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. Apresentação:

Este plano de testes foi criado para garantir que a API ServeRest funcione como esperado e siga as regras de negócio definidas. Ele cobre as etapas de planejamento, análise, execução e documentação dos testes, com cenários baseados nas User Stories e na documentação técnica do Swagger.

---

3. Objetivo do Plano:

O objetivo deste plano é garantir que as funcionalidades da API ServeRest atendam aos critérios definidos nas User Stories e no Swagger. Queremos identificar problemas, garantir a segurança e usabilidade das funcionalidades, além de priorizar cenários que podem ser automatizados para facilitar futuros testes.

---

4. Escopo dos Testes:

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

---

5. 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 JWT.
  - 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.

---

6. 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 Exploratórios para identificar comportamentos inesperados.
- Testes de Fluxo Completo para validar a integração entre funcionalidades.
- Verificação de Conformidade com o Padrão REST.

---

7. Mapa Mental da aplicação:

---

8. Cenários de Testes Planejados

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 201 (Created).
- Valor padrão para o campo `administrador` deve ser configurado (se existir).

ID: US001-CT08

Título: Listar Todos os Usuários
Pré-condição: Devem existir usuários cadastrados no sistema.
Passos:

1. Fazer uma requisição GET na rota `/usuarios`.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Resposta deve conter uma lista com todos os usuários cadastrados.

ID: US001-CT09

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 404 (Not Found).
- Mensagem de erro indicando que o usuário não foi encontrado.

ID: US001-CT10

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).
- Os dados do usuário devem ser atualizados conforme enviados no payload.

ID: US001-CT11

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-CT12

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-CT13

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.

ID: US001-CT14

Título: Excluir Usuário Existente
Pré-condição: Deve existir um usuário previamente cadastrado.
Passos:

1. Fazer uma requisição DELETE na rota `/usuarios/{_id}` usando o ID de um usuário existente.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- O usuário deve ser removido do sistema.

ID: US001-CT15

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

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

Resultado Esperado:

- Retornar status HTTP 404 (Not Found) ou mensagem de erro apropriada.

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 informações do usuário autenticado.

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 o campo `email`.

Resultado Esperado:

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

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 o campo `password`.

Resultado Esperado:

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

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 os dados do produto criado.

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: Listar Todos os Produtos Cadastrados
Pré-condição: Devem existir produtos cadastrados no sistema.
Passos:

1. Fazer uma requisição GET na rota `/produtos`.

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Resposta deve conter a lista completa dos produtos cadastrados.

ID: US003-CT05

Título: Buscar Produto por ID Válido
Pré-condição: Deve existir um produto com o ID informado.
Passos:

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

Resultado Esperado:

- Retornar status HTTP 200 (OK).
- Resposta deve conter os dados do produto correspondente ao ID.

ID: US003-CT06

Título: Buscar Produto por ID Inexistente
Pré-condição: O ID informado não deve corresponder a nenhum produto cadastrado.
Passos:

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

Resultado Esperado:

- Retornar status HTTP 404 (Not Found).
- Mensagem de erro indicando que o produto não foi encontrado.

ID: US003-CT07

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-CT08

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-CT09

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-CT10

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-CT11

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-CT12

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

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

Resultado Esperado:

- Retornar status HTTP 404 (Not Found).
- Mensagem de erro indicando que o produto não foi encontrado.

ID: US003-CT13

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: Listar carrinhos cadastrados com sucesso.
Pré-condição: Existir pelo menos um carrinho vinculado ao usuário autenticado.
Passos:

1. Realizar uma requisição GET para a rota `/carrinhos` com um token válido no header `Authorization`.

Resultado Esperado:

- O sistema retorna o código de status 200 (OK).
- A resposta contém uma lista de carrinhos vinculados ao usuário autenticado.

ID: US004-CT02

Título: Listar carrinhos quando nenhum carrinho está vinculado ao usuário.
Pré-condição: O usuário autenticado não possui carrinhos cadastrados.
Passos:

1. Realizar uma requisição GET para a rota `/carrinhos` com um token válido no header `Authorization`.

Resultado Esperado:

- O sistema retorna o código de status 200 (OK).
- A resposta contém uma lista vazia.

ID: US004-CT03

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).
- O carrinho é cadastrado e vinculado ao usuário autenticado.

ID: US004-CT04

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-CT05

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-CT06

Título: Buscar carrinho por ID com sucesso.
Pré-condição:

- Um carrinho deve estar cadastrado no sistema com o ID fornecido.
  Passos:

1. Realizar uma requisição GET para a rota `/carrinhos/{_id}` com o ID válido do carrinho no endpoint.

Resultado Esperado:

- O sistema retorna o código de status 200 (OK).
- A resposta contém os detalhes do carrinho correspondente ao ID.

ID: US004-CT07

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-CT08

Título: Concluir compra com sucesso.
Pré-condição:

- O usuário autenticado possui um 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).
- O carrinho é excluído do sistema.

ID: US004-CT09

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-CT10

Título: Cancelar compra com sucesso.
Pré-condição:

- O usuário autenticado possui um carrinho ativo.

Passos:

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

Resultado Esperado:

- O sistema retorna o código de status 200 (OK).
- O carrinho é excluído e o estoque dos produtos é reabastecido.

ID: US004-CT11

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

1. Realizar uma requisição DELETE para a rota `/carrinhos/cancelar-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-CT12

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 ausente, inválido ou expirado".

Cenários de Testes Gerais

ID: GEN-CT01
Título: Testar Todos os Endpoints Disponíveis no Swagger com Métodos HTTP Válidos
Pré-condição: Os endpoints documentados no Swagger devem estar implementados e acessíveis para testes.
Passos:

1. Consultar a documentação Swagger para identificar os endpoints e métodos válidos.
2. Configurar as chamadas no Postman utilizando os métodos HTTP válidos (GET, POST, PUT, DELETE, etc.) para cada endpoint.
3. Realizar as requisições e validar as respostas e os dados retornados.

Resultado Esperado:

- Retornar o status HTTP esperado para cada método/teste.
- Os dados retornados devem estar em conformidade com a documentação Swagger.

ID: GEN-CT02
Título: Realizar Chamadas Inválidas com Métodos HTTP Inadequados
Pré-condição: Endpoints documentados no Swagger estão disponíveis para testes.
Passos:

1. Identificar os métodos HTTP inadequados para cada endpoint (ex.: DELETE em uma rota de listagem).
2. Configurar e executar chamadas no Postman utilizando os métodos HTTP inadequados.

Resultado Esperado:

- Retornar status HTTP 405 (Method Not Allowed).
- Exibir uma mensagem de erro clara sobre a restrição do método utilizado.

ID: GEN-CT03
Título: Enviar Payload Vazio ou com Formato Inválido em Qualquer Endpoint
Pré-condição: Os endpoints documentados no Swagger estão disponíveis e requerem payloads válidos.
Passos:

1. Identificar os endpoints que aceitam payloads.
2. Configurar chamadas no Postman com payload vazio ou com dados no formato inválido.
3. Executar as requisições e observar os resultados.

Resultado Esperado:

- Retornar status HTTP 400 (Bad Request).
- Exibir uma mensagem de erro clara descrevendo o problema no payload enviado.

---

9. Matriz de Risco

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	Listar Todos os Usuários	Muito Alto	Muito Alto	Alto
US001-CT09	Buscar Usuário por ID Inexistente	Moderado	Alto	Alto
US001-CT10	Atualizar Usuário Existente com Dados Válidos	Muito Alto	Muito Alto	Alto
US001-CT11	Atualizar Usuário com ID Inexistente	Moderado	Alto	Alto
US001-CT12	Tentar Atualizar Usuário com E-mail Duplicado	Moderado	Alto	Alto
US001-CT13	Atualizar Usuário sem Preencher Todos os Campos Obrigatórios	Moderado	Alto	Alto
US001-CT14	Excluir Usuário Existente	Muito Alto	Muito Alto	Alto
US001-CT15	Tentar Excluir Usuário Inexistente	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 Ausentes	Baixo	Alto	Moderado
US003-CT04	Listar Todos os Produtos Cadastrados	Muito Alto	Muito Alto	Alto
US003-CT05	Buscar Produto por ID Válido	Muito Alto	Muito Alto	Alto
US003-CT06	Buscar Produto por ID Inexistente	Baixo	Alto	Moderado
US003-CT07	Atualizar Produto Existente com Dados Válidos	Muito Alto	Muito Alto	Alto
US003-CT08	Tentar Atualizar Produto com ID Inexistente	Moderado	Alto	Alto
US003-CT09	Tentar Atualizar Produto com Nome Duplicado	Baixo	Alto	Moderado
US003-CT10	Excluir Produto Existente que Não Está Associado a Carrinhos	Muito Alto	Muito Alto	Alto
US003-CT11	Tentar Excluir Produto que Está Associado a Carrinhos	Baixo	Alto	Moderado
US003-CT12	Tentar Excluir Produto Inexistente	Baixo	Alto	Moderado
US003-CT13	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	Listar carrinhos cadastrados com sucesso.	Baixo	Baixo	Baixo
US004-CT02	Listar carrinhos quando nenhum carrinho está vinculado ao usuário.	Baixo	Muito baixo	Baixo
US004-CT03	Cadastrar um carrinho com sucesso.	Baixo	Alto	Médio
US004-CT04	Tentar cadastrar carrinho com condições inválidas.	Moderado	Alto	Alto
US004-CT05	Tentar cadastrar um carrinho sem token.	Alto	Muito Alto	Alto
US004-CT06	Buscar carrinho por ID com sucesso.	Baixo	Moderado	Baixo
US004-CT07	Tentar buscar carrinho com ID inexistente.	Moderado	Alto	Médio
US004-CT08	Concluir compra com sucesso.	Baixo	Alto	Médio
US004-CT09	Tentar concluir compra sem carrinho ativo.	Moderado	Moderado	Médio
US004-CT10	Cancelar compra com sucesso.	Baixo	Alto	Médio
US004-CT11	Tentar cancelar compra sem carrinho ativo.	Moderado	Moderado	Médio
US004-CT12	Tentar cancelar compra sem token ou com token inválido.	Alto	Muito Alto	Alto

Matriz de Risco para Cenários de Testes Gerais:

ID	Título	Probabilidade	Impacto	Risco
GEN-CT01	Testar Todos os Endpoints Disponíveis no Swagger com Métodos HTTP Válidos	Muito Alto	Muito Alto	Alto

GEN-CT02	Realizar Chamadas Inválidas com Métodos HTTP Inadequados	Alto	Médio	Moderado
GEN-CT03	Enviar Payload Vazio ou com Formato Inválido em Qualquer Endpoint	Alto	Alto	Alto

---

10. Priorização da Execução dos Cenários de Teste

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

1. Impacto no Negócio:

- Altíssima Prioridade: Cenários que, se falharem, comprometem diretamente os principais fluxos do negócio ou tornam a aplicação inutilizável. Exemplo: autenticação (login válido) e validação de token.
- Alta Prioridade: Cenários que garantem a conformidade com as regras de negócio principais, mas cuja falha não bloqueia totalmente o uso da aplicação.

2. Frequência de Uso:

- Cenários relacionados a funcionalidades frequentemente utilizadas pelos usuários, como autenticação, manipulação de produtos e carrinhos, receberam maior prioridade.

3. Severidade da Falha:

- Cenários cujas falhas podem gerar inconsistências graves no sistema ou quebrar integrações, como transações inválidas ou duplicadas, foram priorizados em níveis altos.
  Exemplo: evitar duplicidade de produtos e exclusão de produtos associados a carrinhos.

4. Riscos Associados:

- Cenários que representam riscos significativos à segurança, privacidade, ou conformidade com normas, como o uso indevido de tokens ou validações inadequadas de e-mails e senhas.
  Exemplo: segurança da autenticação via token Bearer.

5. Cobertura das Regras de Negócio:

- Cenários que validam diretamente os critérios de aceitação definidos nas User Stories foram priorizados.
  Exemplo: validação de senhas no cadastro de usuários (mínimo de 5 e máximo de 10 caracteres).

6. Complexidade e Dependências:

- Cenários com alto número de dependências de outras APIs ou funcionalidades foram ajustados em sua prioridade para equilibrar os testes.
  Exemplo: CRUD de produtos depende de autenticação e cadastro de usuários.

7. Probabilidade de Falha:

- Funcionalidades mais complexas ou sujeitas a entradas diversas foram priorizadas por terem maior probabilidade de apresentar erros.
  Exemplo: tentativas de login com e-mails ou senhas inválidas.

8. Experiência do Usuário (UX):

- Cenários que impactam diretamente a experiência do usuário final, como mensagens de erro claras ou fluxos bem definidos, receberam maior prioridade para garantir usabilidade.
  Exemplo: impedir cadastro de usuários com e-mails inválidos ou de provedores restritos.

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.**
- **CT04: Tentar login com senha inválida.**

Produtos (US003):

- **CT13: Tentar realizar qualquer ação na rota de produtos sem autenticação.**

Carrinhos (US004):

- **CT05: 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.**
- **CT10: Atualizar usuário existente com dados válidos.**

Login (US002):

- **CT03: Testar expiração do token após 10 minutos.**

Produtos (US003):

- **CT02: Criar produto com nome duplicado.**
- **CT10: Excluir produto existente que não está associado a carrinhos.**
- **CT11: Tentar excluir produto que está associado a carrinhos.**

Carrinhos (US004):

- **CT09: Tentar concluir compra sem carrinho ativo.**

Média Prioridade: Cenários Complementares e Alternativos

Estes cenários cobrem validações secundárias e cenários alternativos.

Usuários (US001):

- CT06: Criar usuário com campo "nome" vazio ou ausente.
- CT09: Buscar usuário por ID inexistente.
- CT11: Atualizar usuário com ID inexistente.
- CT12: Tentar atualizar usuário com e-mail duplicado.
- CT13: Atualizar usuário sem preencher todos os campos obrigatórios.

Login (US002):

- CT02: Tentar login com e-mail não cadastrado.
- CT05: Tentar login sem preencher o campo de e-mail.
- CT06: Tentar login sem preencher o campo de senha.

Produtos (US003):

- CT01: Criar produto com todos os campos válidos.
- CT03: Criar produto com campos obrigatórios ausentes.
- CT07: Atualizar produto existente com dados válidos.
- CT08: Tentar atualizar produto com ID inexistente.
- CT09: Tentar atualizar produto com nome duplicado.

Carrinhos (US004):

- CT03: Cadastrar um carrinho com sucesso.
- CT04: Tentar cadastrar carrinho com condições inválidas.
- CT06: Buscar carrinho por ID com sucesso.
- CT07: Tentar buscar carrinho com ID inexistente.
- CT12: Tentar cancelar compra sem token ou com token inválido.

Cenários Gerais (GEN):

- CT01: Cobertura geral de endpoints.
- CT02: Verificar segurança na API (como autenticação e autorização).
- CT03: Testar estabilidade e desempenho básico da aplicação.

Baixa Prioridade: Cenários de Menor Impacto

Cobrem funcionalidades não críticas ou verificações exploratórias.

Usuários (US001):

- CT08: Listar todos os usuários.
- CT14: Excluir usuário existente.
- CT15: Tentar excluir usuário inexistente.

Produtos (US003):

- CT04: Listar todos os produtos cadastrados.
- CT05: Buscar produto por ID válido.
- CT06: Buscar produto por ID inexistente.
- CT12: Tentar excluir produto inexistente.

Carrinhos (US004):

- **CT01: Listar carrinhos cadastrados com sucesso.**
- **CT02: Listar carrinhos quando nenhum carrinho está vinculado ao usuário.**
- **CT08: Concluir compra com sucesso.**
- **CT10: Cancelar compra com sucesso.**
- **CT11: Tentar cancelar compra sem carrinho ativo.**

---

11. Testes candidatos a automação

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
- **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.
- **CT08**: Listar todos os usuários.
  - Motivo: Listagens estáveis e básicas são ideais para automação, garantindo que falhas não ocorram em funcionalidades amplamente utilizadas.

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.
- **CT04**: Listar todos os produtos cadastrados.
  - Motivo: Garantir que dados sejam recuperados corretamente em listagens é essencial e fácil de testar automaticamente devido à previsibilidade do fluxo.
- **CT07**: Atualizar produto existente com dados válidos.
  - Motivo: A estabilidade de operações CRUD é essencial para o sistema. Automatizar permite validações constantes e rápidas.
- **CT10**: 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.

- **CT13**: 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**: Cadastrar um carrinho com sucesso.
  - Motivo: Testar cenários principais repetidos em diversos fluxos aumenta a confiabilidade do sistema, e a automação garante consistência.
- **CT05**: 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.
- **CT08**: Concluir compra com sucesso.
  - Motivo: Fluxos principais, como finalizar uma compra, são críticos. Automatizá-los assegura que o sistema funcione corretamente mesmo após atualizações.
- **CT09**: Tentar concluir compra sem carrinho ativo.
  - Motivo: Cenários inválidos são frequentes e exigem validações rigorosas. Automatizar acelera a identificação de comportamentos incorretos.

Cenários Gerais (GEN):

- **CT02**: Verificar segurança na API (autenticação e autorização).
  - Motivo: A segurança precisa ser garantida continuamente. Automatizar esses testes possibilita monitoramento constante e eficiente contra vulnerabilidades.
- **CT03**: Testar estabilidade e desempenho básico da aplicação.
  - Motivo: Testes de carga ou estresse são inviáveis manualmente devido à complexidade. Automatizá-los permite validações mais completas e rápidas.

---

12. Cobertura de Testes:

Na API ServeRest temos 9 endpoints, sendo eles:

- Login
  - /login
- Usuários:
  - /usuarios
  - /usuarios/{_id}
- Produtos:
  - /produtos
  - /produtos/{_id}
- Carrinhos:
  - /carrinhos
  - /carrinhos/{_id}
  - /carrinhos/concluir-compra
  - /carrinhos/cancelar-compra
- Total = 9 endpoints

No Postman foi realizado pelo menos uma request em 7 endpoints diferentes, sendo eles:

- Altíssima Prioridade:
  - /usuarios
  - /login
  - /produtos
  - /produtos/{_id}
  - /carrinhos
- Alta Prioridade:
  - /usuarios/{_id}
  - /carrinhos/concluir-compra
    - Endpoints repetidos:
      - /usuario
      - /usuario/{_id}
      - /login
      - /produtos
      - /produtos/{_id}
- Média Prioridade eu repeti testes nos seguintes endpoints:
  - /usuarios
  - /usuarios/{_id}
- Então foi testado 7 endpoints diferentes.

Para calcular a cobertura: quantidade de testes nos endpoints (7) / quantidade de endpoints na API (9) x 100.

Cobertura: 7/9 x 100 = 77%