

Challenge - Plano de Testes - ServeRest

1. Identificação do Plano de Testes [🔗](#)

- Nome do projeto: ServeRest
- Responsável: Cassia Yumi Iwamoto Basso
- Data de criação: 05/05/2025

2. Apresentação [🔗](#)

Este documento apresenta o planejamento de testes da API ServeRest, que simula um e-commerce, com funcionalidades como cadastro de usuários, login, gerenciamento de produtos e carrinho de compras.

3. Objetivo do Plano [🔗](#)

Descrever a abordagem e o escopo dos testes a serem realizados na [API ServeRest](#) para garantir sua qualidade funcional e não funcional.

4. Escopo dos Testes [🔗](#)

- ✓ Endpoints: login, usuarios, produtos e carrinhos. [🔗](#)
- ✓ Validação de regras de negócio. [🔗](#)
- ✓ Verificação de status HTTP. [🔗](#)
- ✗ Fora do escopo: Testes de segurança avançados, Teste de carga.

5. Análise [🔗](#)

A API da ServeRest possui quatro grupos principais: Usuários, Login, Produtos e Carrinhos. Serão testados:

- Grupo Usuários: testes de criação, listagem, edição, exclusão, busca por ID e validação de regras
- Grupo Login: autenticação e geração de token JWT
- Grupo Produtos: Testes de CRUD, busca por ID e validação de regras.
- Grupo Carrinhos: Testes de criação, busca, exclusão e verificação de retorno de produtos ao estoque.

Serão aplicados os seguintes testes:

- Testes funcionais (ex: cadastro de usuário, login, criar carrinho)
- Testes de validação de regras de negócio
- Testes de fluxo completo
- Testes negativos
- Testes de segurança/autorização

Em relação à autenticação e controle e acesso serão validados o perfil admin para rotas protegidas, verificação de restrições de acesso sem token e testes com token expirado ou ausente.

Será realizado a análise da cobertura com base no Path Coverage, utilizando-se o seguinte critério:

- Total de URIs disponíveis: X
- URIs testadas: Y
- **Cobertura:** Path Coverage= $(Y / X) * 100\%$

Será realizada a análise dos riscos, tais como os riscos associados a falhas em login, produtos ou carrinho e priorização utilizando-se uma matriz de risco.

Serão analisados os testes candidatos à automação.

Os dados de testes utilizados poderão ser usados da seguinte forma:

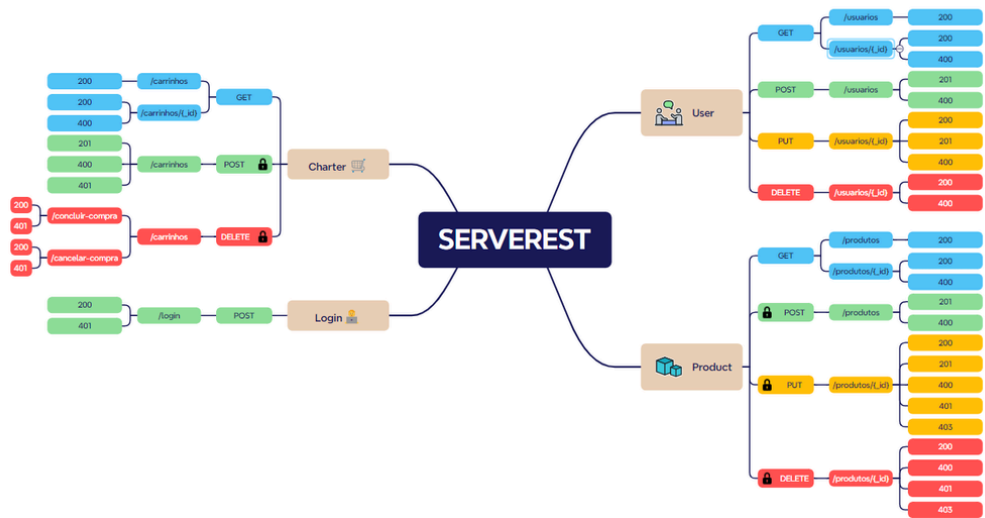
- Usuários com diferentes permissões
- Produtos com diferentes preços e quantidades
- Tokens válidos e inválidos
- Carrinhos com múltiplos produtos

6. Técnicas aplicadas


Com base na documentação da API e nas users stories disponibilizadas, foram selecionadas as seguintes técnicas de teste:

- Partição de Equivalência: testes de entradas válidas e inválidas;
- Análise de Valor Limite: testes visando os limites máximos e mínimos para entrada;
- Teste de Valores Nulos e Vazios
- Testes baseados em regras de negócio;
- Testes exploratórios: visando encontrar comportamento não previstos nos requisitos ou na documentação;
- Técnica de transição de estados: especialmente quanto ao token e autenticação.

7. Mapa Mental da aplicação



8. Cenários de Testes Planejados

Com base nas users stories apresentadas e na documentação da API  foram planejados os seguintes cenários de testes:

1 US 001 - Usuários

Cenário de Teste 01
Descrição: Criar um novo usuário com dados válidos
Pré-condição: Nenhuma.
Condições:

- Os vendedores (usuários) deverão possuir os campos NOME, E-MAIL, PASSWORD e ADMINISTRADOR;
- Não deverá ser possível cadastrar usuários com e-mails de provedor gmail e hotmail;
- Os e-mails devem seguir um padrão válido de e-mail para o cadastro;
- As senhas devem possuir no mínimo 5 caracteres e no máximo 10 caracteres;
- Não deverá ser possível fazer ações e chamadas para usuários inexistentes;
- Não deve ser possível criar um usuário com e-mail já utilizado.

Passo a passo:

1. Enviar uma requisição POST para /usuarios.
2. No corpo da requisição, incluir dados, conforme entradas abaixo.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado Esperado
CT001	Criar usuário válido	{ "nome": "Maria Teste", "email": " maria@teste.com ", "password": "123456", "administrador": "true" }	Response 201 Created Mensagem: "Cadastro realizado com sucesso"
CT002	Criar usuário com e-mail já cadastrado	{ "nome": "João Teste", "email": " maria@teste.com ", "password": "123456", "administrador": "true" }	Response 400 Bad Request Mensagem: "Este email já está sendo usado"
CT003	Criar usuário com e-mail de domínio gmail.com	{ "nome": "Maria Teste", "email": " teste@gmail.com ", "password": "123456", "administrador": "true" }	Response 400 Bad Request Mensagem: "Não é possível realizar o cadastro com gmail".
CT004	Criar usuário com e-mail de domínio hotmail.com	{ "nome": "Maria Teste", "email": " teste@hotmail.com ", "password": "123456", "administrador": "true" }	Mesmo comportamento do CT003
CT005	Criar usuário com e-mail em formato inválido	{ "nome": "Maria Teste", "email": " teste.teste.com ", "password": "123456", "administrador": "true" }	Response 400 Bad Request Mensagem: "Formato de e-mail inválido"
CT006	Criar usuário com senha menor que 5 caracteres	{ "nome": "Maria Teste", "email": " teste12@qa.com ", "password": "t", }	Response 400 Bad Request Mensagem: "Senha deve possuir de 5 a 10"

		"administrador": "true" }	caracteres"
CT007	Criar usuário com senha maior que 10 caracteres	{ "nome": "Maria Teste", "email": " teste12@qa.com ", "password": "0123456789101112", "administrador": "true" }	Retorno 400 Bad Request Mensagem: "Senha deve possuir de 5 a 10 caracteres"
CT008	Criar usuário com admin "false"	{ "nome": "Maria Teste", "email": " teste12@qa.com ", "password": "0123456789101112", "administrador": "false" }	Response 201 Created Mensagem: "Cadastro realizado com sucesso"
CT009	Enviar payload vazio	{}	Retorno 400 Bad Request Mensagens de erro explicativas
CT010	Verificar se os campos obrigatórios (nome, email, senha, admin) são validados corretamente	{ "email": " teste12@qa.com ", "password": "0123456", "administrador": "false" }	Response 400 Bad Request Mensagens de erro específicas por campo
CT011	Validar campos extras ou desnecessários no payload	{ "nome": "Maria Teste", "email": " teste12@qa.com ", "password": "0123456789101112", "administrador": "false", "adicional": "adicional" }	Reponse 400 Bad Request Mensagens de erro específica do campo desnecessário

Cenário de Teste 02

Descrição: Atualizar usuário com dados válidos

Pré-condição: Usuário deve ter sido criado anteriormente.

Condições:

- Não deve ser possível cadastrar usuário com e-mail já utilizado utilizando PUT;
- Caso não seja encontrado usuário com o ID informado no PUT, um novo usuário deverá ser criado

Passo a passo:

1. Enviar uma requisição PUT para /usuarios.
2. No corpo da requisição, incluir dados, conforme entradas abaixo.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado Esperado
----	-----------	----------------	--------------------

CT012	Atualizar usuário existente com novos dados válidos	ID válido	Response 200 OK
CT013	Atualizar usuário com ID inexistente	ID inexistente	Response 201 Created Mensagem: Cadastro realizado com sucesso
CT014	Atualizar usuário com e-mail já existente (mesmo de outro usuário)	E-mail duplicado	Response 400 Bad Request Mensagem de erro “Este email já está sendo usado”

Cenário de Teste 03

Descrição: Listar usuários

Pré-condição: Nenhuma.

Condições:

- Não deverá ser possível fazer ações e chamadas para usuários inexistentes.

Passo a passo:

1. Enviar uma requisição GET para /usuarios.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
CT015	Listar todos os usuários	GET /usuarios	Response 200 OK Lista de usuários existentes ou vazio
CT016	Listar usuário com ID válido	GET /usuarios/:id	Response 200 OK Lista de usuários existentes ou vazio
CT017	Listar usuário com ID inválido	GET /usuarios/:id	Response 400 Bad Request Lista de usuários existentes ou vazio

Cenário de Teste 04

Descrição: Excluir usuário

Pré-condição: Usuário cadastrado e validado.

Condições:

- Não deverá ser possível fazer ações e chamadas para usuários inexistentes.

Passo a passo:

1. Enviar uma requisição DELETE para /usuarios.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
----	-----------	----------------	--------------------

CT018	Deletar usuário existente	DELETE /usuarios/:id	Response 200 OK Mensagem de sucesso
CT019	Deletar usuário com ID inválido ou inexistente	DELETE /usuarios/:id	Response 400 Bad Request Mensagem “Usuário não encontrado”

2 US 002 - Login [🔗](#)

Cenário de Teste 04

Descrição: Realizar autenticação com sucesso.

Pré-condição: Usuário cadastrado e validado.

Condições:

- Usuários não cadastrados não deverão conseguir autenticar;
- Usuários com senha inválida não deverão conseguir autenticar;
- No caso de não autenticação, deverá ser retornado um status code 401 (Unauthorized);
- Usuários existentes e com a senha correta deverão ser autenticados;
- A autenticação deverá gerar um token Bearer;
- A duração da validade do token deverá ser de 10 minutos.

Passo a passo:

1. Enviar uma requisição POST para /login.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado Esperado
CT020	Login com credenciais válidas	{ "email": "fulano@qa.com", "password": "teste" }	Response 200 OK Token Bearer gerado na resposta
CT021	Login com e-mail não cadastrado	{ "email": "inexiste@qa.com", "password": "teste" }	Response 401 Unauthorized Mensagem: “E-mail e/ou senha inválidos”
CT022	Login com senha incorreta	{ "email": "fulano@qa.com", "password": "senha" }	Response 401 Unauthorized Mensagem: “E-mail e/ou senha inválidos”
CT023	Login com e-mail em formato inválido	{ "email": "fulano.qa.com", "password": "teste" }	Response 400 Bad Request Mensagem: “E-mail deve ser um e-mail válido”

CT024	Login com payload incompleto	{ "password": "teste" }	Retorno 400 Bad Request Mensagem de erro específica
CT025	Login com payload vazio	{}	Response 400 Bad Request Mensagem de erro específica
CT026	Verificar se token expira após 10 minutos	Realizar login Aguardar mais de 10 minutos. Acessar rota protegida com token. Ex: POST/Produtos	Response 401 Unauthorized Mensagem: Token de acesso ausente, inválido, expirado ou usuário do token não existe mais.
CT027	Verificar se mensagens de erro não expõem dados sensíveis	Ex: senha incorreta	Mensagem genérica: “Usuário e/ou senha inválidos”, sem especificar se o usuário existe

3 US 003 - Produtos [🔗](#)

Cenário de Teste 05

Descrição: Cadastrar produtos com sucesso.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.

Condições:

- Usuários não autenticados não devem conseguir realizar ações na rota de Produtos;
- Não deve ser possível realizar o cadastro de produtos com nomes já utilizados.

Passo a passo:

1. Enviar uma requisição POST para /produtos.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado Esperado
CT028	Cadastrar produto com usuário autenticado	{ "nome": "Logitech ", "preco": 520, "descricao": "Mouse", "quantidade": 3 }	Response 201 Created Mensagem: “Cadastro realizado com sucesso”
CT029	Cadastrar produto com nome já existente	{ "nome": "Logitech ", "preco": 520, "descricao": "Mouse", "quantidade": 3 }	Retorno 400 Bad Request Mensagem: “Já existe produto com esse nome”

CT030	Cadastrar produto sem usuário estar autenticado	Sem token <pre>{ "nome": "Logitech ", "preco": 520, "descricao": "Mouse", "quantidade": 3 }</pre>	Retorno 401 Unauthorized Mensagem: “Token de acesso ausente, inválido, expirado ou usuário do token não existe mais”.
CT031	Cadastrar produto com campos obrigatórios ausentes	<pre>{ "nome": "Logitech X", "descricao": "Mouse", "quantidade": 3 }</pre>	Retorno 400 Bad Request Mensagem de erro de validação do campo ausente.

Cenário de Teste 06

Descrição: Atualizar dados do produtos com sucesso.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.

Condições:

- Usuários não autenticados não devem conseguir realizar ações na rota de Produtos;
- Caso não exista produto com o o ID informado na hora do UPDATE, um novo produto deverá ser criado;
- Produtos criados através do PUT não poderão ter nomes previamente cadastrados.

Passo a passo:

1. Enviar uma requisição PUT para /produtos/:id.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
CT032	Atualizar produto com ID inexistente	Path: ID inexistente <pre>{ "nome": "Logitech Vertical", "preco": 200, "descricao": "Teclado", "quantidade": 2 }</pre>	Response 201 Created Mensagem: “Cadastro realizado com sucesso”.
CT033	Atualizar produto com nome já utilizado	<pre>{ "nome": "Logitech Vertical", "preco": 200, "descricao": "Teclado", "quantidade": 2 }</pre>	Retorno 400 Bad Request Mensagem: “Já existe produto com esse nome”

CT034	Atualizar produto com token inválido	Token inválido <pre>{ "nome": "Logitech Vertical", "preco": 200, "descricao": "Teclado", "quantidade": 2 }</pre>	Retorno 401 Unauthorized Mensagem: “Token de acesso ausente, inválido, expirado ou usuário do token não existe mais”.
-------	--------------------------------------	-----------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------

Cenário de Teste 07

Descrição: Excluir produtos com sucesso.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.
- Produto válido

Condições:

- Usuários não autenticados não devem conseguir realizar ações na rota de Produtos;
- Não deve ser possível excluir produtos que estão dentro de carrinhos (dependência API Carrinhos).

Passo a passo:

1. Enviar uma requisição POST para /produtos e cadastrar um produto.
2. Enviar uma requisição DELETE para /produtos/:id.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
CT035	Excluir produto existente e fora de carrinho	Nenhuma	Retorno 200 OK Mensagem: Registro excluído com sucesso
CT036	Tentar excluir produto que está em um carrinho ativo	Pré-requisito: Fazer uma requisição POST/CreateCart	Retorno 400 Bad Request Mensagem: Não é permitido excluir produto que faz parte de carrinho
CT037	Tentar excluir produto sem autenticação	Token ausente ou inválido	Retorno 401 Unauthorized Mensagem: “Token de acesso ausente, inválido, expirado ou usuário do token não existe mais”.

Cenário de Teste 08

Descrição: Listar produtos com sucesso.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.

Condições:

- Usuários não autenticados não devem conseguir realizar ações na rota de Produtos.

Passo a passo:

1. Enviar uma requisição GET/produtos.

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
CT038	Listar todos os produtos	Nenhuma	Retorno 200 OK Lista de produtos
CT039	Buscar produto por ID válido	GET/produtos/:id	Retorno 200 OK Detalhes do produto
CT040	Buscar produto por ID inválido	GET/produtos/:id ID inexistente	Retorno 400 Bad Request Mensagem: Produto não encontrado
CT041	Listar produtos e token inválido	Token inválido	Retorno 400 Bad Request

4 API Carrinho [🔗](#)

Cenário de Teste 09

Descrição: Criar carrinho com sucesso.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.

Condições:

- Apenas um carrinho por usuário.

Passo a passo:

Caso o usuário já seja cadastrado, seguir para o item 2:

1. Enviar uma requisição POST/usuarios:

```
1 {  
2   "nome": "Fulano da Silva",  
3   "email": "beltrano@qa.com.br",  
4   "password": "teste",  
5   "administrador": "true"  
6 }
```

2. Enviar uma requisição POST/login:

```
1 {  
2   "email": "beltrano@qa.com.br",  
3   "password": "teste"  
4 }
```

3. Enviar uma requisição POST/carrinhos com o payload previsto na Entrada/Ação.

Casos de Teste associados:

ID	Cenário	Entrada / Ação	Resultado Esperado
CT042	Criar carrinho com um ou mais produtos válidos	{ "produtos": [{ "idProduto": "BeeJh5lz3k6kSIzA", "quantidade": 1 }, { "idProduto": "YaeJ455lz3k6kSIzA", "quantidade": 3 }] }	Retorno 201 Created Mensagem: Cadastro realizado com sucesso.
CT043	Criar carrinho sem autenticação	Token ausente, inválido ou expirado	Retorno 401 Unauthorized Mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
CT044	Criar carrinho com produto inexistente	{ "produtos": [{ "idProduto": "ACB12345abc6kSIzA", "quantidade": 1 }] }	Retorno 400 Bad Request Mensagem: ""Não é permitido possuir produto duplicado Não é permitido ter mais de 1 carrinho Produto não encontrado Produto não possui quantidade suficiente"
CT045	Criar novo carrinho quando o usuário já possui um carrinho ativo	Usuário que já tem carrinho cadastrado	Retorno 400 Bad Request Mensagem: ""Não é permitido possuir produto duplicado Não é permitido ter mais de 1 carrinho Produto não encontrado Produto não possui quantidade suficiente"

Cenário de Teste 10

Descrição: Ao criar o carrinho com sucesso, é feita a redução da quantidade no cadastro do produto inserido no carrinho.

Pré-condição:

- Usuário deve estar autenticado no sistema.
- Token válido.

Condições:

- Após a criação do carrinho com sucesso, é feita a redução da quantidade no cadastro de cada produto inserido no carrinho.

Passo a passo:

Caso o usuário já seja cadastrado, seguir para o item 2:

1. Enviar uma requisição POST/usuarios:

1 {

```
2  "nome": "Fulano da Silva",
3  "email": "beltrano@qa.com.br",
4  "password": "teste",
5  "administrador": "true"
6  }
```

2. Enviar uma requisição POST/login:

```
1  {
2    "email": "fulano@qa.com",
3    "password": "teste"
4  }
```

3. Enviar uma requisição POST/produtos:

```
1  {
2    "nome": "Logitech MX Vertical",
3    "preco": 470,
4    "descricao": "Mouse",
5    "quantidade": 2
6  }
```

4. Enviar uma requisição POST/carrinhos:

```
1  {
2    "produtos": [
3      {
4        "idProduto": "{{_idProduct}}",
5        "quantidade": 1
6      }
7    ]
8  }
```

5. Enviar uma requisição GET/produtos/{{_idProduct}}

Casos de Teste associados:

ID	Descrição	Entrada / Ação	Resultado esperado
CT046	Criar carrinho e verificar redução no estoque	Não há	Retorno 201 Created Ao acessar a rota GET/produtos/:id a quantidade deve ser igual a 1

Cenário de Teste 11

Descrição: Listar carrinhos cadastrados.

Pré-condição:

- Não há

Condições:

- Carrinhos retornados são únicos por usuário.

Passo a passo:

1. Acessar a rota: GET/carrinhos.

ID	Descrição	Entrada / Ação	Resultado esperado
CT047	Listar carrinhos existentes	Nenhuma	Retorno 200 OK

			Lista de carrinhos e lista de produtos.
Cenário de Teste 12			
Descrição: Listar carrinhos por ID.			
Pré-condição:			
<ul style="list-style-type: none">Não há.			
Condições:			
<ul style="list-style-type: none">Não há.			
Passo a passo:			
1. Acessar a rota: GET/carrinhos/:id.			
ID	Descrição	Entrada / Ação	
CT048	Buscar carrinho por ID válido	ID do carrinho (path)	Retorno 200 Carrinho encontrado Exibe detalhes do carrinho
CT049	Buscar carrinho por ID inexistente	ID inválido do carrinho (path)	Retorno 400 Carrinho não encontrado Mensagem: "Carrinho não encontrado".
Cenário de Teste 13			
Descrição: Excluir carrinhos.			
Pré-condição:			
<ul style="list-style-type: none">Usuário deve estar autenticado no sistema.Token válido.			
Condições:			
<ul style="list-style-type: none">Não há.			
Passo a passo:			
1. Acessar a rota: DELETE/carrinhos/concluir-compra.			
ID	Descrição	Entrada / Ação	Resultado
CT050	Excluir carrinho existente	Token válido	Response 200 OK Mensagem: "Registro excluído com sucesso Não foi encontrado carrinho para esse usuário"
CT051	Excluir carrinho e usuário com token inválido	Token inválido	Response 401 Token ausente, inválido ou expirado Mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"

Cenário de Teste 14			
<p>Descrição: Excluir carrinhos e retornar produtos para estoque.</p> <p>Pré-condição:</p> <ul style="list-style-type: none"> • Usuário deve estar autenticado no sistema. • Token válido. <p>Condições:</p> <ul style="list-style-type: none"> • Ao cancelar uma compra o carrinho é excluído e o estoque dos produtos desse carrinho é reabastecido. <p>Passo a passo:</p> <ol style="list-style-type: none"> 1. Acessar a rota: DELETE/carrinhos/cancelar-compra. 2. Acessar a rota: GET/produtos/:id 3. Verificar a quantidade disponível em estoque. 			
ID	Descrição	Entrada / Ação	Resultado esperado
CT052	Excluir carrinho e retornar produtos ao estoque	Token válido	<p>Response 200 Registro excluído com sucesso Não foi encontrado carrinho para esse usuário</p> <p>Mensagem: "Registro excluído com sucesso Não foi encontrado carrinho para esse usuário"</p> <p>}</p>
CT053	Excluir carrinho e usuário com token inválido	Token inválido	<p>Response 401</p> <p>Mensagem: ""Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"</p>

9. Matriz de Risco

Para a elaboração da matriz de risco, foram utilizadas as seguintes escalas:

Probabilidade: Muito baixa, Baixa, Média, Alta, Muito Alta.

Impacto: Muito baixo, Baixo, Médio, Alto e Muito Alto.

Risco: Baixo, Médio e Alto.

Caso de Teste	Probabilidade	Impacto	Risco
CT001	Baixa	Muito Alto	Médio
CT002	Baixa	Baixo	Baixo
CT003	Alta	Muito Alto	Alto
CT004	Alta	Muito Alto	Alto

CT005	Baixa	Baixo	Baixo
CT006	Baixa	Alta	Médio
CT007	Baixa	Alta	Médio
CT008	Alta	Baixo	Médio
CT009	Baixa	Baixo	Baixo
CT010	Baixa	Alta	Médio
CT011	Média	Alto	Alto
CT012	Baixa	Alto	Médio
CT013	Alta	Alto	Alto
CT014	Baixa	Muito Alto	Alto
CT015	Baixa	Média	Médio
CT016	Baixa	Alto	Médio
CT017	Média	Alta	Alto
CT018	Baixa	Baixo	Baixo
CT019	Baixa	Baixo	Baixo
CT020	Baixa	Muito Alto	Alto
CT021	Baixa	Muito Alto	Alto
CT022	Muito Alta	Muito Alto	Muito Alto
CT023	Baixa	Muito Alto	Alto
CT024	Alta	Alto	Alto
CT025	Alta	Baixo	Médio
CT026	Alta	Alto	Alto
CT027	Baixa	Muito Alto	Alto
CT028	Baixa	Médio	Médio
CT029	Média	Muito Alto	Alto
CT030	Média	Muito Alto	Alto
CT031	Baixo	Muito Alto	Médio
CT032	Alta	Muito Alto	Alto
CT033	Alta	Muito Alto	Alto
CT034	Alta	Muito Alto	Alto
CT035	Baixa	Alto	Médio
CT036	Média	Muito Alto	Alto

CT037	Média	Muito Alto	Alto
CT038	Baixa	Muito Alto	Médio
CT039	Baixa	Baixo	Baixo
CT040	Média	Muito Alto	Alto
CT041	Alta	Muito Alto	Alto
CT042	Baixa	Médio	Médio
CT043	Alta	Alto	Alto
CT044	Média	Alto	Alto
CT045	Baixa	Alta	Alto
CT046	Baixa	Baixo	Baixo
CT047	Baixa	Baixo	Baixo
CT048	Média	Baixo	Baixo
CT049	Baixa	Baixo	Baixo
CT050	Baixa	Baixo	Baixo
CT051	Média	Muito Alto	Alto
CT052	Baixa	Baixo	Baixo
CT053	Média	Muito Alto	Alto

10. Priorização da Execução dos Cenários de Teste [↗](#)

Com base na documentação da API e, principalmente, as users stories apresentadas, bem como o fluxo da API, os cenários de teste foram priorizados da seguinte forma:

- 1º) Realização dos testes com cenários de risco muito alto;
- 2º) Realização dos testes com cenários de risco alto;
- 3º) Realização dos testes com cenários de risco médio;
- 4º) Realização dos testes com cenários de risco baixo ou muito baixo;

Assim, a organização da ordem dos testes a serem realizados, com base na priorização, está descrita na tabela* abaixo:

Risco Muito Alto – Prioridade: Muito Alta [↗](#)

- CT-022 , CT-051 , CT-053

Risco Alto – Prioridade: Alta [↗](#)

- CT-003 , CT-004 , CT-011 , CT-013 , CT-014 , CT-017 , CT-020 , CT-021 , CT-023
- CT-024 , CT-026 , CT-027 , CT-029 , CT-030 , CT-032 , CT-033 , CT-034
- CT-036 , CT-037 , CT-040 , CT-041 , CT-043 , CT-044

🟡 Risco Médio – Prioridade: Média [🔗](#)

- CT-001, CT-006, CT-007, CT-008, CT-010, CT-012, CT-015, CT-016
- CT-025, CT-028, CT-031, CT-035, CT-038, CT-042, CT-045

🟢 Risco Baixo – Prioridade: Baixa [🔗](#)

- CT-002, CT-005, CT-009, CT-018, CT-019, CT-039, CT-046
- CT-047, CT-048, CT-049, CT-050, CT-052

*Tabela gerada com o uso do ChatGPT para facilitar a leitura e organização da ordem dos testes a serem realizados.

11. Cobertura de testes [🔗](#)

A cobertura de testes foi medida considerando o Path Coverage, que “verifica a cobertura da suíte de testes de acordo com os endpoints que a API possui” (CREMA, Nayara. Como verificar a cobertura de testes de APIs REST, [Medium](#)).

📊 Fórmula de Path Coverage [🔗](#)

Path Coverage (%) = (Número de URIs testadas / Número total de URIs disponíveis) × 100

📌 URIs disponíveis na API ServeRest [🔗](#)

Recurso	URI
Usuários	/usuarios
Usuários (por ID)	/usuarios/:id
Login	/login
Produtos	/produtos
Produtos (por ID)	/produtos/:id
Carrinho	/carrinhos
Cancelar compra	/carrinhos/cancelar-compra
Concluir compra	/carrinhos/concluir-compra

Total de URIs disponíveis: 8

✅ URIs testadas [🔗](#)

URI	Status
/usuarios	Testada
/usuarios/:id	Testada
/login	Testada
/produtos	Testada

/produtos/:id	Testada
/carrinhos	Testada
/carrinhos/cancelar-compra	Não testada
/carrinhos/concluir-compra	Não testada

Total de URIs testadas: 6

Resultado [🔗](#)

Path Coverage = $(6 / 8) \times 100 = 75\%$ [🔗](#)

12. Testes candidatos a automação [🔗](#)

Os testes candidatos à automação podem ser agrupados com base em diferentes tipos de validação e execução de requisições consecutivas. Foram selecionados os seguintes testes automatizados a serem implementados:

Grupo Usuários: [CT001, CT010, CT012, CT015, CT018]

Grupo Login: [CT020, CT027]

Grupo Produtos: [CT028, CT035, CT038, CT039]

Grupo Carrinhos: [CT042, CT046, CT047, CT048, CT051]