Plano de Testes – API ServeRest

Projeto: https://compassuol.serverest.dev/
Versao: 1.0

Autor: @Thais Nogueira

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

1. Introduçao

A aplicaçao 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

Incluido:

- Usuários: cadastro, login,atualizaçao e remoçao
- Login: autenticaçao
- Produtos: Cadastro, listagem, atualizaçao, remoçao
- Carrinhos de compras: testes exploratorios

Fora do escopo:

- Testes de performance
- Integrações externas

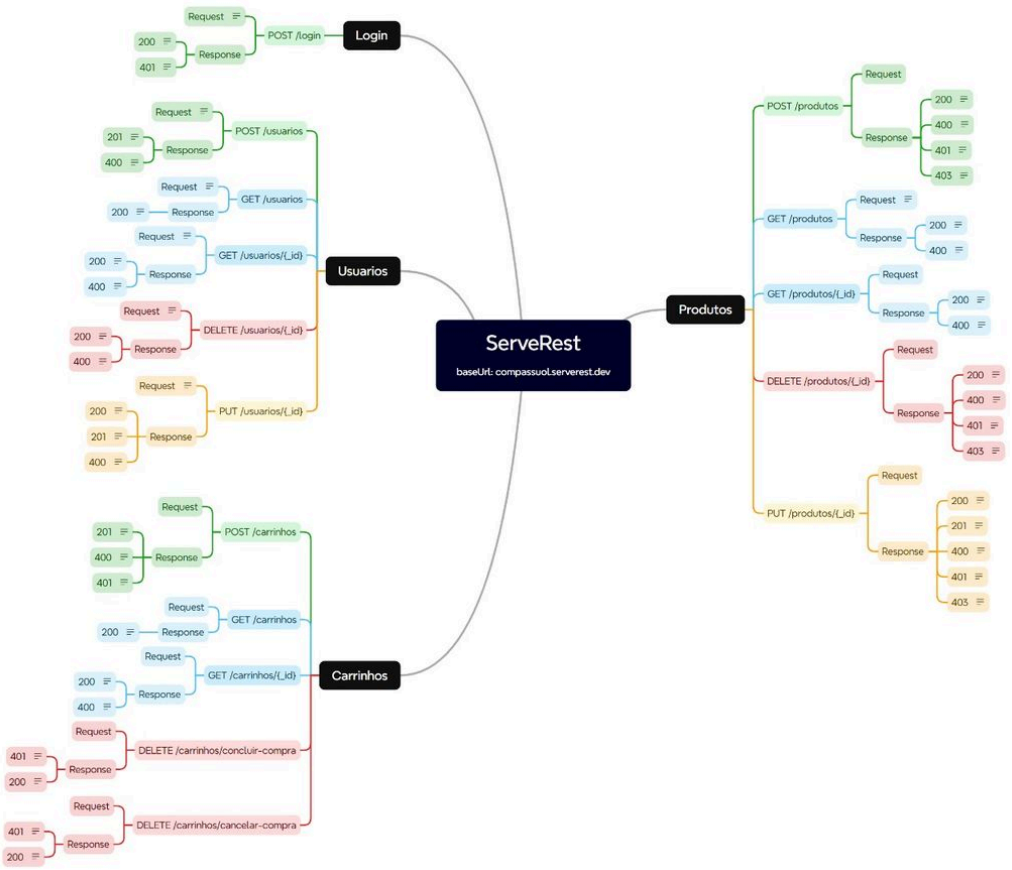
4. Referências

- 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:** Postman, 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. Ánalise 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 malformado.→ 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.Cenario de teste planejado vs Executados

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	Executado	Resultado Obtido	

CT-001 🔴	Cadastro de usuário válido	API disponível e ambiente configurado	1. Enviar requisição para criar um usuário válido.
			2. Validar retorno de sucesso e dados criados.
				{ "nome": "Ana Silva", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	201
					{
					"message": "Cadastro realizado com sucesso",
					"_id": ""
					}
						Alta	✅	status code 201 Created
								
								{
								
								"message": "Cadastro realizado com sucesso",
								"_id": "qTQ4ick6SPLCGer5"
								}	

CT-002	Cadastro com e-mail já existente	Usuário já cadastrado no sistema	1. Enviar requisição para criar um usuário com e-mail duplicado.
			2. Validar retorno de erro.
				{ "nome": "João Silva", "email": "Anateste@teste.com.brm", "password": "teste", "administrador": "true" }	Erro: "E-mail já cadastrado. "	Alta	✅	status code 400Bad Request
								
								{
								
								"message": "Este email já está sendo usado"
								}	

CT-003	Cadastro com e-mail inválido	API disponível e ambiente configurado	1. Enviar requisição para criar um usuário com e-mail inválido.
			2. Validar retorno de erro.
				{ "nome": "João Silva", "email": "joaoteste.com.br", "password": "teste", "administrador": "true" }	Status code 400 Bad Request
					{
					"email": "email deve ser um email válido"
					}
						Baixa	✅	Status code 400 Bad Request
								{
								"email": "email deve ser um email válido"
								}
									A mensagem da response nao esta escrita no documento do Swegger
									Abrir Issues de melhoria
									SER-6: CT-003 POST/usuarios - Mensagem de erro para e-mail inválido não documentada no Swagger TAREFAS PENDENTES

CT-004	Cadastro com domínio bloqueado (Gmail, Hotmail)	API disponível e ambiente configurado	1. Enviar requisição para criar um usuário com domínio bloqueado. 2. Validar retorno de erro.	{ "nome": "João Silva", "email": "joao@gmail.com",
				"password": "teste", "administrador": "true" }
					400 (Bad Request) "Domínio de e-mail não permitido."	Media	✅	Gmail e hotmail: Status code: 201 Created
								{
								"message": "Cadastro realizado com sucesso",
								"_id": "t7ZdfHMIhQ28pyFy"
								}
									Nao deveria ser permitido criar o cadastro com email gmail e hotmail
									Abril Issues
									SER-5: CT-004 POST/usuarios -Cadastro permitido com domínio de e-mail bloqueado (Gmail, Hotmail) TAREFAS PENDENTES

CT-005 🔴	Cadastro sem campos obrigatórios	API disponível e ambiente configurado	1. Enviar requisição sem os campos obrigatórios. 2. Validar retorno de erro.	{ "nome": "", "email": "", "password": "", "administrador": "" }	400 (Bad Request) "Campos obrigatórios não informados."	Alta	✅	Status code 400 (Bad Request) { "nome": "nome não pode ficar em branco", "email": "email não pode ficar em branco", "password": "password	A mensagem da response nao esta escrita no documento do Swegger Abrir Issues de melhoria SER-7: CT-005 POST/usuarios - Mensagens de erro para campos obrigatórios não documentadas no Swagger TAREFAS PENDENTES

								não pode ficar em branco", "administrador": "administrador deve ser 'true' ou 'false'" }	
CT-006	Cadastro com senha fora do limite permitido	API disponível e ambiente configurado	1. Enviar requisição com senha menor que 5 ou maior que 10 caracteres. 2. Validar retorno de erro.	{ "nome": "Maria", "email": "maria@teste.com", "password": "123", "administrador": "true" }	Erro: "A senha deve ter entre 5 e 10 caracteres."	Média	✅	Status code 201 Created { "message": "Cadastro realizado com sucesso", "_id": "PMfde6K2YzKxkJE2" }	Nao deveria ser permitido criar cadastro com senha menor ou maior que 10 Abrir issues SER-8: CT-006- PO ST/usuarios - Cadastro permitido com senha fora do limite permitido (menor que 5 ou maior que 10 caracteres) TAREFAS PENDENTES

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	Executado	Resultado Obtido

CT-007	Alterar usuário válido	Usuário cadastrado no sistema	1. Inserir o Id no put/usuarios/{_id} 2. Enviar parametros para atualizar um usuário válido, alterando o nome 3. Validar retorno de sucesso e dados atualizados.	_id:qTQ4ick6SPLCGer5 { "nome": "Ana", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	Status code 200 OK {"message": "Registro alterado com sucesso"}	Alta	✅	Status code 200 OK {"message": "Registro alterado com sucesso"}
CT-008	Alterar usuário com e-mail duplicado	Outro usuário já existe com o mesmo e-mail	1. Enviar requisição para atualizar um usuário com e-mail duplicado. 2. Validar retorno de erro.	{ "nome": "Ana Silva", "email": "Anateste@teste.com.br", "password": "teste", "administrador": "true" }	status code 400 bad request Erro: "E-mail já cadastrado."	Alta	Não	-
CT-009	Alterar usuário sem campos obrigatórios		1. Enviar requisição sem os campos obrigatórios. 2. Validar retorno de erro.	{ "nome": "", "email": "", "password": "", "administrador": "" }	status code 400 bad request Erro: "Campos obrigatórios não informados."	Média	Não	-
CT-010	Alterar usuário com e-mail inválido		1. Enviar requisição para atualizar um usuário com e-mail inválido. 2. Validar retorno de erro.	{ "nome": "Ana", "email": "Anateste.com.br", "password": "teste", "administrador": "true" }	status code 400 bad request Erro: "Formato de e-mail inválido."	Média	Não	-
CT-011	Alterar usuário em ID inexistente	ID não encontrado no banco de dados	1. Enviar requisição para atualizar um usuário com ID inexistente. 2. Validar se um novo usuário é criado.	{ "nome": "Novo Usuário", "email": "novo@teste.com", "password": "54321", "administrador": "true" }	Usuário criado com sucesso.	Baixa	Não	-

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	Executado	Resultado Obtido	
CT-012	Listagem de usuários válida	Usuários cadastrados no sistema	1. Enviar requisição para listar usuários. 2. Validar retorno de sucesso e dados carregados.	-	Lista de usuários cadastrados.	Baixa	Não	-	
CT-013 🔴	Listagem de usuários sem autenticação	-	1. Enviar requisição para listar usuários sem autenticação. 2. Validar retorno de erro.	-	Erro: "Não autorizado."	Media	✅	status code 200 OK retorno de sucesso e dados carregados.	Nao deveria gerar a lista de usuarios sem autenticaçao por questoes de segurança Abrir um Issues SER-9: CT-013 - GET/Usuarios - Listagem de usuários sem autenticação permitida pela API TAREFAS PENDENTES
CT-014	Listagem de usuários com ID inválido	-	1. Enviar requisição para listar um usuário com um ID inválido 2. Validar retorno de erro.	Em porametros inserir Key : _id Value: 1234	Lista de usuario vazia	Média	Não	-	

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	Executado	Resultado Obtido
CT-015	Exclusão de usuário válido	Usuário existente e autenticado	1. Enviar requisição para excluir o usuário. 2. Validar retorno de sucesso.	ID: UQEPVoJ0rG5Jgj4E	Usuário removido do sistema.	Baixa	✅	Status code 200 OK {"message": "Registro excluído com sucesso" }
CT-016	Exclusão de usuário inexistente	-	1. Enviar requisição para excluir um usuário com ID inexistente 2. Validar retorno de erro.	-	Erro: "Usuário não encontrado."	Média	Não	-
CT-017	Exclusão de usuário sem autenticação	API disponível e ambiente de teste configurado	1. Enviar requisição para excluir um usuário sem autenticação. 2. Validar retorno de erro.	-	Erro: "Não autorizado."	Media	Não	-

Login US002

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	Executado	Resultado Obtido	
CT-018 🔴	Login com credenciais válidas	Usuário existente no sistema	1. Enviar uma requisição POST /login. 2. Preencher o payload com as credenciais: E-mail e senha	{ "email": "Anateste@teste.com.br", "password": "teste" }	status code 200 {"message": "Login realizado com sucesso", "authorization": "Bearer }	Alta	✅	Status code 200 {"message": "Login realizado com sucesso", "authorization": "Bearer ..."}	
CT-019 🔴	Login com e-mail não cadastrado	Nenhum usuário cadastrado no sistema	1. Enviar requisição para realizar login com um e-mail não existente. 2. Validar retorno de erro.	{ "email": "naoexiste@teste.com", "password": "teste" }	status code 401 Erro: "Credenciais inválidas."	Alta	✅	status code 401 "Email e/ou senha inválidos"	
CT-020 🔴	Login com senha incorreta	Usuário existente no sistema	1. Enviar requisição para realizar login com senha incorreta 2. Validar retorno de erro.	{"email": "Anateste@teste.com.br", "password": "senhaincorreta" }	status code 401 Erro: "Credenciais inválidas."	Alta	✅	status code 401 {"message": "Email e/ou senha inválidos" }	
CT-021	Login com campos vazios	API disponível e ambiente de	1. Enviar requisição para realizar login sem preencher os	{ "email": "", "password": "" }	Erro: "Campos obrigatórios não	Baixa	✅	-Status code 400Bad	A mensagem da response nao esta escrita no documento do Swegger Abrir Issues de melhoria

		teste configurado	campos obrigatórios. 2. Validar retorno de erro.		preenchidos."			Request {"email": "email não pode ficar em branco", "password": "password não pode ficar em branco"}	SER-11: CT-021- POST /login - Mensagens de erro para login com campos vazios não documentadas no Swagger TAREFAS PENDENTES
CT-022	Login com e-mail mal formatado		1. Enviar requisição para realizar login com e-mail sem "@". 2. Validar retorno de erro.	{ "email": "Anateste.com.br", "password":"teste}	Erro: "Formato de e-mail inválido."	Baixa	✅	Status code 400Bad Request {"email": "email deve ser um email válido"}	A mensagem da response nao esta escrita no documento do Swegger Abrir Issues de melhoria SER-12: CT-022- POST/login-Mensagens de erro para login com e-mail mal formatado não documentadas no Swagger TAREFAS PENDENTES
CT-023	Login com payload vazio		1. Enviar requisição para realizar login sem payload. 2. Validar retorno de erro.	{ }	"Campos obrigatórios não preenchidos."	Baixa	Não	-	
CT-024	Login com estrutura de payload incorreta		1. Enviar requisição para realizar login com payload faltando atributos (ex.: sem email) 2. Validar erro.	{ "password": "12345" }	Erro: "Campo 'email' não informado."	Baixa	Não	-	

CT-025	Expiração de token após 10 minutos	Token gerado no login	1. Realizar login para gerar o token. 2. Esperar 10 minutos. 3. Tentar acessar um recurso autenticado.	{ "Authorization": "Bearer <token_gerado> " }	Erro: "Token expirado."	Alta	Nao		

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,descriçao 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	Executado	Resultado Obtido	
CT-026 🔴	Cadastro de produto válido	Usuário autenticado no sistema	1. Realizar login para gerar o token. 2. Enviar requisição para criar um produto válido. 3. Validar retorno de sucesso e dados criados.	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": 10 }	"Cadastro realizado com sucesso"	Alta	✅	Status code 201 {"message": "Cadastro realizado com sucesso", "_id": "jcAMnCpGiqEFAE7S" }	
CT-027	Cadastro de produto com nome	Produto já existente com o mesmo nome	1. Enviar requisição para criar um produto com nome duplicado. 2.	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao":	Erro: "Produto já cadastrado."	Alta	✅	Status code Bad request {"message": "Já existe	

	duplicado		Validar retorno de erro.	"Smartphone Samsung Galaxy", "quantidade": 10 }				produto com esse nome"}	
CT-028	Cadastro de produto com preço negativo	API disponível e ambiente de teste configurado	1. Enviar requisição para criar um produto com valor negativo. 2. Validar retorno de erro.	{ "nome": "Celular Samsung", "preco": -1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": 10 }	Erro: "Preço inválido."	Média	✅	status code 400 bad request {"preco": "preco deve ser um número positivo"}	A mensagem da response nao esta escrita no documento do Swegger Abrir Issues de melhoria 🐞 SER-13: CT-028- POST/produto- Mensagem de erro para preço negativo não documentada no Swagger TAREFAS PENDENTES
CT-029	Cadastro de produto com quantidade negativa	API disponível e ambiente de teste configurado	1. Enviar requisição para criar um produto com quantidade negativa. 2. Validar retorno de erro.	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy", "quantidade": -10 }	Erro: "Quantidade inválida."	Média	Não	-	
CT-030	Cadastro de produto com campos obrigatórios faltando	API disponível e ambiente de teste configurado	1. Enviar requisição para criar um produto sem os campos obrigatórios. 2. Validar retorno de erro.	{ "nome": "", "preco": "", "descricao": "", "quantidade": "" }	Erro: "Campos obrigatórios não informados."	Baixa	Não	-	

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, descriçao,quantdade

ID	Caso de Teste	Pré-condição	Passos	Payload	Resultado Esperado	Prioridade	Executado	Resultado Obtido
CT-031	Atualização de produto válido	Produto existente no sistema	1. Enviar requisição para atualizar um produto válido. 2. Validar retorno de sucesso e dados atualizados.	{ "nome": "Celular Samsung Atualizado", "preco": 1500.00, "descricao": "Smartphone Samsung Galaxy Atualizado", "quantidade": 10 }	JSON contendo: `nome`, `preco`, `descricao`, `quantidade`	Alta	Não	-
CT-032	Atualização de produto com nome duplicado	Produto existente com o mesmo nome	1. Enviar requisição para atualizar um produto com nome duplicado. 2. Validar retorno de erro.	{ "nome": "Celular Samsung", "preco": 1500.00, "descricao": "Smartphone Galaxy", "quantidade": 10 }	Erro: "Nome do produto já cadastrado."	Alta	Não	-
CT-033	Atualização de produto sem autenticação	API disponível e ambiente de teste configurado	1. Enviar requisição para atualizar um produto sem token JWT. 2. Validar retorno de erro.	{ "nome": "Celular Samsung", "preco": 1800.00, "descricao": "Smartphone Galaxy", "quantidade": 10 }	Erro: "Não autorizado."	Media	Não	-

CT-034	Atualização de produto com campos obrigatórios faltando	Produto existente no sistema	1. Enviar requisição para atualizar um produto sem os campos obrigatórios. 2. Validar retorno de erro.	{ "nome": "", "preco": "", "descricao": "", "quantidade": "" }	Erro: "Campos obrigatórios não informados. "	Média	Não	-
CT-035	Atualização de produto em ID inexistente	ID não existe no banco de dados	1. Enviar requisição para atualizar um produto com um ID inexistente. 2. Validar se um novo produto é criado.	/_id: Wbg12g45gd455we { "nome": "Celular Inexistente", "preco": 2000.00, "descricao": "Novo Produto", "quantidade": 5 }	Produto criado com sucesso.	Baixa	✅	Status code 201 Created {"message": "Cadastro realizado com sucesso", "_id": "6lEdZlft4ohwYbvn"}

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	Executado	Resultado Obtido	
CT-036	Listagem de	Produtos cadastrad	1. Enviar requisi	-	Lista de produtos	Alta	✅	status code	

	produtos válida	os e usuário autenticado	ção para listar produtos 2. Validar retorno de sucesso e dados carregados.		gerados			200OK Lista de produto gerado	
CT-037	Listagem sem autenticação	Sem autenticaçao	1. Enviar requisição para listar produtos sem token JWT. 2. Validar retorno de erro.	-	Erro: "Não autorizado."	Media	✅	status code 200OK Lista de produto gerado	Nao deveria ser permitido gerar lista de produtos sem autenticaçao Abrir Issues SER-14: CT-037- GET/produtos- Listagem de produtos permitida sem autenticação TAREFAS PENDENTES
CT-038	Listagem com ID inválido		1. Enviar requisição para listar um produto com um ID inválido 2. Validar retorno de erro.	Em parametro Key: _id value:123	Lista de usuario vazia	Média	Não	-	

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	Executado	Resultado Obtido
CT-039	Exclusão de produto válido	Produto existente e não associado a carrinho Com autenticaçao	1. Enviar requisição para excluir o produto. 2. Validar retorno de sucesso.	_id: jcAMnCpGiqEFAE7S	Produto removido do sistema.	Alta	✅	status code 200 OK {"message": "Registro excluído com sucesso"}
CT-040 🔴	Exclusão de produto vinculado a um carrinho	Produto associado a um carrinho	1. Enviar requisição para excluir um produto que está em um carrinho. 2. Validar retorno de erro.	Id: lJRn5q3HI8djD3gTlJRn5q3HI8djD3gT	Erro: "Produto vinculado a um carrinho. Não pode ser excluído."	Alta	Não	status code 400 Bad Request {"message": "Não é permitido excluir produto que faz parte de carrinho", "idCarrinhos": ["Qwieyo2PAozlXS7Y"] }
CT-041	Exclusão de produto sem autenticaçã o		1. Enviar requisição para excluir um produto sem token JWT. 2. Validar	-id: lJRn5q3HI8djD3gT	Erro: "Não autorizado. "	Baixa	✅	status code 401 Unauthorized {"message": "Token de acesso ausente, inválido, expirado ou

			retorno de erro.					usuário do token não existe mais" }
CT-042	Exclusão de produto com ID inválido		1. Enviar requisição para excluir um produto com ID inválido. 2. Validar retorno de erro.	id:Ss1234fhg158dg15	"Produto não encontrado."	Baixa	✅	status code 200 OK {"message": "Nenhum registro excluído"}

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	Executado	Resultado Obtido
CT-043 🔴	Criação de carrinho válido	Usuário autenticado e produto existente	1. Enviar requisição para criar um carrinho com produto válido. 2. Validar retorno	{ "produtos": [{ "idProduto": "lJRn5q3HI8djD3gT", "quantidade": 1	produto adicionado ao carrinho	Alta	✅	status code 201 created {"message": "Cadastro realizado com sucesso", "_id": "Qwieyo2PAozlXS7Y"}

			de sucesso.	}] }				
CT-044	Criação de carrinho sem autenticação		1. Enviar requisição para criar um carrinho sem token JWT. 2. Validar retorno de erro.	{ "produtos": [{ "idProduto": "lJRn5q3HI8djD3gT", "quantidade": 1 }] }	Erro: "Não autorizado."	Media	Não	-
CT-045	Criação de carrinho com produto inexistente	Produto não está cadastrado no sistema	1. Enviar requisição para criar um carrinho com produto inexistente. 2. Validar retorno de erro.	{ "produtos": [{ "idProduto": "produtoInvalido", "quantidade": 2 }] }	Erro: "Produto não encontrado."	Media	Não	-
CT-046	Criação de carrinho com produto duplicado	Produto duplicado no mesmo carrinho	1. Enviar requisição para criar um carrinho com produtos duplicados. 2. Validar retorno de erro.	{ "produtos": [{ "idProduto": "umproduto", "quantidade": 2 }, { "idProduto": "umproduto", "quantidade": 2 }] }	Erro: "Produto duplicado no carrinho."	Média	Não	-

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	Executado	Resultado Obtido
CT-047	Visualização de carrinho válido	Carrinho criado previamente	1. Enviar requisição para visualizar o carrinho. 2. Validar retorno de sucesso e produtos exibidos.	"_id": "Qwieyo2PAozlXS7Y"	visualizar carrinho	Alta	✅	status code 200 OK Carrinho encontrado e exibido
CT-048	Visualização de carrinho inexistente		1. Enviar requisição para visualizar um carrinho inexistente. 2. Validar retorno de erro.	ID: Qwieyo2PAozlXS48	Erro: "Carrinho não encontrado."	Média	✅	Status code 400 Bad Request {"message": "Carrinho não encontrado" }
CT-049	Visualização de carrinho sem autenticação	API disponível e ambiente de teste configurado	1. Enviar requisição para visualizar um carrinho sem token JWT. 2.	-	Erro: "Não autorizado."	Média	Não	-

			Validar retorno de erro.					

Cenário de Teste 03- Finalizaçao 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	Executado	Resultado Obtido
CT-050 🔴	Finalização de compra com carrinho válido	Carrinho criado previamente e autenticado	1. Enviar requisição para finalizar a compra. 2. Validar retorno de sucesso.		"Carrinho excluído com sucesso"	Alta	✅	status code 200OK {"message": "Registro excluído com sucesso"}
CT-051	Finalização de compra com carrinho inexistente		1. Enviar requisição para finalizar a compra de um carrinho inexistente. 2. Validar retorno de erro.	-	Erro: "Carrinho não encontrado."	Média	✅	Status code 200 OK {"message": "Não foi encontrado carrinho para esse usuário"}
CT-052	Finalização de compra sem autenticaçã o		1. Enviar requisição para finalizar a compra sem token JWT. 2. Validar retorno de erro.	-	Erro: "Não autorizado."	Média	✅	status code 401 Unauthorized {"message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"

								}
CT-053 🔴	Finalização de compra e validação de esvaziamento	Compra finalizada previamente	1. Finalizar compra. 2. Consultar o carrinho após finalização. 3. 3. Validar se está vazio.	-	Retorno de carrinho vazio após finalizar compra.	Alta	✅	Status code:400 bad request { "message": "Carrinho não encontrado" }

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	Executado	Resultado Obtido	
CT-054 🔴	Cancelamento de compra com carrinho válido	Carrinho criado previamente e autenticado	1. Enviar requisição para cancelar a compra. 2. Validar retorno de sucesso.		"Compra cancelada com sucesso e produtos retornados ao estoque"	Alta	✅	Status code 200OK {"message": "Registro excluído com sucesso. Estoque dos produtos reabastecido" }	A mensagem da response nao esta escrita no documento do Swegger Abrir Issues de melhoria SER-16: CT-054- DELETE/carrinho- Mensagem de cancelamento de compra divergente do Swagger3 TAREFAS PENDENTES
CT-055	Cancelamento		1. Enviar requisição para		Erro: "Carrinho não encontrado."	Média	✅	Status code	-

	de compra com carrinho inexistente		cancelar a compra de um carrinho inexistente. 2. Validar retorno de erro.				200OK {"message": "Não foi encontrado carrinho para esse usuário" }	
CT-056 🔴	Cancelamento de compra sem autenticação		1. Enviar requisição para cancelar a compra sem token JWT. 2. Validar retorno de erro.		Erro: "Não autorizado."	Alta	✅	Status code 401 Unauthorized {"message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais" }	-

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-050a CT-053	Sim
Carrinho - Cancelamento de Compra	CT-054 a CT-056	Sim

10. 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	Critico
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
🔴 **Alta**	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).
🟠 **Média**	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	
🟢 **Baixa**	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. Resumo da execucao

Funcionalidade	Total de Casos Mapeados	Executados	Sucesso	Falha
Cadastro de Usuários	17	9	4	5
Login	8	5	3	2
Produtos	17	9	7	2

Carrinhos	14	10	9	1
Total Geral	**56**	**33**	**23**	**10**

14. Cobertura de Testes

Cobertura (%) = (N° de testes executados : / Total de itens mapeados:) × 100

Total de Casos de Teste Mapeados:

- Cadastro de Usuários: 14 casos
- Login: 7 casos
- Produtos: 15 casos
- Carrinhos: 14 casos
- Finalização de Compra: 5 casos
  Total Geral: 56 casos

Casos de Teste Executados:

- Cadastro de Usuários: 9 executados
- Login: 5 executados
- Produtos: 9 executados
- Carrinhos: 7 executados
- Finalização de Compra: 3 executados
  Total Executado: 33 casos
- ✅ Cobertura de Testes: 58,92%

15. 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.

ID	Caso de Teste	Pré-condições	Passos	Método	Endpoint	Payload	Resultado Esperado	Resultado Obtido
CT001	Cadastro de Usuário Válido	API disponível e ambiente configurado	1. Enviar requisição para criar um usuário válido.	POST	/usuarios	{ "nome": "Ana da Silva", "email": "{{uniqueEmail}}", "password": "teste", "administrador": "true" }	201 Created - Usuário criado e ID gerado.	✅ Passou
CT018	Login com Credenciais Válidas	Usuário criado e email válido	2. Realizar o login com as credencias válidas.	POST	/login	{ "email": "Anateste@teste.com.br", "password": "teste" }	200 OK - JWT Token gerado.	✅ Passou
CT026	Cadastro de Produto Válido	Usuário logado e token válido	3. Cadastrar um produto no sistema.	POST	/produtos	{ "nome": "{{uniqueProductName}}", "preco": 7000.00, "descricao": "{{uniqueProductDescription}}", "quantidade": 5 }	201 Created - Produto criado e ID gerado.	✅ Passou
CT043	Criação de Carrinho Válido	Produto criado e usuário logado	4. Adicionar o produto ao carrinho.	POST	/carrinhos	{ "produtos": [{ "idProduto": "{{productId}}", "quantidade": 1 }]	201 Created - Carrinho criado e ID gerado.	✅ Passou

						}		
CT047	Visualização de Carrinho	Carrinho criado e usuário autenticado	5. Visualizar o carrinho para validar os itens.	GET	/carrinhos/{{cartId}}	-	200 OK - Listagem de produtos no carrinho.	✅ Passou
CT050	Finalização de Compra	Carrinho com produtos	6. Finalizar a compra.	DELETE	/carrinhos/concluir-compra	-	200 OK - Compra finalizada com sucesso.	✅ Passou
CT054	Cancelamento de Compra	Compra finalizada	7. Cancelar a compra e remover produtos do carrinho.	DELETE	/carrinhos/{{cartId}}	-	200 OK - Carrinho removido com sucesso.	✅ Passou

16. Documentação de Issues e Melhorias

Type	Key	Resumo	Prioridade	Status
🐞	SER-16	CT-054- DELETE/carrinho- Mensagem de cancel...	= Medium	TAREFAS PENDENTES
🐞	SER-14	CT-037- GET/produtos- Listagem de produtos pe...	^ High	TAREFAS PENDENTES
🐞	SER-13	CT-028- POST/produto- Mensagem de erro para ...	= Medium	TAREFAS PENDENTES
🐞	SER-12	CT-022- POST/login-Mensagens de erro para log...	= Medium	TAREFAS PENDENTES
🐞	SER-11	CT-021- POST /login - Mensagens de erro para l...	= Medium	TAREFAS PENDENTES
🐞	SER-9	CT-013 - GET/Usuarios - Listagem de usuários se...	^^ Highest	TAREFAS PENDENTES
🐞	SER-8	CT-006- POST/usuarios - Cadastro permitido co...	^ High	TAREFAS PENDENTES
🐞	SER-7	CT-005 POST/usuarios - Mensagens de erro par...	^ High	TAREFAS PENDENTES
🐞	SER-6	CT-003 POST/usuarios - Mensagem de erro para...	= Medium	TAREFAS PENDENTES
🐞	SER-5	CT-004 POST/usuarios -Cadastro permitido com ...	^ High	TAREFAS PENDENTES

10 itens

Sincronizado agora

Issues e melhorias identificados