

# Atualização do Plano de Testes – API ServeRest

Projeto: <https://compassuol.serverest.dev/>

Versão: 1.0


Autor: @Thais Nogueira

Data: 5 de mai. de 2025 - 9 de mai. de 2025

- 1. Introdução
- 2. Objetivo
- 3. Escopo dos Testes
- 4. Referências
- 5. Estratégia de Teste
- 6. Mapa Mental da Aplicação
- 7. Técnicas Aplicadas
- 8. Análise de teste
- 9. Cenário de teste
  - Usuários US001
  - Login US002
  - Produto US003
  - Carrinho
- 10. Matriz de Rastreabilidade
- 11. Matriz de Risco
- 12. Priorização da execução dos cenários de teste
- 13. Cobertura de Testes
- 14. Testes Candidatos à Automação- Fluxo completo
- 

## 1. Introdução

A aplicação Serverest simula uma loja virtual oferecendo endpoints para cadastro de usuários, autenticação e gestão de produtos e gerenciamento de carrinhos.

Este plano documenta os testes realizados diretamente sobre a API pública, com base na documentação Swagger oficial  [Ser](#)  
[veRest](#)

## 2. Objetivo

Garantir que a API ServeRest esteja de acordo com os requisitos funcionais e não funcionais esperados, por meio de testes baseados nas User Stories fornecidas e na documentação Swagger. Identificar inconsistências, bugs e oportunidades de melhoria.

## 3. Escopo dos Testes

Incluído:

- Usuários: cadastro, login, atualização e remoção
- Login: autenticação
- Produtos: Cadastro, listagem, atualização, remoção
- Carrinhos de compras: testes exploratórios

Fora do escopo:

- Testes de performance
- Integrações externas

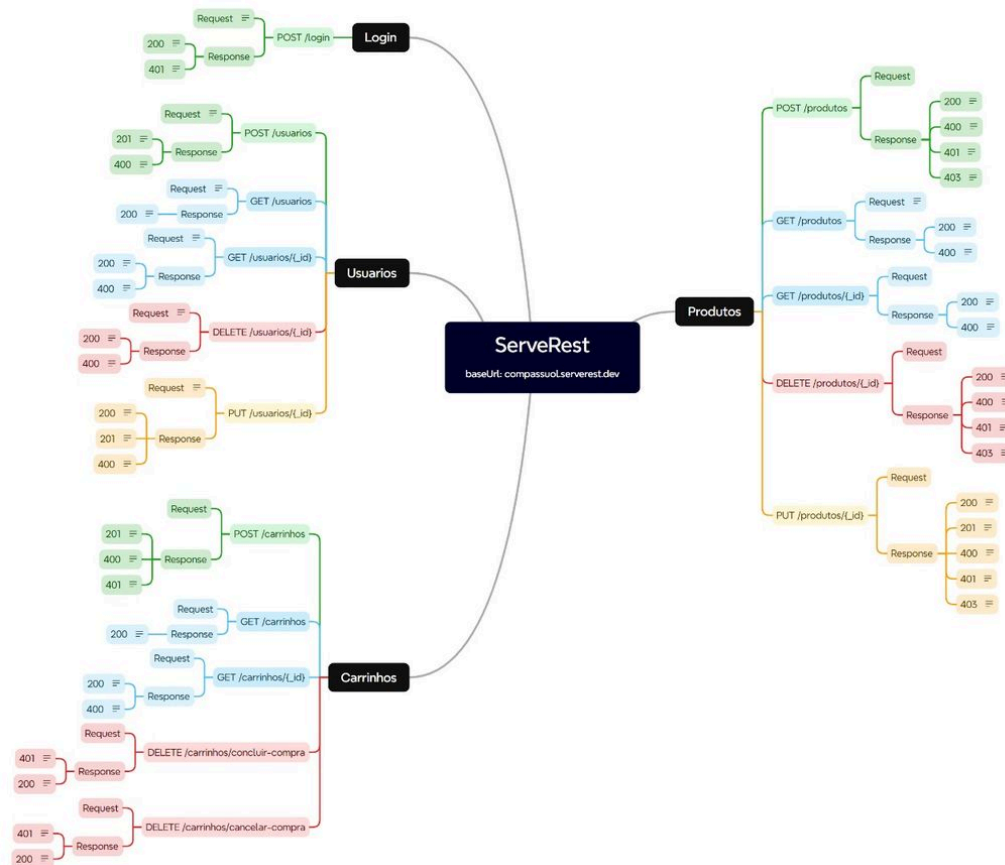
## 4. Referências [↗](#)

- Swagger: [↗ ServeRest](#)
- User Stories: [US001 \(Usuários\)](#), [US002 \(Login\)](#), [US003 \(Produtos\)](#)
- Documentação interna: Confluence
- Registro de bugs e melhorias: Jira

## 5. Estratégia de Teste [↗](#)

- **Nível:** Testes de API
- **Tipos:** Funcionais, exploratórios e validação de regras de negócio
  - **Ferramentas:** Robot Framework, Swagger, Confluence, Jira

## 6. Mapa Mental da Aplicação [↗](#)



## 7. Técnicas Aplicadas [↗](#)

- Partição de equivalência (ex: e-mail válido x inválido)
- Valor limite (ex: senhas com 4, 5, 10 caracteres)
- Teste exploratório com entradas inválidas
- Testes baseados em requisitos documentados (Swagger + User Stories)
- Análise de regras de negócio

## 8. Análise de teste

### 1. Usuários

**Funcionalidades:** (Origem: User Story US001 e Swagger)

- Criar usuários (POST /usuarios)
- Atualizar usuários (PUT /usuarios/:id)
- Listar usuários (GET /usuarios)
- Remover usuários (DELETE /usuarios/:id)

**Regras de Negócio:** (Origem: User Story US001)

- E-mail deve ser único, válido e de domínio permitido (não pode ser Gmail ou Hotmail) → Particionamento de Equivalência:
- Senha deve ter entre 5 e 10 caracteres. → Análise de Valor Limite
- Nome, E-mail, Senha e Administrador são campos obrigatórios. → testes negativos
- PUT em ID inexistente deve criar um novo usuário. → teste de fluxo
- Operações para IDs inexistentes devem retornar erro adequado. → testes negativos

**Fluxos Principais:** (Origem: Swagger)

1. Cadastro de usuário válido → Particionamento de Equivalência
2. Edição de usuário existente → Particionamento de Equivalência
3. Listagem de usuários cadastrados → Teste de Fluxo
4. Exclusão de usuário por ID. → Teste de Fluxo

**Cenários Alternativos e Inválidos:** (Origem: User Story US001)

- Tentativa de cadastro com e-mail já existente.
- Cadastro com e-mail inválido.
- Domínio não permitido (Gmail, Hotmail).
- Senha fora do limite de caracteres.
- PUT em ID inexistente (cria um novo usuário).
- Operações (GET, PUT, DELETE) em ID inválido ou inexistente.

### 2. Login

**Funcionalidades:** (Origem: User Story US002 e Swagger)

- Autenticação de usuário (POST /login)
- Geração de Token JWT válido por 10 minutos.

**Regras de Negócio:** (Origem: User Story US002)

- Apenas usuários cadastrados podem se autenticar. → Particionamento de Equivalência
- Senha incorreta retorna 401 Unauthorized. → Teste de Segurança
- Token JWT deve ser gerado e ter validade de 10 minutos. → Análise de Valor Limite

**Fluxos Principais:** (Origem: Swagger)

1. Login com credenciais corretas → autenticação bem-sucedida com token. → *Particionamento de Equivalência*

Cenários Alternativos e Inválidos: (Origem: User Story US002)

- Login com e-mail inexistente. → Testes Negativos
- Login com senha incorreta. → Testes Negativos
- Login com campos vazios. → Testes Negativos

- Login com e-mail malformatado. → Particionamento de Equivalência
- Expiração do token após 10 minutos. → Análise de Valor Limite

### 3. Produtos

**Funcionalidades:** *(Origem: User Story US003 e Swagger)*

- Cadastrar produtos (POST /produtos)
- Atualizar produtos (PUT /produtos/:id)
- Listar produtos (GET /produtos)
- Excluir produtos (DELETE /produtos/:id)

**Regras de Negócio:** *(Origem: User Story US003)*

- Somente usuários autenticados podem gerenciar produtos. → teste de segurança
- Não é permitido cadastrar produtos com nomes duplicados. → Particionamento de Equivalência:
- Não é permitido excluir produtos que estejam em carrinhos. → Particionamento de Equivalência:
- PUT em ID inexistente deve criar um novo produto.  
CRUD

**Fluxos Principais:** *(Origem: Swagger)*

1. Cadastro de produto válido. → *Particionamento de Equivalência*
2. Edição de produto existente. → *Particionamento de Equivalência*
3. Listagem de produtos cadastrados. → *Teste de Fluxo*
4. Exclusão de produto por ID. → *Teste de Fluxo*

**Cenários Alternativos e Inválidos:** *(Origem: User Story US003)*

- Acesso sem autenticação.
- Cadastro com nome já existente.
- PUT com ID inexistente (deve criar).
- PUT com nome já existente (deve falhar).
- DELETE de produto que está em um carrinho.

### 4. Carrinho

**Funcionalidades:** *(Origem: Swagger)*

- Criar carrinho (POST /carrinhos)
- Listar carrinhos (GET /carrinhos)
- Adicionar e remover produtos em um carrinho (PUT /carrinhos/:id)
- Excluir carrinho (DELETE /carrinhos/:id)

**Regras de Negócio:** *(Origem: Swagger)*

- Apenas usuários autenticados podem criar carrinhos. → Teste de Segurança
- Só é possível adicionar produtos existentes. → Particionamento de Equivalência
- Produtos em carrinho não podem ser excluídos pela API de produtos. → Teste de Fluxo
- Finalizar carrinho só é permitido se ele tiver produtos válidos. → Teste de Fluxo

**Fluxos Principais:** *(Origem: Swagger)*

1. Criar carrinho com produtos válidos.
2. Adicionar e remover produtos.

3. Finalizar compra com sucesso.
4. Listar carrinho e verificar conteúdo.

#### Cenários Alternativos e Inválidos: (Origem: Swagger)

- Criar carrinho sem autenticação. → *Teste de Segurança*
- Adicionar produto inexistente. → *Particionamento de Equivalência*
- Tentar deletar produto em carrinho. → *Teste de Condição de Fronteira*
- Finalizar carrinho vazio. → *Testes Negativos*
- Enviar dados incompletos (campos obrigatórios ausentes). → *Testes Negativos*

## 9. Cenário de teste [↗](#)

### Usuários US001 [↗](#)

#### US001 Cenário de Teste 01 - Cadastro de Usuário

**Descrição:** Validar o processo de cadastro de usuários com diferentes combinações de dados válidos e inválidos.

**Pré-condição:** API disponível e ambiente de teste configurado.

**Condições:**

- Os campos obrigatórios devem estar preenchidos corretamente: nome, e-mail, password e administrador.
- O e-mail deve seguir um padrão válido (ex.: user@dominio.com).
- O domínio do e-mail não pode ser Gmail ou Hotmail.
- A senha deve ter entre 5 e 10 caracteres.

**Passo a passo:**

1. Enviar uma requisição POST /usuarios.
2. Informar os seguintes dados: nome, e-mail, password e administrador.

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-001	Cadastro de usuário válido	API disponível e ambiente configurado	1. Preparar dados válidos: nome, e-mail, senha e administrador 2. Montar payload JSON com os dados 3. Enviar requisição POST para /usuarios 4. Validar status code 201 5. Validar presença de _id e mensagem de sucesso na resposta	{ "nome": "Ana Silva", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	201 { "message": "Cadastro realizado com sucesso", "_id": "" }	Alta
CT-002	Cadastro com e-mail já existente	Usuário já cadastrado no sistema	1. Cadastrar um usuário com um e-mail válido 2. Tentar cadastrar outro usuário com o mesmo e-mail 3. Enviar requisição POST para /usuarios	{ "nome": "João Silva", "email": "Anateste@teste.com.br", "password": "teste",	Erro: "E-mail já cadastrado."	Alta

			4. Validar status code 400 5. Validar mensagem de erro sobre e-mail já existente	"administrador": "true" }		
CT-003	Cadastro com e-mail inválido	API disponível e ambiente configurado	1. Montar payload com e-mail malformatado 2. Enviar requisição POST para /usuarios 3. Validar status code 400 4. Verificar mensagem: email deve ser um email válido	{ "nome": "João Silva", "email": "joaoteste.com.br", "password": "teste", "administrador": "true" }	Status code 400 Bad Request { "email": "email deve ser um email válido" } }	Baixa
CT-004	Cadastro com domínio bloqueado (Gmail, Hotmail)	API disponível e ambiente configurado	1. Montar payload com e-mail do tipo <a href="mailto:joao@gmail.com">gmail.com</a> e hotmail.com 2. Enviar requisição POST para /usuarios 3. Validar status code 400 4. Validar mensagem de erro sobre domínio não permitido	{ "nome": "João Silva", "email": "joao@gmail.com", "password": "teste", "administrador": "true" }	400 (Bad Request) "Domínio de e-mail não permitido."	Media
CT-005	Cadastro sem campos obrigatórios	API disponível e ambiente configurado	1. Montar payload com nome, e-mail, senha e administrador vazios 2. Enviar requisição POST para /usuarios 3. Validar status code 400 4. Validar que todos os campos são indicados como obrigatórios na resposta	{ "nome": "", "email": "", "password": "", "administrador": "" }	400 (Bad Request) "Campos obrigatórios não informados."	Alta
CT-006	Cadastro com senha fora do limite permitido	API disponível e ambiente configurado	1. Montar payload com senha menor que 5 ou maior que 10 caracteres 2. Enviar requisição POST para /usuarios 3. Validar status code 400 4. Verificar mensagem de erro relacionada à senha	{ "nome": "Maria", "email": "maria@teste.com", "password": "123", "administrador": "true" }	Erro: "A senha deve ter entre 5 e 10 caracteres."	Média

#### US001 Cenário de Teste 02 - Editar Usuário

**Descrição:** Validar o processo de atualização de usuários já cadastrados e o comportamento para diferentes cenários.

**Pré-condição:** O usuário deve estar previamente cadastrado no sistema

##### Condições:

- O usuário deve estar cadastrado no sistema.
- O e-mail informado deve ser válido e não duplicado.
- Caso o ID não exista, um novo usuário deve ser criado.

##### Passo a passo:

1. Enviar uma requisição PUT/usuarios/{\_id}
2. Preencher o payload com os dados atualizados do usuário: nome, e-mail, password e administrador.

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-007	Alterar usuário válido	Usuário cadastrado no sistema	<ol style="list-style-type: none"> <li>1. Cadastrar um usuário previamente</li> <li>2. Montar payload com novo nome e dados válidos</li> <li>3. Enviar requisição PUT para /usuarios/{_id} com o ID do usuário</li> <li>4. Validar status code 200 Verificar mensagem de sucesso na resposta</li> </ol>	_id:qTQ4ick6SPLCGer5  { "nome": "Ana", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	Status code 200 OK  {"message": "Registro alterado com sucesso"}	Alta
CT-008	Alterar usuário com e-mail duplicado	Outro usuário já existe com o mesmo e-mail	<ol style="list-style-type: none"> <li>1. Cadastrar dois usuários distintos</li> <li>2. Tentar atualizar o segundo usuário com o e-mail do primeiro</li> <li>3. Enviar requisição PUT para /usuarios/{_id} com o ID do segundo usuário</li> <li>4. Validar status code 400</li> <li>5. Verificar mensagem de erro sobre e-mail duplicado</li> </ol>	{ "nome": "Ana Silva", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	status code 400 bad request  Erro: "E-mail já cadastrado."	Alta
CT-009	Alterar usuário sem campos obrigatórios		<ol style="list-style-type: none"> <li>1. Cadastrar um usuário previamente</li> <li>2. Montar payload com nome, e-mail, senha e administrador vazios</li> <li>3. Enviar requisição PUT para /usuarios/{_id} com o ID do usuário</li> <li>4. Validar status code 400</li> <li>5. Verificar mensagens de erro indicando os campos obrigatórios ausentes</li> </ol>	{ "nome": "", "email": "", "password": "", "administrador": "" }	status code 400 bad request  Erro: "Campos obrigatórios não informados."	Média
CT-010	Alterar usuário com e-mail inválido		<ol style="list-style-type: none"> <li>1. Cadastrar um usuário previamente</li> <li>2. Montar payload com e-mail inválido ex: sem @</li> </ol>	{ "nome": "Ana", "email": "Anateste.com.br", "password": "teste", "administrador": "true" }	status code 400 bad request Erro: "Formato de e-mail inválido."	Média

			3. Enviar requisição PUT para /usuarios/{_id} com o ID do usuário 4. Validar status code 400 5. Verificar mensagem de erro sobre e-mail inválido			
CT-011	Alterar usuário em ID inexistente	ID não encontrado no banco de dados	1. Montar payload com dados válidos 2. Enviar requisição PUT para /usuarios/{_id} com um ID inexistente 3. Validar status code 201 4. Verificar se um novo usuário foi criado com os dados enviados	{ "nome": "Novo Usuário", "email": "novo@teste.com", "password": "54321", "administrador": "true" }	Usuário criado com sucesso.	Baixa

#### US001 Cenário de Teste 03 - Listagem de Usuários

**Descrição:** Validar o processo de listagem de usuários cadastrados no sistema.

**Pré-condição:** usuários previamente cadastrados.

**Condições:**

- O usuário deve estar cadastrado no sistema.
- Caso não existam usuários, a lista deve retornar vazia.

**Passo a passo:**

1. Enviar uma requisição GET /usuarios.
2. Validar retorno de sucesso e dados carregados.

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-012	Listagem de usuários válida	Usuários cadastrados no sistema	1. Cadastrar pelo menos um usuário previamente 2. Enviar requisição GET para /usuarios 3. Validar status code 200 4. Verificar que a lista de usuários foi retornada com os dados esperados	-	Lista de usuários cadastrados.	Baixa
CT-013	Listagem de usuários sem autenticação	-	1. Enviar requisição GET para /usuarios sem token de autenticação 2. Validar status code 401 3. Verificar se a listagem for permitida sem autenticação	-	Erro: "Não autorizado."	Média
CT-014	Listagem de usuários	-	1. Enviar requisição GET para /usuarios_id, com ID inexistente ou malformatado 2. Validar status code 200	Em parâmetros	Lista de usuário vazia	Média



	com ID inválido		3. Verificar se a resposta contém uma lista vazia ou mensagem adequada	inserir Key : _id Value: 1234		
--	-----------------	--	--	-------------------------------------	--	--

#### US001 Cenário de Teste 04 - Exclusão de Usuários

**Descrição:** Validar o processo de exclusão de um usuário existente no sistema.

**Pré-condição:** O usuário deve estar previamente cadastrado no sistema.

**Condições:**

- O usuário deve estar cadastrado no sistema.
- O ID informado deve ser válido

**Passo a passo:**

1. Enviar uma requisição DELETE /usuarios/{\_id}
2. Verificar se o usuário foi removido da listagem de usuários.

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-015	Exclusão de usuário válido	Usuário existente e autenticado	<ol style="list-style-type: none"> <li>1. Cadastrar um usuário previamente</li> <li>2. Enviar requisição DELETE para /usuarios/{_id} com o ID do usuário</li> <li>3. Validar status code 200</li> <li>4. Verificar mensagem de sucesso na resposta</li> </ol>	ID: UQEPVoJ0rG5Jgj4E	Mensagem: Registro excluído com sucesso	Baixa
CT-016	Exclusão de usuário inexistente	-	<ol style="list-style-type: none"> <li>1. Enviar requisição DELETE para /usuarios/{_id} com um ID inexistente</li> <li>2. Validar status code 400</li> <li>3. Verificar mensagem de erro informando que o usuário não foi encontrado</li> </ol>	-	Erro: "Usuário não encontrado."	Média
CT-017	Exclusão de usuário sem autenticação	API disponível e ambiente de teste configurado	<ol style="list-style-type: none"> <li>1. Cadastrar um usuário previamente</li> <li>2. Enviar requisição DELETE para /usuarios/{_id} sem incluir token JWT</li> <li>3. Validar status code 401</li> <li>4. Verificar mensagem de erro indicando falta de autenticação</li> </ol>	-	Erro: "Não autorizado."	Média

## US002 Cenário de Teste 01 - Login de Usuário

**Descrição:** Validar o processo de login de usuários no sistema, gerando o token JWT para acesso às funcionalidades autenticadas.




**Pré-condição:** O usuário deve estar previamente cadastrado no sistema com um e-mail e senha válidos.

### Condições:

- O e-mail deve existir no banco de dados.
- A senha informada deve estar correta.
- O retorno deve conter o token JWT com validade de 10 minutos.

### Passo a passo:

1. Enviar uma requisição POST /login.
2. Preencher o payload com as credenciais: E-mail e senha

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-018 	Login com credenciais válidas	Usuário existente no sistema	<ol style="list-style-type: none"><li>1. Cadastrar usuário com e-mail e senha válidos</li><li>2. Montar payload com as credenciais</li><li>3. Enviar requisição POST para /login</li><li>4. Validar status code 200</li><li>5. Verificar presença do token JWT na resposta</li></ol>	{ "email": "Anateste@teste.com.br", "password": "teste" }	status code 200  { "message": "Login realizado com sucesso", "authorization": "Bearer " }	Alta
CT-019 	Login com e-mail não cadastrado	Nenhum usuário cadastrado no sistema	<ol style="list-style-type: none"><li>1. Montar payload com e-mail não cadastrado</li><li>2. Enviar requisição POST para /login</li><li>3. Validar status code 401</li><li>4. Verificar mensagem de erro indicando credenciais inválidas</li></ol>	{ "email": "naoexiste@teste.com", "password": "teste" }	status code 401  Erro: "Credenciais inválidas."	Alta
CT-020 	Login com senha incorreta	Usuário existente no sistema	<ol style="list-style-type: none"><li>1. Cadastrar usuário com senha válida</li><li>2. Montar payload com senha incorreta</li><li>3. Enviar requisição POST para /login</li><li>4. Validar status code 401</li><li>5. Verificar mensagem de erro indicando falha de autenticação</li></ol>	{ "email": "Anateste@teste.com.br", "password": "senhaincorreta" }	status code 401 Erro: "Credenciais inválidas."	Alta
CT-021	Login com campos	API disponível e	<ol style="list-style-type: none"><li>1. Montar payload com e-mail e senha vazios</li></ol>	{ "email": "", "password": "" }	Erro: "Campos obrigatórios não"	Baixa

	vazios	ambiente de teste configurado	2. Enviar requisição POST para /login 3. Validar status code 400 4. Verificar mensagens de erro para os campos obrigatórios		preenchidos."	
CT-022	Login com e-mail mal formatado		1. Montar payload com e-mail inválido sem @ 2. Enviar requisição POST para /login 3. Validar status code 400 4. Verificar mensagem de erro para o campo e-mail	{ "email": "Anateste.com.br", "password": "teste" }	Erro: "Formato de e-mail inválido."	Baixa
CT-023	Login com payload vazio		1. Enviar requisição POST para /login com payload vazio 2. Validar status code 400 3. Verificar mensagens de erro sobre campos obrigatórios	{ }	"Campos obrigatórios não preenchidos."	Baixa
CT-024	Login com estrutura de payload incorreta		1. Montar payload com atributo password apenas, omitindo o email 2. Enviar requisição POST para /login 3. Validar status code 400 4. Verificar mensagem de erro sobre campo obrigatório ausente	{ "password": "12345" }	Erro: "Campo 'email' não informado."	Baixa
CT-025	Expiração de token após 10 minutos	Token gerado no login	1. Realizar login e salvar o token gerado 2. Aguardar 10 minutos 3. Tentar acessar um endpoint protegido com o token expirado 4. Validar status code 401 5. Verificar mensagem de erro informando que o token expirou	{ "Authorization": "Bearer <token_gerado>" }	Erro: "Token expirado."	Alta

## Produto US003 [🔗](#)

### US003 Cenário de Teste 01 - Cadastro de Produto

**Descrição:** Validar o processo de criação de produtos no sistema.


**Pré-condição:** API disponível, usuário autenticado e ambiente de teste configurado.

**Condições:**

- O usuário deve estar autenticado e possuir um token válido.
- O nome do produto não pode estar duplicado..
- todos os campos obrigatórios devem estar preenchidos corretamente:nome,preço,descricao e quantidade

**Passo a passo:**

1. Enviar uma requisição POST /produtos
2. Preencher o payload com os dados do produto

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-026 	Cadastro de produto válido	Usuário autenticado no sistema	<ol style="list-style-type: none"> <li>1. Autenticar usuário e obter token JWT</li> <li>2. Montar payload com nome, preço, descrição e quantidade válidos</li> <li>3. Enviar requisição POST para /produtos com token no header</li> <li>4. Validar status code 201</li> <li>5. Verificar mensagem de sucesso e presença do ID do produto</li> </ol>	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": 10 }	"Cadastro realizado com sucesso"	Alta
CT-027	Cadastro de produto com nome duplicado	Produto já existente com o mesmo nome	<ol style="list-style-type: none"> <li>1. Cadastrar um produto com nome específico</li> <li>2. Montar payload com o mesmo nome e outros dados válidos</li> <li>3. Enviar requisição POST para /produtos</li> <li>4. Validar status code 400</li> <li>5. Verificar mensagem de erro informando que o produto já existe</li> </ol>	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": 10 }	Erro: "Produto já cadastrado."	Alta
CT-028	Cadastro de produto com preço negativo	API disponível e ambiente de teste configurado	<ol style="list-style-type: none"> <li>1. Montar payload com preço negativo</li> <li>2. Enviar requisição POST para /produtos</li> <li>3. Validar status code 400</li> <li>4. Verificar mensagem de erro para o campo preço</li> </ol>	{ "nome": "Celular Samsung", "preco": -1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": 10 }	Erro: "Preço inválido."	Média
CT-029	Cadastro de produto com quantidade negativa	API disponível e ambiente de teste configurado	<ol style="list-style-type: none"> <li>1. Montar payload com quantidade negativa</li> <li>2. Enviar requisição POST para /produtos</li> <li>3. Validar status code 400</li> </ol>	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": -10 }	Erro: "Quantidade inválida."	Média

			4. Verificar mensagem de erro indicando campo quantidade inválido			
CT-030	Cadastro de produto com campos obrigatórios faltando	API disponível e ambiente de teste configurado	<ol style="list-style-type: none"> <li>1. Montar payload com nome, preço, descrição e quantidade vazios</li> <li>2. Enviar requisição POST para /produtos</li> <li>3. Validar status code 400</li> <li>4. Verificar mensagens de erro para cada campo obrigatório</li> </ol>	{ "nome": "", "preco": "", "descricao": "", "quantidade": "" }	Erro: "Campos obrigatórios não informados."	Baixa

### US003 Cenário de Teste 02 - Atualização de Produto

**Descrição:** Validar o processo de atualização de produtos no sistema..

**Pré-condição:** API disponível, usuário autenticado e ambiente de teste configurado.

#### Condições:

- O usuário deve estar autenticado e possuir um token válido.
- O produto deve existir no banco de dados para ser atualizado.
- O nome do produto não pode ser alterado para um nome já existente em outro produto.
- Caso o ID não exista, um novo produto deverá ser criado.

#### Passo a passo:

1. Enviar uma requisição PUT/produtos/:id.
2. Preencher o payload com os dados do produto: nome,preço, descricao,quantidade

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-031	Atualização de produto válido	Produto existente no sistema	<ol style="list-style-type: none"> <li>1. Cadastrar um produto previamente</li> <li>2. Montar payload com dados atualizados</li> <li>3. Enviar requisição PUT para /produtos/{_id} com token JWT</li> <li>4. Validar status code 200</li> <li>5. Verificar mensagem de sucesso e alterações aplicadas</li> </ol>	{ "nome": "Celular Samsung Atualizado", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy Atualizado", "quantidade": 10 }	JSON contendo: nome , preco , descricao , quantidade	Alta
CT-032	Atualização de produto	Produto existente com o mesmo nome	<ol style="list-style-type: none"> <li>1. Cadastrar dois produtos distintos</li> </ol>	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao":	Erro: "Nome do produto já cadastrado."	Alta

	com nome duplicado		2. Tentar atualizar o segundo com o nome do primeiro 3. Enviar requisição PUT para /produtos/{_id} 4. Validar status code 400 5. Verificar mensagem de erro sobre nome duplicado	"Smartphone Galaxy", "quantidade": 10 }		
CT-033	Atualização de produto sem autenticação	API disponível e ambiente de teste configurado	1. Cadastrar um produto previamente 2. Montar payload com novos dados 3. Enviar requisição PUT para /produtos/{_id} sem autenticação 4. Validar status code 401 5. Verificar mensagem de erro por ausência de token	{ "nome": "Celular Samsung", "preco": 1800.00, "descricao": "Smartphone Galaxy", "quantidade": 10 }	Erro: "Não autorizado."	Média
CT-034	Atualização de produto com campos obrigatórios faltando	Produto existente no sistema	1. Montar payload com todos os campos obrigatórios em branco 2. Enviar requisição PUT para /produtos/{_id} 3. Validar status code 400 4. Verificar mensagens de erro para todos os campos obrigatórios	{ "nome": "", "preco": "", "descricao": "", "quantidade": "" }	Erro: "Campos obrigatórios não informados."	Média
CT-035	Atualização de produto em ID inexistente	ID não existe no banco de dados	1. Montar payload com dados válidos 2. Enviar requisição PUT para /produtos/{_id} com ID inexistente 3. Validar status code 201 4. Verificar que um novo produto foi criado	/_id: Wbg12g45gd455we  { "nome": "Celular Inexistente", "preco": 2000.00, "descricao": "Novo Produto", "quantidade": 5 }	Produto criado com sucesso.	Baixa

#### US003 Cenário de Teste 03 - Listagem de Produtos

**Descrição:** Validar o processo de listagem de produtos no sistema.

**Pré-condição:** API disponível, usuário autenticado e ambiente de teste configurado.

#### Condições:

- O usuário deve estar autenticado e possuir um token válido.
- Caso não existam produtos cadastrados, a lista deve retornar vazia

**Passo a passo:**

1. Enviar uma requisição GET /produtos

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-036	Listagem de produtos válida	Produtos cadastrados e usuário autenticado	1. Autenticar usuário e obter token JWT 2. Enviar requisição GET para /produtos com token JWT 3. Validar status code 200 4. Verificar que a lista contém produtos válidos	-	Lista de produtos gerados	Alta
CT-037	Listagem sem autenticação	Sem autenticação	1. Enviar requisição GET para /produtos sem autenticação 2. Validar status code 401 3. Verificar mensagem de erro indicando falta de autenticação	-	Erro: "Não autorizado."	Média
CT-038	Listagem com ID inválido		1. Enviar requisição GET para /produtos_id= 2. Validar status code 200 3. Verificar que a lista está vazia ou resposta apropriada	Em parametro Key: _id value:123	Lista de usuario vazia	Média

**US003 Cenário de Teste 04 - Exclusão de Produto**

**Descrição:** Validar o processo de exclusão de produtos no sistema

**Pré-condição:** API disponível, usuário autenticado, produto existente e ambiente de teste configurado.

**Condições:**

- O usuário deve estar autenticado e possuir um token válido.
- O produto deve existir no banco de dados.
- Não deve estar em um carrinho vinculado para permitir exclusão.

**Passo a passo:**

1. Enviar uma requisição DELETE/produtos/:id.
2. Verificar se o produto foi removido da listagem de produtos

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-039	Exclusão de produto válido	1. Cadastrar produto sem vínculo com carrinho	1. Enviar requisição para excluir o produto.	_id: jcAMnCpGiqEFAE7S	Produto removido do sistema.	Alta

		2. Enviar requisição DELETE para /produtos/{_id} com autenticação 3. Validar status code 200 4. Verificar mensagem de sucesso	2. Validar retorno de sucesso.			
CT-040	Exclusão de produto vinculado a um carrinho	Produto associado a um carrinho	1. Criar carrinho com um produto 2. Tentar excluir esse produto com DELETE /produtos/{_id} 3. Validar status code 400 4. Verificar mensagem indicando vínculo com carrinho	Id: lJRn5q3HI8djD3gTlJRn5q3HI8djD3gT	Erro: "Produto vinculado a um carrinho. Não pode ser excluído."	Alta
CT-041	Exclusão de produto sem autenticação		1. Cadastrar um produto 2. Enviar requisição DELETE para /produtos/{_id} sem token JWT 3. Validar status code 401 4. Verificar mensagem de erro	-id: lJRn5q3HI8djD3gT	Erro: "Não autorizado."	Baixa
CT-042	Exclusão de produto com ID inválido		1. Enviar requisição DELETE para /produtos/{_id} com ID inválido 2. Validar status code 200 3. Verificar mensagem de que nenhum registro foi excluído	id:Ss1234fhg158dg15	"Produto não encontrado."	Baixa

## Carrinho [🔗](#)

### Cenário de Teste 01- Criação de Carrinho de Compras

**Descrição:** Validar o processo de criação de um carrinho de compras no sistema.

**Pré-condição:** PI disponível, usuário autenticado e ambiente de teste configurado.


#### Condições:

- O usuário deve estar autenticado e possuir um token válido.
- Os produtos devem existir no banco de dados.
- Não é permitido adicionar produtos duplicados no carrinho.

#### Passo a passo:

1. Enviar uma requisição POST carrinhos.
2. Preencher o payload com os dados dos produtos: Id do produto e quantidade



ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-043 	Criação de carrinho válido	Usuário autenticado e produto existente	<ol style="list-style-type: none"> <li>1. Cadastrar produto e obter ID</li> <li>2. Autenticar usuário</li> <li>3. Montar payload com idProduto e quantidade</li> <li>4. Enviar requisição POST para /carrinhos</li> <li>5. Validar status code 201</li> <li>6. Verificar ID do carrinho na resposta</li> </ol>	<pre>{   "produtos": [     {       "idProduto": "lJRn5q3HI8djD3gT",       "quantidade": 1     }   ] }</pre>	produto adicionado ao carrinho	Alta
CT-044	Criação de carrinho sem autenticação		<ol style="list-style-type: none"> <li>1. Montar payload com produto válido</li> <li>2. Enviar requisição POST para /carrinhos sem autenticação</li> <li>3. Validar status code 401</li> <li>4. Verificar mensagem de erro de autenticação</li> </ol>	<pre>{   "produtos": [     {       "idProduto": "lJRn5q3HI8djD3gT",       "quantidade": 1     }   ] }</pre>	Erro: "Não autorizado."	Média
CT-045	Criação de carrinho com produto inexistente	Produto não está cadastrado no sistema	<ol style="list-style-type: none"> <li>1. Montar payload com ID de produto inexistente</li> <li>2. Enviar requisição POST para /carrinhos</li> <li>3. Validar status code 400</li> <li>4. Verificar mensagem de erro</li> </ol>	<pre>{ "produtos": [ {   "idProduto": "produtoInvalido",   "quantidade": 2 } ] }</pre>	Erro: "Produto não encontrado."	Média
CT-046	Criação de carrinho com produto duplicado	Produto duplicado no mesmo carrinho	<ol style="list-style-type: none"> <li>1. Montar payload com dois itens com mesmo idProduto</li> <li>2. Enviar requisição POST para /carrinhos</li> <li>3. Validar status code 400</li> <li>4. Verificar mensagem de erro sobre produto duplicado</li> </ol>	<pre>{ "produtos": [ {   "idProduto": "umproduto",   "quantidade": 2 }, {   "idProduto": "umproduto",   "quantidade": 2 } ] }</pre>	Erro: "Produto duplicado no carrinho."	Média

#### Cenário de Teste 02- Visualização do Carrinho

**Descrição:** Validar o processo de visualização dos itens no carrinho..

**Pré-condição:** API disponível, usuário autenticado, carrinho criado previamente e ambiente de teste configurado.

##### Condições:

1. O usuário deve estar autenticado e possuir um token válido.
2. O carrinho deve existir no banco de dados.

##### Passo a passo:

1. Enviar uma requisição GET /carrinhos/:id.

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-047	Visualização de carrinho válido	Carrinho criado previamente	<ol style="list-style-type: none"> <li>1. Criar carrinho e obter ID</li> <li>2. Autenticar usuário</li> <li>3. Enviar requisição GET para /carrinhos/{_id}</li> <li>4. Validar status code 200</li> <li>5. Verificar se produtos estão presentes na resposta</li> </ol>	"_id": "Qwieyo2PAozlXS7Y"	visualizar carrinho	Alta
CT-048	Visualização de carrinho inexistente		<ol style="list-style-type: none"> <li>1. Enviar requisição GET para /carrinhos/{_id} com ID inexistente</li> <li>2. Validar status code 400</li> <li>3. Verificar mensagem de erro</li> </ol>	ID: Qwieyo2PAozlXS48	Erro: "Carrinho não encontrado."	Média
CT-049	Visualização de carrinho sem autenticação	API disponível e ambiente de teste configurado	<ol style="list-style-type: none"> <li>1. Enviar requisição GET para /carrinhos/{_id} sem autenticação</li> <li>2. Validar status code 401</li> <li>3. Verificar mensagem de erro</li> </ol>	-	Erro: "Não autorizado."	Média

### Cenário de Teste 03- Finalização de compra

**Descrição:** Validar o processo de finalização de compra no sistema.


**Pré-condição:** usuário autenticado, carrinho criado previamente e ambiente de teste configurado.

#### Condições:

1. O usuário deve estar autenticado e possuir um token válido.
2. O carrinho deve existir no banco de dados.

#### Passo a passo:

1. Enviar uma requisição DELETE /carrinhos/concluir-compra
2. Verificar se o carrinho foi esvaziado

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-050 	Finalização de compra com carrinho válido	Carrinho criado previamente e autenticado	<ol style="list-style-type: none"> <li>1. Criar carrinho com produtos válidos</li> <li>2. Autenticar usuário</li> <li>3. Enviar requisição DELETE para /carrinhos/concluir-compra</li> <li>4. Validar status code 200</li> </ol>	-	"Carrinho excluído com sucesso"	Alta

			5. Verificar mensagem de sucesso			
CT-051	Finalização de compra com carrinho inexistente		1. Enviar requisição DELETE para /carrinhos/concluir-compra com usuário sem carrinho 2. Validar mensagem de erro sobre carrinho inexistente	-	Erro: "Carrinho não encontrado."	Média
CT-052	Finalização de compra sem autenticação		1. Criar carrinho como usuário autenticado 2. Enviar requisição DELETE para /carrinhos/concluir-compra sem autenticação 3. Validar status code 401 4. Verificar mensagem de erro	-	Erro: "Não autorizado."	Média
CT-053	Finalização de compra e validação de esvaziamento	Compra finalizada previamente	1. Finalizar compra de carrinho existente 2. Enviar GET para /carrinhos/{_id} 3. Validar que carrinho não existe mais	-	Retorno de carrinho vazio após finalizar compra.	Alta

#### Cenário de Teste 04- Cancelamento de Compra

**Descrição:** Validar o processo de cancelamento de compra, onde os produtos são devolvidos ao estoque

**Pré-condição:** usuário autenticado, carrinho criado previamente

##### Condições:

1. O usuário deve estar autenticado e possuir um token válido.
2. O carrinho deve existir no banco de dados.

##### Passo a passo:

1. Enviar uma requisição DELETE /carrinhos/cancelar-compra
2. Verificar se o carrinho foi esvaziado e retornado ao estoque

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade
CT-054	Cancelamento de compra com carrinho válido	Carrinho criado previamente e autenticado	1. Enviar requisição DELETE para /carrinhos/cancelar-compra 2. Validar status code 200 3. Verificar mensagem de cancelamento e estoque reabastecido		"Compra cancelada com sucesso e produtos retornados ao estoque"	Alta

CT-055	Cancelamento de compra com carrinho inexistente		1. Enviar requisição DELETE para /carrinhos/cancelar-compra com usuário sem carrinho 2. Validar mensagem informando ausência de carrinho		Erro: "Carrinho não encontrado."	Média
CT-056	Cancelamento de compra sem autenticação		1. Criar carrinho com autenticação 2. Enviar requisição DELETE para /carrinhos/cancelar-compra sem autenticação 3. Validar status code 401 4. Verificar mensagem de erro		Erro: "Não autorizado."	Alta

## 10. Matriz de Rastreabilidade [🔗](#)

User Story	Casos de Teste	Cobertura
US001 - Cadastro de Usuário	CT-001 a CT-006	Sim
US001 - Atualização de Usuário	CT-007 a CT-011	Sim
US001 - Listagem de Usuário	CT-012 a CT-014	Sim
US001 - Exclusão de Usuário	CT-015 a CT-017	Sim
US002 - Login de Usuário	CT-018 a CT-025	Sim
US003 - Cadastro de Produto	CT-026 a CT-030	Sim
US003 - Atualização de Produto	CT-031 a CT-035	Sim
US003 - Listagem de Produto	CT-036 a CT-038	Sim
US003 - Exclusão de Produto	CT-039 a CT-042	Sim
Carrinho - Criação	CT-043 a CT-046	Sim
Carrinho - Visualização	CT-047 a CT-049	Sim
Carrinho - Finalização de Compra	CT-050 a CT-053	Sim
Carrinho - Cancelamento de Compra	CT-054 a CT-056	Sim

## 11. Matriz de Risco [🔗](#)

ID	Descrição	Impacto	Probabilidade	Nível de Risco
----	-----------	---------	---------------	----------------

CT-001	Cadastro de usuário válido	Alto	Alta	Crítico
CT-002	Cadastro com e-mail já existente	Médio	Média	Moderado
CT-003	Cadastro com e-mail inválido	Baixo	Baixa	Baixo
CT-004	Cadastro com domínio bloqueado	Médio	Média	Moderado
CT-005	Cadastro sem campos obrigatórios	Alto	Alta	Crítico
CT-006	Cadastro com senha fora do limite permitido	Médio	Média	Moderado
CT-007	Atualização de usuário válido	Médio	Média	Moderado
CT-008	Atualização de usuário com e-mail duplicado	Médio	Média	Moderado
CT-009	Atualização de usuário sem campos obrigatórios	Médio	Média	Moderado
CT-010	Atualização de usuário com e-mail inválido	Médio	Média	Moderado
CT-011	Atualização de usuário em ID inexistente	Baixo	Baixa	Baixo
CT-012	Listagem de usuários válida	Baixo	Baixa	Baixo
CT-013	Listagem de usuários sem autenticação	Médio	Média	Moderado
CT-014	Listagem de usuários com ID inválido	Médio	Média	Moderado
CT-015	Exclusão de usuário válido	Baixo	Baixa	Baixo
CT-016	Exclusão de usuário inexistente	Médio	Média	Moderado
CT-017	Exclusão de usuário sem autenticação	Médio	Média	Moderado
CT-018	Login com credenciais válidas	Alto	Alta	Crítico
CT-019	Login com e-mail não cadastrado	Alto	Alta	Crítico

CT-020	Login com senha incorreta	Alto	Alta	Crítico
CT-021	Login com campos vazios	Baixo	Baixa	Baixo
CT-022	Login com e-mail mal formatado	Baixo	Baixa	Baixo
CT-023	Login com payload vazio	Baixo	Baixa	Baixo
CT-024	Login com estrutura de payload incorreta	Baixo	Baixa	Baixo
CT-025	Expiração de token após 10 minutos	Médio	Média	Moderado
CT-026	Cadastro de produto válido	Alto	Alta	Crítico
CT-027	Cadastro de produto com nome duplicado	Médio	Média	Moderado
CT-028	Cadastro de produto com preço negativo	Médio	Média	Moderado
CT-029	Cadastro de produto com quantidade negativa	Médio	Média	Moderado
CT-030	Cadastro de produto com campos obrigatórios faltando	Baixo	Baixa	Baixo
CT-031	Atualização de produto válido	Médio	Média	Moderado
CT-032	Atualização de produto com nome duplicado	Médio	Média	Moderado
CT-033	Atualização de produto sem autenticação	Médio	Média	Moderado
CT-034	Atualização de produto com campos obrigatórios faltando	Médio	Média	Moderado
CT-035	Atualização de produto em ID inexistente	Baixo	Baixa	Baixo
CT-036	Listagem de produtos válida	Médio	Média	Moderado
CT-037	Listagem sem autenticação	Médio	Média	Moderado
CT-038	Listagem com ID inválido	Médio	Média	Moderado

CT-039	Exclusão de produto válido	Médio	Média	Moderado
CT-040	Exclusão de produto vinculado a um carrinho	Alto	Alta	Crítico
CT-041	Exclusão de produto sem autenticação	Baixo	Baixa	Baixo
CT-042	Exclusão de produto com ID inválido	Baixo	Baixa	Baixo
CT-043	Criação de carrinho válido	Alto	Alta	Crítico
CT-044	Criação de carrinho sem autenticação	Médio	Média	Moderado
CT-045	Criação de carrinho com produto inexistente	Médio	Média	Moderado
CT-046	Criação de carrinho com produto duplicado	Médio	Média	Moderado
CT-047	Visualização de carrinho válido	Médio	Média	Moderado
CT-048	Visualização de carrinho inexistente	Médio	Média	Moderado
CT-049	Visualização de carrinho sem autenticação	Médio	Média	Moderado
CT-050	Finalização de compra com carrinho válido	Alto	Alta	Crítico
CT-051	Finalização de compra com carrinho inexistente	Médio	Média	Moderado
CT-052	Finalização de compra sem autenticação	Médio	Média	Moderado
CT-053	Finalização de compra e validação de esvaziamento	Alto	Alta	Crítico
CT-054	Cancelamento de compra com carrinho válido	Alto	Alto	Crítico
CT-055	Cancelamento de compra com carrinho inexistente	Media	Media	Moderado

CT-056	Cancelamento de compra sem autenticação	Alta	Alta	Critico
--------	---	------	------	---------

## 12. Priorização da execução dos cenários de teste



Prioridade	Casos de Teste	Justificativa
● <b>Alta</b>	CT-001, CT-005, CT-018, CT-019, CT-020, CT-026, CT-040, CT-043, CT-050, CT-053,CT-054,CT-056	Fluxos críticos para o funcionamento da aplicação e conclusão de compra. (Login, Cadastro, Finalização de Compra).
● <b>Média</b>	CT-002, CT-004, CT-006, CT-007, CT-008, CT-009, CT-010, CT-013, CT-014, CT-016, CT-017, CT-025, CT-027, CT-028, CT-029, CT-031,	Fluxos intermediários que impactam a experiência, mas não bloqueiam o objetivo principal. ((Listagem de produtos, Atualização de Usuários).)
	CT-032, CT-033, CT-034, CT-036, CT-037, CT-038, CT-039, CT-044, CT-045, CT-046, CT-047, CT-048, CT-049, CT-050, CT-051, CT-052,CT-055	
● <b>Baixa</b>	CT-003, CT-011, CT-012, CT-015, CT-021, CT-022, CT-023, CT-024, CT-030, CT-035, CT-041, CT-042	Fluxos não críticos que possuem impacto isolado e não impedem o uso principal da aplicação.

## 13. Cobertura de Testes

Cobertura (%) = (Caso de testes mapeados : / Casos de teste planejado:) × 100

**Total de Casos de planejado: 56**

**Casos de Teste mapeado: 56**

✅ Cobertura de Testes: 100%

## 14. Testes Candidatos à Automação- Fluxo completo

garantir a consistência e a validação contínua de todas as etapas principais do processo, desde o cadastro de usuário até o cancelamento de um carrinho de compras

ID	Descrição	Justificativa para Automação
CT001	Cadastro de usuário válido	Processo repetitivo e fundamental para o fluxo.
CT018	Login com credenciais válidas	Necessário para autenticação em outras operações.



CT-026	Cadastro de produto válido	Base para operações de venda e manipulação de estoque.
CT-043	Criação de carrinho válido	Verifica a adição de produtos de forma segura
CT-047	Visualização de carrinho válido	Valida a listagem dos itens no carrinho
CT-050	Finalização de compra com sucesso	Garante que o processo de funciona corretamente.
CT-054	Cancelamento de compra com sucesso	Assegura que o carrinho é removido após o cancelamento.