinhos/{_id}` com um ID inexistente no endpoint.

Resultado Esperado:

- O sistema retorna o código de status 400 (Bad Request).
- A resposta contém a mensagem "Carrinho não encontrado".

ID: US004-CT05

Título: Tentar concluir compra sem carrinho ativo.
Pré-condição: O usuário autenticado não possui carrinho ativo.
Passos:

1. Realizar uma requisição DELETE para a rota `/carrinhos/concluir-compra` com um token válido no header `Authorization`.

Resultado Esperado:

- O sistema retorna o código de status 200 (OK).
- A resposta contém a mensagem "Não foi encontrado carrinho para esse usuário".

ID: US004-CT06

Título: Tentar cancelar compra sem token ou com token inválido.
Pré-condição: Nenhum token ou possuir um token inválido no header Authorization.

Passos:

1. Realizar uma requisição **DELETE** para a rota `/carrinhos/cancelar-compra` sem um token ou com um token inválido/expirado.

Resultado Esperado:

- O sistema retorna o código de status 401 (Unauthorized).
- A resposta contém a mensagem "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais".

---

8. Matriz de Risco:

Matriz de Risco para cada cenário de teste

Matriz de Risco para Cenários de Testes para US 001 - [API] Usuários:

ID	Título	Probabilidade	Impacto	Risco
US001-CT01	Criação de Usuário com Todos os Campos Válidos	Muito Alto	Muito Alto	Alto
US001-CT02	Tentar Criar Usuário com E-mail Duplicado	Moderado	Alto	Alto
US001-CT03	Criar Usuário com E-mail Inválido	Moderado	Alto	Alto
US001-CT04	Criar Usuário com E-mail de Provedor Restrito	Moderado	Alto	Alto
US001-CT05	Criar Usuário com Senha Fora dos Limites de Caracteres	Moderado	Alto	Alto
US001-CT06	Criar Usuário com Campo “nome” Vazio ou Ausente	Moderado	Alto	Alto
US001-CT07	Criar Usuário sem Campo ADMINISTRADOR	Muito Alto	Muito Alto	Alto
US001-CT08	Buscar Usuário por ID Inexistente	Moderado	Alto	Alto

US001-CT09	Atualizar Usuário Existente com Dados Válidos	Muito Alto	Muito Alto	Alto
US001-CT10	Atualizar Usuário com ID Inexistente	Moderado	Alto	Alto
US001-CT11	Tentar Atualizar Usuário com E-mail Duplicado	Moderado	Alto	Alto
US001-CT12	Atualizar Usuário sem Preencher Todos os Campos Obrigatórios	Moderado	Alto	Alto

Matriz de Risco para Cenários de Testes para US 002 - [API] Login:

ID	Título	Probabilidade	Impacto	Risco
US002-CT01	Login com Credenciais Válidas	Muito Alto	Muito Alto	Alto
US002-CT02	Tentar Login com E-mail Não Cadastrado	Moderado	Alto	Alto
US002-CT03	Testar Expiração do Token Após 10 Minutos	Moderado	Alto	Alto
US002-CT04	Tentar Login com Senha Inválida	Moderado	Alto	Alto
US002-CT05	Tentar Login Sem Preencher o Campo de E-mail	Baixo	Moderado	Médio
US002-CT06	Tentar Login Sem Preencher o Campo de Senha	Baixo	Moderado	Médio

Matriz de Risco para Cenários de Testes para US 003 - [API] Produtos:

ID	Título	Probabilidade	Impacto	Risco
US003-CT01	Criar Produto com Todos os Campos Válidos	Muito Alto	Muito Alto	Alto
US003-CT02	Tentar Criar Produto com Nome Duplicado	Baixo	Alto	Moderado
US003-CT03	Criar Produto com Campos Obrigatórios	Baixo	Alto	Moderado

	Ausentes			
US003-CT04	Atualizar Produto Existente com Dados Válidos	Muito Alto	Muito Alto	Alto
US003-CT05	Tentar Atualizar Produto com ID Inexistente	Moderado	Alto	Alto
US003-CT06	Tentar Atualizar Produto com Nome Duplicado	Baixo	Alto	Moderado
US003-CT07	Excluir Produto Existente que Não Está Associado a Carrinhos	Muito Alto	Muito Alto	Alto
US003-CT08	Tentar Excluir Produto que Está Associado a Carrinhos	Baixo	Alto	Moderado
US003-CT09	Tentar Realizar Qualquer Ação na Rota de Produtos sem Autenticação	Muito Alto	Muito Alto	Alto

Matriz de Risco para Cenários de Testes para US 004 - [API] Carrinhos:

ID do Cenário	Título do Cenário	Probabilidade	Impacto	Risco
US004-CT01	Cadastrar um carrinho com sucesso.	Baixo	Alto	Médio
US004-CT02	Tentar cadastrar carrinho com condições inválidas.	Moderado	Alto	Alto
US004-CT03	Tentar cadastrar um carrinho sem token.	Alto	Muito Alto	Alto
US004-CT04	Tentar buscar carrinho com ID inexistente.	Moderado	Alto	Médio
US004-CT05	Tentar concluir compra sem carrinho ativo.	Moderado	Moderado	Médio
US004-CT06	Tentar cancelar compra sem token ou com token inválido.	Alto	Muito Alto	Alto

9. Priorização da Execução dos Cenários de Teste:

Testes Priorizados

A priorização dos cenários de teste foi baseada nos seguintes critérios:

1. **Impacto no Negócio e Severidade da Falha**
   a. Cenários que comprometem os fluxos principais ou causam inconsistências graves no sistema são priorizados.
   b. Exemplos: Autenticação (login válido), validação de tokens, e regras de unicidade
2. **Frequência de Uso e Probabilidade de Falha**
   a. Funcionalidades mais utilizadas e sujeitas a entradas diversas recebem maior prioridade, dado o impacto potencial no usuário.
   b. **Exemplos**: CRUD de produtos e operações em carrinhos.
3. **Segurança e Conformidade**
   a. Cenários que envolvem riscos à segurança, privacidade, ou regras de negócio críticas, como autenticação ou validação de dados sensíveis.
   b. **Exemplos**: Uso indevido de tokens ou validações de e-mail.
4. **Experiência do Usuário e Cobertura de Regras de Negócio**
   a. Cenários que afetam diretamente a usabilidade e cumprem os critérios das User Stories são priorizados.
   b. **Exemplos**: Mensagens de erro claras e validações de cadastro.

A Priorização ficou da seguinte maneira:

Altíssima Prioridade: Cenários Críticos e Bloqueantes

Estes cenários são essenciais para o funcionamento básico e envolvem bloqueios ou falhas graves.

Usuários (US001):

- **CT01: Criar usuário com todos os campos válidos.**
- **CT04: Criar usuário com e-mail de provedor restrito.**
- **CT05: Criar usuário com senha fora dos limites de caracteres.**

Login (US002):

- **CT01: Login com credenciais válidas.**
- **CT03: Testar expiração do token após 10 minutos.**
- **CT04: Tentar login com senha inválida.**

Produtos (US003):

- **CT09: Tentar realizar qualquer ação na rota de produtos sem autenticação.**

Carrinhos (US004):

- **CT03: Tentar cadastrar um carrinho sem token.**

Alta Prioridade: Cenários de Validação e Fluxo Principal

Cenários importantes que validam fluxos principais, mas que não causam falhas catastróficas.

Usuários (US001):

- **CT02: Criar usuário com e-mail duplicado.**
- **CT03: Criar usuário com e-mail inválido.**

- **CT07: Criar usuário sem campo administrador.**
- **CT09: Atualizar usuário existente com dados válidos.**

Produtos (US003):

- **CT01: Criar produto com todos os campos válidos.**
- **CT02: Criar produto com nome duplicado.**
- **CT07: Excluir produto existente que não está associado a carrinhos.**
- **CT08: Tentar excluir produto que está associado a carrinhos.**

Carrinhos (US004):

- **CT05: Tentar concluir compra sem carrinho ativo.**

Média Prioridade: Cenários Moderadamente Relevantes

Estes cenários cobrem verificações úteis para validação de fluxos secundários e suportam a robustez da aplicação.

Usuários (US001):

- **CT06: Criar usuário com campo "nome" vazio ou ausente.**
- **CT10: Atualizar usuário com ID inexistente.**

Login (US002):

- **CT02: Tentar login com e-mail não cadastrado.**

Produtos (US003):

- **CT04: Atualizar produto existente com dados válidos.**
- **CT05: Tentar atualizar produto com ID inexistente.**

Carrinhos (US004):

- **CT01: Cadastrar um carrinho com sucesso.**
- **CT02: Tentar cadastrar carrinho com condições inválidas.**
- **CT06: Tentar cancelar compra sem token ou com token inválido.**

Baixa Prioridade: Cenários Complementares e Alternativos

Estes cenários cobrem validações secundárias e cenários de menor impacto ou baixa criticidade.

Usuários (US001):

- **CT08: Buscar usuário por ID inexistente.**
- **CT11: Tentar atualizar usuário com e-mail duplicado.**
- **CT12: Atualizar usuário sem preencher todos os campos obrigatórios.**

Login (US002):

- **CT05: Tentar login sem preencher o campo de e-mail.**
- **CT06: Tentar login sem preencher o campo de senha.**

Produtos (US003):

- **CT03: Criar produto com campos obrigatórios ausentes.**
- **CT06: Tentar atualizar produto com nome duplicado.**

Carrinhos (US004):

- **CT04: Tentar buscar carrinho com ID inexistente.**

---

10. Testes candidatos à automação:

Testes para automatizar

Usuários (US001):

- **CT01**: Criar usuário com todos os campos válidos.
  - Motivo: Fluxos essenciais como este são frequentemente utilizados como base para outros testes. Automatizá-los reduz retrabalho e garante eficiência em validações contínuas.
- **CT02**: Criar usuário com e-mail duplicado.
  - Motivo: Regras de negócio específicas, como evitar duplicidades, precisam ser consistentes. Automatizar ajuda a garantir que não ocorram falhas em diferentes ambientes
- **CT03**: Criar usuário com e-mail inválido
  - Motivo: Este cenário é essencial para validar a entrada de dados críticos. A automação permite testar rapidamente múltiplas combinações de formatos inválidos, garantindo que o sistema trate corretamente todas as entradas inadequadas e preserva a integridade dos dados do usuário.
- **CT06**: Criar usuário com campo "nome" vazio ou ausente.
  - Motivo: Entradas inválidas são uma categoria comum de erro. Automatizar a validação torna mais rápido identificar falhas nesses cenários.

Login (US002):

- **CT01**: Login com credenciais válidas.
  - Motivo: O login é pré-requisito para diversas operações no sistema. Automatizar economiza tempo ao evitar a repetição manual desse fluxo.
- **CT02**: Tentar login com e-mail não cadastrado.
  - Motivo: Automatizar cenários de erro frequentes, como esse, assegura que o sistema retorne respostas adequadas e consistentes.

Produtos (US003):

- **CT01**: Criar produto com todos os campos válidos.
  - Motivo: Cenários básicos e fundamentais como este são amplamente utilizados em ciclos de testes e automação reduz esforços manuais.
- **CT02**: Criar produto com nome duplicado.
  - Motivo: Validações de unicidade nos dados são fundamentais para evitar inconsistências. Automatizar facilita a detecção rápida de problemas nesse aspecto.
- **CT07**: Excluir produto existente que não está associado a carrinhos.
  - Motivo: Operações CRUD são comuns e repetitivas, tornando-as ideais para automação devido à sua previsibilidade.
- **CT08**: Tentar excluir produto que está associado a carrinhos
  - Motivo: Esse cenário protege as regras de negócio ao evitar que produtos em uso sejam excluídos, garantindo a consistência do sistema. A automação reduz a probabilidade de falhas em pontos críticos de integração entre carrinhos e produtos.
- **CT09**: Tentar realizar qualquer ação na rota de produtos sem autenticação.

- Motivo**:** Testar restrições de acesso é crucial para garantir a segurança da aplicação, e automatizar permite validação em larga escala.

Carrinhos (US004):

- **CT03**: Tentar cadastrar um carrinho sem token.
  - Motivo: Cenários que envolvem autenticação são críticos, pois protegem o sistema contra acessos não autorizados. A automação torna esse controle consistente.
- **CT05**: Tentar concluir compra sem carrinho ativo.
  - Motivo: Cenários inválidos são frequentes e exigem validações rigorosas. Automatizar acelera a identificação de comportamentos incorretos.
- **CT06**: Tentar cancelar compra sem token ou com token inválido
  - Motivo: Cenários que envolvem autenticação e segurança evitam acessos não autorizados e protegem as operações sensíveis. A automação desse caso ajuda a garantir que o sistema mantenha um controle rigoroso e eficiente sobre tokens de autenticação.

---

11. Cobertura de Testes:

Cobertura endpoints

Utilize um heatmap que marca endpoints testados com cores.

- Verde: Testado.
- Amarelo: Testado parcialmente ou em repetição.
- Vermelho: Não testado.

Endpoint	Prioridade	Cobertura
/login	Altíssima	🟢
/usuarios	Altíssima	🟢
/usuarios/{_id}	Alta	🟢
/produtos	Altíssima	🟢
/produtos/{_id}	Altíssima	🟢
/carrinhos	Altíssima	🟢
/carrinhos/{_id}	Baixa	🟢
/carrinhos/concluir-compra	Alta	🟢
/carrinhos/cancelar-compra	Média	🟢

Na API ServeRest temos 9 endpoints, que são os mesmos mostrados na coluna “Endpoint”

Para calcular a cobertura: quantidade de testes nos endpoints (9) / quantidade de endpoints na API (9) x 100.

Cobertura: 9/9 x 100 = 100%

Cobertura Cenários de Teste US 001 - [API] Usuários

Caso de Teste	Status
US001-CT01: Criação com Todos os Campos Válidos	✅ Passed
US001-CT02: Tentar Criar com E-mail Duplicado	✅ Passed
US001-CT03: Criar com E-mail Inválido	✅ Passed
US001-CT04: Criar com E-mail de Provedor Restrito	❌ Failed
US001-CT05: Criar com Senha Fora dos Limites	❌ Failed
US001-CT06: Criar com Campo “nome” Vazio	✅ Passed
US001-CT07: Criar sem Campo ADMINISTRADOR	✅ Passed
US001-CT08: Buscar por ID Inexistente	✅ Passed
US001-CT09: Atualizar com Dados Válidos	✅ Passed
US001-CT10: Atualizar com ID Inexistente	✅ Passed
US001-CT11: Atualizar com E-mail Duplicado	❌ Failed
US001-CT12: Atualizar sem Campos Obrigatórios	✅ Passed

Cobertura Cenários de Teste US 002 - [API] Login

Caso de Teste	Status
US002-CT01: Login com Credenciais Válidas	✅ Passed
US002-CT02: Tentar Login com E-mail Não Cadastrado	✅ Passed
US002-CT03: Testar Expiração do Token	✅ Passed
US002-CT04: Tentar Login com Senha Inválida	✅ Passed
US002-CT05: Tentar Login Sem Campo de E-mail	✅ Passed
US002-CT06: Tentar Login Sem Campo de Senha	✅ Passed

Cobertura Cenários de Teste US 003 - [API] Produtos

Caso de Teste	Status
US003-CT01: Criar com Todos os Campos Válidos	✅ Passed
US003-CT02: Tentar Criar com Nome Duplicado	✅ Passed
US003-CT03: Criar com Campos Obrigatórios Ausentes	✅ Passed
US003-CT04: Atualizar com Dados Válidos	✅ Passed
US003-CT05: Tentar Atualizar com ID Inexistente	✅ Passed
US003-CT06: Atualizar com Nome Duplicado	❌ Failed

US003-CT07: Excluir Produto Não Associado	✅ Passed
US003-CT08: Tentar Excluir Produto Associado	✅ Passed
US003-CT09: Ação sem Autenticação	❌ Failed

Cobertura Cenários de Teste US 004 - [API] Carrinhos

Caso de Teste	Status
US004-CT01: Cadastrar Carrinho com Sucesso	✅ Passed
US004-CT02: Tentar Cadastrar com Condições Inválidas	✅ Passed
US004-CT03: Tentar Cadastrar sem Token	✅ Passed
US004-CT04: Tentar Buscar com ID Inexistente	✅ Passed
US004-CT05: Tentar Concluir Compra Sem Ativo	✅ Passed
US004-CT06: Tentar Cancelar Sem Token Inválido	✅ Passed

Cobertura Geral dos Cenários de Teste manuais

Cenários de Teste	Testes Totais	Passed	Failed
US 001 - [API] Usuários	12	9 (75%)	3 (25%)
US 002 - [API] Login	6	6 (100%)	0 (0%)
US 003 - [API] Produtos	9	7 (77.78%)	2 (22.22%)
US 004 - [API] Carrinhos	6	6 (100%)	0 (0%)

Cobertura Cenários de Teste Automatizados

Cenários de Teste	**Status**
US001-CT01	✅ Passou
US001-CT02	✅ Passou
US001-CT03	✅ Passou
US001-CT06	✅ Passou
US002-CT01	✅ Passou
US002-CT02	✅ Passou
US003-CT01	✅ Passou
US003-CT02	✅ Passou
US003-CT07	✅ Passou
US003-CT08	✅ Passou

US003-CT09	❌ Falhou
US004-CT03	✅ Passou
US004-CT04	✅ Passou
US004-CT05	✅ Passou

Categoria dos Cenários de Teste	Percentual de Sucesso
US001 (Usuários)	**100%**
US002 (Login)	**100%**
US003 (Produtos)	**80%**
US004 (Carrinhos)	**100%**
Geral	**92.9%**

---

12. Interpretação dos Resultados de Testes Manuais e Automatizados:

Contexto e Ferramentas Utilizadas:

Os testes foram conduzidos utilizando duas abordagens complementares: manuais e automatizados.

- **Testes Manuais**: Executados no Postman, onde foram avaliados 33 casos de teste, organizados em ciclos no QAlity Plus (Jira).
- **Testes Automatizados**: Desenvolvidos e executados no VSCode com **Robot Framework**, focando em validar cenários-chave selecionados.

Os resultados foram registrados no QAlity, consolidando as métricas e os insights para análise.

Panorama Geral dos Resultados:

Dos 33 casos de teste avaliados manualmente, 28 passaram, enquanto 5 falharam, resultando em uma taxa de sucesso de 84.84%.

Nos testes automatizados, 14 casos foram avaliados, com 13 aprovados e 1 falha, alcançando uma taxa de sucesso de **93.33%**.

Abordagem	Testes Totais	Passados	Falhados	Taxa de Sucesso
Testes Manuais	33	28	5	84.84%
Testes Automatizados	14	13	1	92.86%

Destaques Positivos:

- **Rota de Login (US 002)**: 100% de sucesso em ambas as abordagens. A validação do token, autenticação e controle de credenciais foram satisfatórias.

- **Rota de Carrinhos (US 004)**: Também alcançou 100% de sucesso, indicando que as operações de gerenciamento e validação estão robustas.
- **Rota de Produtos (US 003)**: A maioria dos casos foi concluída com sucesso, destacando a funcionalidade básica.

Pontos de Melhoria Identificados:

- **Testes Manuais**:
  - Falhas na rota de usuários (US 001) relacionadas a validações de entrada, como provedores de e-mail restritos, senhas fora do limite padrão e duplicidade de e-mails ao atualizar o usuário, indicam necessidade de ajuste nos requisitos ou na implementação.
  - Na rota de produtos (US 003), a falha na validação de nomes duplicados ao atualizar o produto pode impactar a consistência de dados.
- **Testes Automatizados**:
  - A única falha registrada ocorreu em **US003-CT09 (Ação sem Autenticação)**, indicando um possível problema de segurança ao permitir acesso não autorizado a certas operações.

Links:

- **Test Cycles com os resultados dos Test Cases manuais e automatizados relatados no QAlity:**
  - https://raiquepereira04.atlassian.net/plugins/servlet/ac/com.soldevelo.apps.test_management_premium/test-cycles?board.id=70&project.key=TCTASR&project.id=10069&board.type=simple
- **Bugs Relatados:**
  - Issues e Melhorias | Quadro