ServeRest API - Testes

Squad 3 - LevelUp
Maycon Douglas da Silva

Challenge 03

Objetivo

O objetivo deste plano de testes é garantir a qualidade da API ServeRest, ,validando as regras de negócio e os requisitos das User Stories. A validação será realizada através da execução de testes funcionais, de regressão e de integração, utilizando o Swagger , uma coleção Postman e a documentação das issues, com o foco em identificar bugs e propor melhorias para a aplicação. Com o Robot Framework. A validação será realizada através da execução de testes funcionais, de regressão e de integração, com o objetivo de identificar bugs e propor melhorias para a aplicação.

Escopo dos Testes

Funcionalidades a serem exploradas:

Usuários

Criação, edição, listagem e exclusão de usuários (CRUD)

Validação das regras de negócio, como formatos de e-mail e senha

Cadastro, gerenciamento de usuários, e exclusão

Login

Processo de autenticação

Verificação de cenários de sucesso e falha

Produtos

Criação, edição, listagem e exclusão de produtos (CRUD)

Validação de regras, como nome duplicado

Carrinho de Compras

Análise das operações de adicionar e remover produtos do carrinho

HTTP

Verificação de códigos de status HTTP e conformidade de esquemas JSON

Validação de autenticação por token (Auth)

Ferramentas Utilizadas

- Swagger UI 3.x : Para consultas interativas e exportação do JSON/YAML da API para referência.
- Postman v10.9.0 : Para criação de Collection com todos os endpoints e variáveis de ambiente (baseUrl, apiKey).
- Robot Framework: Para a criação e execução de cenários de teste.
- RequestsLibrary: A principal biblioteca para a execução de requisições HTTP (GET, POST, PUT, DELETE).
- JSONLibrary: Para a manipulação e validação de dados em formato JSON.
- RequestsLibrary:Biblioteca principal para interagir com a API, enviando requisições HTTP (GET, POST, PUT, DELETE) e validando as respostas.
- Collections: Para manipular e validar estruturas de dados, como listas e dicionários.
- String: Para manipular e validar strings.

Metodologia de Teste

Estratégia Utilizada:

CRUD

Criar, Ler, Atualizar, Deletar.

CHIQUE

Campos obrigatórios, Habilitar/Desabilitar formulários, Interrupção da ação

Quebra de fluxos, Usuabilidade dos menus, Estouro de Campos .

Testes Baseados em Especificação

permite validar cada campo e comportamento esperado, detectando inconsistências entre implementação e documentação.

Teste de Segurança

Verificar autenticação, autorização e vulnerabilidades.

Teste Exploratórios

Testar a API de forma livre para encontrar comportamentos inesperados.

Teste de Usabilidade

Avaliar se a API é clara, fácil de entender e usar.

Heurísticas Aplicadas

- Verificação de códigos de erro
- Consistência de esquema JSON (tipos, obrigatoriedade)
- Tratamento de dados inválidos e limites de input
- Autenticação: validade e expiração de tokens

Técnica de Execução

Execução Orientada por Collection no Postman:

Configuração dos ambientes (dev, staging) com variáveis globais.

Importação da Collection gerada a partir do Swagger.

Execução sequencial dos requests em modo Runner, com gravação de logs.

Uso de scripts “Tests” para assertivas de status code.

Execução Orientada por testes com Robot Framework

Execução Orientada por Test Suites

Configuração de Ambientes com Arquivos de Variáveis

Execução via Linha de Comando: robot -d ./results .\{nome arquivo}.robot

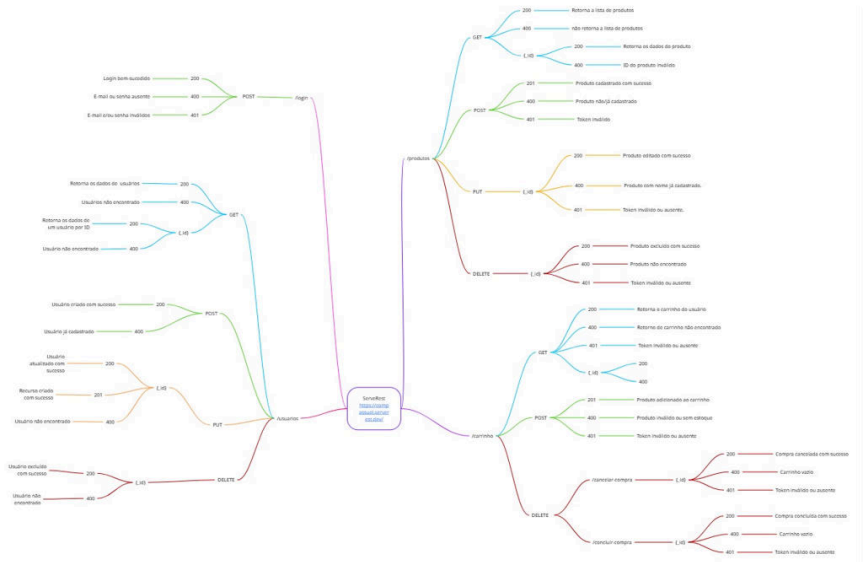
Uso de Keywords de Asserção e Validação

Relatórios e Logs Automáticos

Mapas Mentais

Mapa Front ServeRest https://miro.com/app/board/uXjVJYGmuZA=/

Mapa API ServeRest https://miro.com/app/board/uXjVJVYZNxA=/



Mapa Issues ServeRest https://miro.com/app/board/uXjVJTfSekc=/

Sessões de Teste Planejadas

Sessão	Funcionalidade Foco	Heurísticas aplicadas	Duração Estimada
1	Cadastro e login	CRUD+CHIQUE+MANUAIS+SEGURANÇA+EXPLORATORIOS+USUABILIDADE + PERFOMANCE	1h30min
2	Listagem e edição de usuários	CRUD+CHIQUE+MANUAIS+SEGURANÇA+EXPLORATORIOS+USUABILIDADE + PERFOMANCE	1h30min
4	Geração de relatórios	CRUD+CHIQUE+MANUAIS+SEGURANÇA+EXPLORATORIOS+USUABILIDADE + PERFOMANCE	30 min
5	Testes mobile e acessibilidade	CRUD+CHIQUE+MANUAIS+SEGURANÇA+EXPLORATORIOS+USUABILIDADE + PERFOMANCE	2hrs
6	Avaliação da interface geral (UI)	CRUD+CHIQUE+MANUAIS+SEGURANÇA+EXPLORATORIOS+USUABILIDADE + PERFOMANCE	2hrs

Cenários de Teste Postman (CT)

ID do Cenário	**Módulo**	**Cenário**	**Ação**	**Resultado Esperado**	**Prioridade**	**Status**
CT-01	Usuários	Cadastro de usuário com sucesso	Enviar POST para /usuarios com dados válidos (nome, email, senha).	201 Created. Usuário criado no banco e dados de resposta corretos.	Alta	Executado

CT-02	Usuários	Cadastro com e-mail já existente	Tentar cadastrar um usuário com um e-mail já usado.	400 Bad Request. Mensagem de erro indicando e-mail em uso.	Alta	Executado
CT-03	Usuários	Tentar cadastrar com e-mail inválido	Enviar POST para /usuarios com um e-mail em formato incorreto	400 Bad Request. Mensagem de erro apropriada indicando o formato inválido.	Alta	Executado
CT-04	Usuários	Tentar cadastrar com senha vazia	Enviar POST /usuarios para com campo de senha vazio ou com caracteres especiais não permitidos.	400 Bad Request. Mensagem de erro informando o que a senha é obrigatória.	Média	Executado
CT-05	Usuários	Tentar cadastrar com injeção de script	Enviar POST /usuarios para com codigos de scripts no campo de nome ou e-mail.	400 Bad Request ou erro de validação. A aplicação deve sanitizar a entrada e não permitir a criação do usuário com script.	Média	Executado
CT-06	Usuários	Múltiplos logins com senha incorreta	Tentar fazer login repetidamente com uma senha incorreta para o mesmo usuário.	401 Unauthorized. O sistema deve bloquear ou suspender o usuário.	Alta	Executado

CT-07	Usuários	Login com credenciais válidas	Enviar POST para /login com e-mail e senha corretos.	200 OK. Retorno de um token de autenticação válido.	Alta	Executado
CT-08	Usuários	Login com credenciais inválidas	Enviar POST para /login com senha errada.	401 Unauthorized. Mensagem de erro apropriada.	Média	Executado
CT-09	Usuários	Excluir usuário com carrinho ativo	Enviar DELETE para /usuarios/{_id} de um usuário que possui um carrinho de compras com produtos.	400 Bad Request. Mensagem de erro informando o que o usuário não pode ser excluído devido a um carrinho ativo.	Alta	Não Executado
CT-10	Usuários	Tentar cadastrar com tipos de dados inválidos	Enviar POST para /usuarios com um valor numérico no campo nome ou como administrador	400 Bad Request. Mensagem de erro indicando o tipo de dado incorreto.	Média	Executado
CT-11	Usuários	Tentar cadastrar com campos não existentes	Enviar POST para /usuarios com um campo adicional que não faz parte da API	201 Created. A API deve ignorar o campo adicional e criar o usuário com sucesso.	Baixa	Executado

CT-12	Produto	Tentar cadastrar com campos obrigatórios vazios	Enviar POST para /produtos com campos obrigatórios vazios.	400 Bad Request. Mensagem de erro indicando que o campo é obrigatório.	Alta	Executado
CT-13	Produto	Tentar cadastrar com preço ou quantidade zero.	Enviar POST para /produtos com o campo preco ou quantidade com o valor 0.	201 Created. A API deve aceitar os valores 0. porém depende da equipe de regra de negocios	Média	Executado
CT-14	Produto	Cadastro de produto sem autenticação	Enviar POST para /produtos sem o token de autenticação.	401 Unauthorized. Mensagem de erro de autenticação.	Alta	Executado
CT-15	Produto	Cadastro de produto com sucesso	Enviar POST para /produtos com token e dados válidos.	201 Created. Produto criado e dados de resposta corretos.	Alta	Executado
CT-16	Produto	Tentar cadastrar com nome de produto duplicado	Enviar POST /produtos para com um nome de produto que já existe, mas com letras maiúsculas e minúsculas diferentes	400 Bad Request. O sistema deve retornar uma mensagem de erro indicando que o produto já existe.	Alta	Executado

CT-17	Produto	Tentar editar um produto com ID inexistente	Enviar PUT para /produtos/{_id} com um ID de produto que não existe.	404 Not Found. Mensagem de erro informando que o produto não foi encontrado.	Média	Executado
CT-17	Produto	Tentar cadastrar com preço ou quantidade negativos	Enviar POST para /produtos com o campo preco ou quantidade com o valor negativo.	400 Bad Request. O sistema deve retornar uma mensagem de erro indicando que o valores não podem ser negativos.	Alta	Executado
CT-19	Carrinho	Adicionar produto que não existe	Enviar POST para /carrinhos com um ID de produto que não existe no catálogo. (funcionalidade que ainda precisa ser implementada).	400 Bad Request. Mensagem de erro informando que o produto não foi encontrado.	Média	Não Executado
CT-20	Carrinho	Adicionar produto com estoque insuficiente	Enviar POST para /carrinhos solicitando quantidade maior que o estoque. (funcionalidade que ainda precisa ser	400 Bad Request. Mensagem de erro de estoque.	Média	Não Executado

			implementada).			
CT-21	Carrinho	Finalizar compra com sucesso	Enviar DELETE para /carrinhos/concluir-compra com token válido. (funcionalidade que ainda precisa ser implementada).	200 OK. Estoque do produto reduzido, e carrinho removido.	Alta	Não Executado
CT-22	Carrinho	Finalizar compra com carrinho vazio	Enviar DELETE para /carrinhos/concluir-compra quando o usuário não tiver produtos no carrinho. (funcionalidade que ainda precisa ser implementada).	400 Bad Request. Mensagem de erro informando o que o carrinho está vazio.	Média	Não Executado
CT-23	Carrinho	Finalizar compra com produto removido	Tentar finalizar uma compra com um produto que foi removido do catálogo após ter sido adicionado ao carrinho. (funcionalidade que ainda precisa ser	400 Bad Request. Mensagem de erro informando o que o produto não está mais disponível.		Não Executado

			implementada).			
CT-24	Carrinho	Remover produto individual do carrinho	Tentar remover um único produto do carrinho (funcionalidade que ainda precisa ser implementada).	A implementação de um endpoint DELETE para remover itens individuais deve retornar 200 OK e a mensagem de sucesso.	Alta	Não Executado
CT-25	Carrinho	Cancelar compra com sucesso	Enviar DELETE para /carrinhos/cancelar-compra para esvaziar o carrinho (funcionalidade que ainda precisa ser implementada).	200 OK. Mensagem de sucesso e o estoque dos produtos retornam ao estado anterior.	Alta	Não Executado
CT-26	Carrinho	Finalizar compra com carrinho vazio	Enviar DELETE para /carrinhos/concluir-compra quando o carrinho do usuário estiver vazio.	400 Bad Request. Mensagem de erro indicando que o carrinho está vazio.	Média	Não Executado
CT-27	Segurança	Acesso a rota protegida sem token	Tentar acessar GET /carrinhos	401 Unauthorized. Mensagem de erro	Alta	Executado

			sem enviar um token.	de autenticação.		

Matriz de Risco Postman

Cenário	Probabilidade	Impacto	Risco	Prioridade de Execução
Cadastro de usuário com sucesso	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
Cadastro com e-mail já existente	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
Tentar cadastrar com e-mail inválido	Alta (3)	Média (2)	4 (Alta)	2 - Alta
Tentar cadastrar com senha vazia.	Média (2)	Média (2)	4 (Alta)	2 - Alta
Tentar cadastrar com injeção de script	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
Múltiplos logins com senha incorreta	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Excluir usuário com carrinho ativo	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
Login de usuário com credenciais válidas	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Login de usuário com credenciais inválidas	Média (2)	Alto (3)	6 (Crítico)	2 - Alta
Tentar cadastrar produto sem autenticação	Alta (3)	Média (2)	6 (Crítico)	2 - Alta

Tentar cadastrar com preço ou quantidade negativos	Alta (3)	Média (2)	6 (Crítico)	2 - Alta
Tentar editar produto com ID inexistente	Média (2)	Baixo (1)	2 (Baixa)	3 - Média
Finalizar compra com carrinho vazio	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Finalizar compra com produto removido	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Cancelar compra com sucesso	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
Remover produto individual do carrinho	Média (2)	Média (2)	4 (Alta)	2 - Alta
Tentar cadastrar usuario com campos obrigatórios vazios	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Cadastro de produto com sucesso	Média (2)	Média (2)	4 (Alta)	2 - Alta
Tentar cadastrar com tipos de dados inválidos	Média (2)	Alto (3)	6 (Crítico)	2 - Alta
Tentar cadastrar com campos não existentes	Baixa (1)	Baixo (1)	1 (Baixa)	4 - Baixa
Adicionar produto com	Média (2)	Média (2)	4 (Alta)	2 - Alta

estoque insuficiente				
Finalizar compra com sucesso	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Tentar cadastrar produto com campos obrigatórios vazio	Alta (3)	Alta (3)	9 (Crítico)	1 - Crítica
Tentar cadastrar com preço ou quantidade zero	Média (2)	Média (2)	4 (Alta)	2 - Alta
Finalizar compra com carrinho vazio	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
Acesso a rota protegida sem token	Alta (3)	Baixo (1)	3 (Média/Alta)	3 - Média
Excluir usuário	Média (2)	Alto (3)	3 (Média/Alta)	3 - Média

Cobertura de teste Manuais/Postman

ID do Cenário	**Cenários**	**Cobertura do Requisito**
Usuários	CT-01, CT-02, CT-03, CT-04, CT-09,CT-10,CT-11	Cobertura Completa: Testes do cadastro, login, busca de usuários e a verificação de autenticação (erro e sucesso)
Produtos	CT-12,CT-13,CT-14,CT-15,CT-16,CT-17	Cobertura Parcial: Testes do cadastro de produtos com e sem autenticação. Cenários de edição, exclusão e busca por um produto

		específico ainda precisam ser planejados para uma cobertura completa.
Carrinhos	CT-18, CT-19,CT-20,CT-21,CT-22,CT-23,CT-24,CT-25	Cobertura Parcial: Testes da adição de produtos e a finalização da compra. Cenários para cancelamento de compra, visualização do carrinho e validação do preço total ainda não foram mapeados, sendo pontos a serem adicionados para uma cobertura mais abrangente.
Segurança	CT-26 (e outros cenários negativos)	Cobertura Básica: Verificação do acesso não autorizado. Testes mais avançados de segurança, como injeção de dados, não estão no escopo inicial, mas podem ser considerados em um planejamento futuro.

Mapeamento de Endpoints e Casos de Teste feito no Postman

- **ENDPOINTS /usarios**

Endpoint	**Método**	**Descrição**	**Retorno do Teste**	**Resultado Esperado**
/login	POST	Realiza login do usuário	`{ "message": "Login realizado com sucesso", "authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InRlc3RlQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiMTIzIiwiaWF0IjoxNzU1MjAxMTAyLCJleHAi`	200 OK, JSON , usuario cadastrado

			`OjE3NTUyMDE3MDJ9.zTln6xdLtvfaAI2Mr1YDpVQ8406CYaY7U7ilGGdHuZI"` `}`	
/login invalido				

/usuarios	GET	Lista usuários	`"quantidade": 29,`
			`"usuarios": [`
			`{`
			`"nome": "Fulano da Silva",`
			`"email": "fulano@qa.com",`
			`"password": "teste",`
			`"administrador": "true",`
			`"_id": "0uxuPY0cbmQhpEz1"`
			`},`
				200 OK, JSON com dados gravados

/usuarios/{{_id}}	GET	Busca usuários por ID	`{`
			`"nome": "Fulano da Silva",`
			`"email": "beltrano@atualizar.com.br",`
			`"password": "teste",`
			`"administrador": "true",`
			`"_id": "226QcHYfalO8xAUI"`
			`}`
				200 OK, retornando consulta todos os usuarios por ID

/usuarios	POST	Cadastra Usuário	`{ "message": "Cadastro realizado com sucesso", "_id": "PngWyNRjVl6gCXtI" }`	200 OK, JSON, cadastrando o usuario
/usuarios/{{_id}}	DELETE	Deleta usuário por ID	`{ "message": "Registro excluído com sucesso" }`	200 ok, JSOn retornando DELETE
/usuarios/{{_id}}	PUT	Edita usuário por ID	`{ "message": "Registro alterado com sucesso" }`	200 ok, atualizado com sucesso

- **ENDPOINTS /produtos**

Endpoint	**Método**	**Descrição**	**Retorno do Teste**	**Resultado Esperado**
/produtos	POST	Cadastra produto	{ "message": "Cadastro realizado com sucesso", "_id": "vhXs8O7rJgkO0Ex4" }	201 OK, JSON , produto cadastrado
/produtos/{{_id}}	GET	Busca produto por id	{ "nome": "Logitech MX Vertical", "preco": 470, "descricao": "Mouse", "quantidade": 276, "_id": "BeeJh5lz3k6kSIzA" }	200 OK, JSON Busca por dados gravados

/produtos	GET	Lista produtos cadastrados	{ "quantidade": 9, "produtos": [{ "nome": "Logitech MX Vertical", "preco": 470, "descricao": "Mouse", "quantidade": 276, "_id": "BeeJh5lz3k6kSIzA" },	200 OK, retornando

/produtos sem token	POST	Valida se é possível cadastras um produto sem token	{
			"message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
			}
				401, Unauthorized

/produtos/{{_id}}	DELETE	Deleta produtopor ID	{
			"message": "Não é permitido excluir produto que faz parte de carrinho",
			"idCarrinhos": [
			"HD4RKeRAc4F697tg",
			"TzLjHnGKI0zg2yiH"
			]
			}
			Retornou 400 Bad Request
				200 ok, JSOn retornando DELETE

/produtos/{{_id}}	PUT	Edita produto por ID	{
			"_id": "_id não é permitido"
			}
			Retornou 400 Bad Request
				200 ok, atualizado com sucesso

- **ENDPOINTS /carrinhos**

Endpoint	Método	Descrição	Retorno do Teste	Resultado Esperado

/carrinhos	GET	Lista carrinhos cadastrados	{
			"quantidade": 3,
			"carrinhos": [
			{
			"produtos": [
			{
			"idProduto": "RSECrOSq6PKQvWQr",
			"quantidade": 1,
			"precoUnitario": 14
			}
			],
			"precoTotal": 14,
			"quantidadeTotal": 1,
			"idUsuario": "JUcxHKRX1ZyE9THg",
			"_id": "Ddj67WB2jlew5DLw"
			},
				200 OK Lista de carrinhos

/carrinhos	POST	Cadastra carrinho	{
			"message": "Produto não encontrado",
			"item": {
			"idProduto": "Ddj67WB2jlew5DLw",
			"quantidade": 1,
			"index": 0
			}
			}
			Retornou 400 BAD Request
				201, cadastrado com sucesso

/carrinhos/{_id}	Busca carrinho por ID	GET	"produtos": [{ "idProduto": "BeeJh5lz3k6kSIzA", "quantidade": 1, "precoUnitario": 470 }, { "idProduto": "K6leHdftCeOJj8BJ",	200, carrinho encontrado

/carrinhos/concluir-compra	Exclui carrinho	DELETE	{
			"message": "Não foi encontrado carrinho para esse usuário"
			}
			Retornou 200, porém nenhum registro foi excluido
				200, Registro excluido com sucesso

/carrinhos/cancelar-compra	Exclui carrinho e retorna produtos para o estoque	DELETE	{
			"message": "Não foi encontrado carrinho para esse usuário"
			}
			Retornou 200, porém nenhum registro foi excluido
				200, Registro excluido com sucesso

Riscos e Mitigações encontrados em testes de request

Bug	Risco	Mitigação
Exclusão Inconsistente	Inconsistência na API. O `200 OK` para uma falha confunde sistemas.	Mudar para `404 Not Found` quando o registro a ser excluído não existir.
`ID` de Usuário Editável	Vulnerabilidade de segurança. O campo `_id` não pode ser alterado.	Ignorar ou rejeitar tentativas de alterar o campo `_id` na requisição `PUT`.
Erro na Busca por `ID`	Mensagem de erro enganosa. O `400 Bad Request` para um ID não existente é incorreto.	Mudar o status para `404 Not Found` para indicar recurso não encontrado.
Falha em CT-19 ao CT-25	Inoperabilidade de funcionalidades críticas como exclusão e controle de estoque.	Implementar carrinhos de compra
Erro na Exclusão do Carrinho	Similar ao bug do usuário, com retorno `200 OK` para falhas	O endpoint de `DELETE` deve retornar `404 Not Found` se o carrinho não existir.

Riscos e Mitigações encontrados em testes manuais

Bug	Risco	Mitigação

Não é possível remover produtos individualmente da lista de produtos	A falta de uma função para remover produtos individualmente do carrinho pode levar à perda de vendas, pois o cliente, frustrado com a falta de controle sobre os itens, pode simplesmente abandonar a compra.	Implementar um botão "Remover" ao lado de cada produto no carrinho. Permitindo o cliente gerenciar os itens individualmente.
Não há como adicionar um produto ao carrinho.	O cliente não consegue prosseguir com o processo de compra, pois a funcionalidade principal (adicionar itens ao carrinho de compras) está ausente.	Desenvolver e implementar a funcionalidade de carrinho de compras.
Não há funcionalidades de gerenciamento de conta (editar ou excluir)	Falta de controle sobre dados pessoais, comprometendo a segurança e a conformidade com leis de privacidade. Isso leva à insatisfação do usuário.	Implementar páginas de edição de perfil e exclusão de conta, permitindo ao usuário gerenciar seus dados de forma segura.
Ao corrigir erro de credenciais no cadastro, campos não são limpos e senha é ignorada	O sistema não limpa os campos de senha ao identificar erros em outras credenciais, causando confusão e frustração. O usuário pode desistir do cadastro.	Implementar uma regra para limpar o campo de senha após qualquer erro de validação, forçando o usuário a redigitá-la para segurança e clareza.
Sistema permite que o nome de usuário seja o mesmo que o e-mail	O sistema permite que o nome de usuário seja igual ao e-mail, o que gera inconsistência nos dados e pode causar problemas em futuras funcionalidades.	Adicionar uma regra de validação que impeça o nome de usuário de ser o mesmo que o e-mail, garantindo que cada campo tenha uma função única.
Layout da aplicação em mobile não é responsivo, causando estouro de campos e problemas de visualização.	A falta de responsividade no layout mobile prejudica a experiência do usuário, podendo levar à desistência de compra e perda de clientes, pois a interface se torna ilegível e inutilizável.	Implementar um design responsivo para garantir a visualização correta em dispositivos móveis.
Não é possível editar informações de	A ausência da funcionalidade de edição	Desenvolver a interface de usuário (front-end)

usuários.	de usuário impede que o cliente atualize seus dados pessoais.	para permitir a edição de perfil.
Não é possível excluir um usuário, apenas fazer `logout`.	A impossibilidade de excluir a própria conta vai contra leis de proteção de dados e pode gerar problemas de privacidade.	Implementar uma funcionalidade de exclusão de conta no front-end, permitindo que o usuário remova seus dados permanentemente.

Testes de candidatos á automação com Robot Framework

Validação de Dados e Segurança

- **CT-02:** Cadastro com e-mail já existente ✓
- **CT-06:** Múltiplos logins com senha incorreta ✓
- **CT-08:** Login com credenciais inválidas ✓
- **CT-16:** Tentar cadastrar com nome de produto duplicado ✓
- **CT-17:** Tentar cadastrar com quantidade negativa ✓
- **CT-45:** Tentar cadastrar com preço negativo ✓
- **CT-27:** Acesso a rota protegida sem token ✓
- **CT-44:** Login com credenciais estáticas ✓

Fluxos de Negócio Críticos

- Cadastro de usuário dinâmico com sucesso ✓
- Cadastro de usuário com sucesso ✓
- Login com credenciais válidas ✓
- Cadastro de produto dinâmico com sucesso ✓
- Cadastro de produto com sucesso ✓

Validação de Regras de Negócio

- Tentar cadastrar com e-mail inválido ✓
- Tentar cadastrar com senha vazia ✓
- Tentar editar um usuário com um ID que não existe. X
- Adicionar produto que não existe X

- Tentar cadastrar com injeção de script ✓
- Tentar cadastrar com campos obrigatórios vazios ✓
- Adicionar produto com estoque insuficiente. X
- Tentar cadastrar com preço ou quantidade zero ✓
- Excluir usuário com carrinho ativo. X
- Finalizar compra com carrinho vazio. X
- Finalizar compra com produto removido X
- Remover produto individual do carrinho. X

Validação de Mensagens

- Tentar cadastrar com campos não existentes ✓
- Tentar editar um produto com ID inexistente ✓
- Adicionar produto que não existe X
- Adicionar produto com estoque insuficiente X
- Finalizar compra com sucesso ✓
- Finalizar compra com carrinho vazio X
- Finalizar compra com produto removido X
- Remover produto individual do carrinho X
- Cancelar compra com sucesso ✓

CRUD

- Edição de usuário com sucesso ✓
- Tentar editar um usuário com um ID que não existe X
- Exclusão de usuário com sucesso ✓
- Listar todos os usuários com sucesso ✓
- Buscar por um usuário específico por ID ✓
- Edição de produto com sucesso ✓
- Tentar editar um produto com ID que não existe X
- Exclusão de produto com sucesso ✓
- Listar todos os produtos com sucesso ✓
- Buscar por um produto específico por ID ✓

Fluxo de Autenticação Incorreta:

Testar se a API retorna 401 Unauthorized para endpoints protegidos quando se tenta acessá-los com um token inválido, expirado ou ausente. ✓

Cenários de Teste com Robot Framework (CT)

Cenário	Tags	Test Step	Expected Result

Cenário 01: POST Cadastrar usuário comum estático	POST, USUARIOS, ESTATICO	Realizar uma requisição POST com dados estáticos para criar um usuário comum.	O usuário é criado com sucesso, retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.
Cenário 02: POST Cadastrar usuário admin estático	POST, USUARIOS, ESTATICO	Realizar uma requisição POST com dados estáticos para criar um usuário admin.	O usuário admin é criado com sucesso, retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.
Cenário 03: POST Cadastrar usuário com email existente estático	POST, USUARIOS, NEGATIVO, ESTATICO	Tentar criar um usuário usando um email que já está cadastrado.	A requisição falha, retornando o status code 400 e a mensagem 'Este email já está sendo usado'.
Cenário 04: POST Login com credenciais válidas estático	POST, LOGIN, ESTATICO	Realizar uma requisição POST com email e senha válidos.	O login é bem-sucedido, retornando o status code 200 e a mensagem 'Login realizado com sucesso'.
Cenário 05: POST Login com credenciais inválidas estático	POST, LOGIN, NEGATIVO, ESTATICO	Tentar realizar login com um email ou senha que não correspondem a um usuário existente.	O login falha, retornando o status code 401 e a mensagem 'Email e/ou senha inválidos'.
Cenário 06: POST Múltiplos logins incorretos estático	POST, LOGIN, NEGATIVO, ESTATICO	Tentar logar várias vezes com a mesma credencial de usuário, mas com senhas diferentes e incorretas.	Cada tentativa de login falha, retornando o status code 401 e a mensagem 'Email e/ou senha inválidos'.
Cenário 07: POST Criar	POST, PRODUTOS, ESTATICO	Fazer login como admin para obter um token e criar	O produto é criado com sucesso,

produto com dados estáticos		um novo produto com dados fixos.	retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.
Cenário 08: POST Criar produto duplicado estático	POST, PRODUTOS, NEGATIVO, ESTATICO	Tentar criar um produto com um nome que já existe no sistema.	A criação do produto falha, retornando o status code 400 e a mensagem 'Já existe produto com esse nome'.
Cenário 09: POST Criar produto com preço negativo estático	POST, PRODUTOS, NEGATIVO, ESTATICO	Tentar criar um produto com o campo 'preco' com um valor negativo.	A criação do produto falha, retornando o status code 400 e a mensagem de erro no campo 'preco'.
Cenário 10: POST Criar produto com quantidade negativa estático	POST, PRODUTOS, NEGATIVO, ESTATICO	Tentar criar um produto com o campo 'quantidade' com um valor negativo.	A criação do produto falha, retornando o status code 400 e a mensagem de erro no campo 'quantidade'.
Cenário 11: POST Criar produto sem token estático	POST, PRODUTOS, NEGATIVO, ESTATICO	Tentar criar um produto sem incluir o token de autorização.	A requisição falha, retornando o status code 401 e a mensagem 'Token de acesso ausente, inválido, expirado ou usuário do token não existe mais'.
Cenário 12: POST Criar carrinho com dados estáticos	POST, CARRINHOS, ESTATICO	Criar um usuário, um produto, fazer login para obter um token e criar um carrinho com o produto.	O carrinho é criado com sucesso, retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.

Cenário 13: GET Listar usuários estático	GET, USUARIOS, ESTATICO	Realizar uma requisição GET para o endpoint de usuários.	A requisição é bem-sucedida, retornando o status code 200 e a lista de usuários.
Cenário 14: GET Obter usuário por ID válido estático	GET, USUARIOS, ESTATICO	Criar um usuário e buscar por seu ID para validar o retorno dos dados.	A requisição é bem-sucedida, retornando o status code 200 e os dados do usuário correspondente.
Cenário 15: GET Obter usuário por ID inválido estático	GET, USUARIOS, NEGATIVO, ESTATICO	Tentar buscar um usuário utilizando um ID que não tem o formato alfanumérico correto.	A requisição falha, retornando o status code 400 e a mensagem 'id deve ter exatamente 16 caracteres alfanuméricos'.
Cenário 16: GET Listar produtos estático	GET, PRODUTOS, ESTATICO	Realizar uma requisição GET para o endpoint de produtos.	A requisição é bem-sucedida, retornando o status code 200 e a lista de produtos.
Cenário 17: GET Obter produto por ID válido estático	GET, PRODUTOS, ESTATICO	Criar um produto e buscar por seu ID para validar o retorno dos dados.	A requisição é bem-sucedida, retornando o status code 200 e os dados do produto correspondente.
Cenário 18: GET Obter produto por ID inválido estático	GET, PRODUTOS, INVALIDO, ESTATICO	Tentar buscar um produto utilizando um ID que não tem o formato alfanumérico correto.	A requisição falha, retornando o status code 400 e a mensagem 'id deve ter exatamente 16 caracteres alfanuméricos'.
Cenário 19: GET Listar carrinhos estático	GET, CARRINHOS, ESTATICO	Realizar uma requisição GET para o endpoint de carrinhos.	A requisição é bem-sucedida, retornando o status code 200

			e a lista de carrinhos.
Cenário 20: PUT Atualizar carrinho estático	PUT, CARRINHOS, ESTATICO	Criar um carrinho e atualizar seus dados com um novo produto	O carrinho é atualizado com sucesso, retornando o status code 200 e a mensagem 'Registro alterado com sucesso'.
Cenário 21: GET Produto com ID não encontrado estático	GET, PRODUTOS, NEGATIVO, ESTATICO	Tentar buscar um produto com um ID válido, mas que não está no banco de dados.	A busca falha, retornando o status code 400 e a mensagem 'id deve ter exatamente 16 caracteres alfanuméricos'.
Cenário 22: PUT Atualizar usuário estático	PUT, USUARIOS, ESTATICO	Criar um usuário e atualizá-lo com novos dados estáticos.	A atualização é bem-sucedida, retornando o status code 200 e a mensagem 'Registro alterado com sucesso'.
Cenário 23: PUT Atualizar produto estático	PUT, PRODUTOS, ESTATICO	Criar um produto e atualizá-lo com novos dados estáticos.	A atualização é bem-sucedida, retornando o status code 200 e a mensagem 'Registro alterado com sucesso'.
Cenário 24: PUT Atualizar produto sem alterar dados estático	PUT, PRODUTOS, ESTATICO	Criar um produto e realizar uma requisição PUT com os mesmos dados originais.	A requisição é bem-sucedida, retornando o status code 200 e a mensagem 'Registro alterado com sucesso'.
Cenário 25: PUT Atualizar produto com preco negativo estatico	PUT, PRODUTOS, NEGATIVO, ESTATICO	Criar um produto e tentar atualizá-lo com um valor de preço negativo.	A atualização falha, retornando o status code 400 e a mensagem de

			erro no campo 'preco'.
Cenário 26: PUT Atualizar produto com quantidade negativa estatico	PUT, PRODUTOS, NEGATIVO, ESTATICO	Criar um produto e tentar atualizá-lo com um valor de quantidade negativo.	A atualização falha, retornando o status code 400 e a mensagem de erro no campo 'quantidade'.
Cenário 27: PUT Atualizar produto sem token estatico	PUT, PRODUTOS, NEGATIVO, ESTATICO	Tentar atualizar um produto sem incluir o token de autorização.	A requisição falha, retornando o status code 401 e a mensagem 'Token de acesso ausente, inválido, expirado ou usuário do token não existe mais'.
Cenário 28: PUT Atualizar usuario com email duplicado estatico	PUT, USUARIOS, NEGATIVO, ESTATICO	Tentar atualizar o email de um usuário para um email que já está cadastrado por outro usuário.	A atualização falha, retornando o status code 400 e a mensagem 'Este email já está sendo usado'.
Cenário 29: PUT Atualizar produto com nome duplicado estatico	PUT, PRODUTOS, NEGATIVO, ESTATICO	Tentar atualizar o nome de um produto para um nome já existente no sistema.	A atualização falha, retornando o status code 400 e a mensagem 'Já existe produto com esse nome'.
Cenário 30: DELETE Excluir produto estático	DELETE, PRODUTOS, ESTATICO	Criar um produto e excluí-lo usando o seu ID e um token de administrador.	O produto é excluído com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.
Cenário 31: DELETE Cancelar compra do carrinho estático	DELETE, CARRINHOS, ESTATICO	Criar um carrinho com um produto e depois cancelar a compra.	A compra é cancelada com sucesso, o carrinho é removido e o estoque do

			produto é reabastecido.
Cenário 32: DELETE Concluir compra do carrinho estático	DELETE, CARRINHOS, ESTATICO	Criar um carrinho com um produto e depois concluir a compra.	A compra é concluída com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.
Cenário 33: DELETE Excluir usuario estatico	DELETE, USUARIOS, ESTATICO	Criar um usuário admin e excluí-lo utilizando seu ID e um token válido.	O usuário é excluído com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.
Cenário 34: DELETE Excluir produto com ID de usuário estático	DELETE, PRODUTOS, NEGATIVO, ESTATICO	Tentar excluir um produto com o ID de um usuário, usando o token de um administrador.	A exclusão falha, retornando o status code 403 e a mensagem 'Rota exclusiva para administradores'.
Cenário 35: POST Cadastrar usuário dinâmico	POST, DINAMICO	Criar um usuário com dados dinâmicos.	O usuário é criado com sucesso, retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.
Cenário 36: POST Login usuário dinâmico	POST, DINAMICO	Realizar login com um usuário dinâmico e validar a resposta.	O login é bem-sucedido, retornando o status code 200 e a mensagem 'Login realizado com sucesso'.
Cenário 37: POST Criar produto com token de administrador	POST, DINAMICO	Criar um usuário administrador, logar para obter um token e criar um produto com	O produto é criado com sucesso, retornando o status code 201 e a mensagem

		dados dinâmicos usando o token.	'Cadastro realizado com sucesso'.
Cenário 38: POST Adicionar e Criar carrinho completo dinâmico	POST, DINAMICO	Criar um usuário admin, um produto dinâmico e depois criar um carrinho com esse produto.	O carrinho é criado com sucesso, retornando o status code 201 e a mensagem 'Cadastro realizado com sucesso'.
Cenário 39: POST Registro com email existente dinâmico	POST, DINAMICO	Criar um usuário dinâmico e, em seguida, tentar criar um segundo usuário com o mesmo email.	O sistema retorna o status code 400 e a mensagem 'Este email já está sendo usado'.
Cenário 40: POST Múltiplos logins com senha incorreta dinâmico	POST, DINAMICO	Criar um usuário dinâmico e tentar logar várias vezes com a senha errada.	Cada tentativa de login falha, retornando o status code 401 e a mensagem 'Email e/ou senha inválidos'.
Cenário 41: POST Login com credenciais inválidas dinâmico	POST, DINAMICO	Tentar realizar login com um email e senha que não existem no sistema.	O login falha, retornando o status code 401 e a mensagem 'Email e/ou senha inválidos'.
Cenário 42: POST Registro com nome de produto duplicado dinâmico	POST, DINAMICO	Criar um produto dinamicamente e tentar criar um segundo produto com o mesmo nome.	O sistema retorna o status code 400 e a mensagem 'Já existe produto com esse nome'.
Cenário 43: POST Registro com preço negativos dinâmico	POST, DINAMICO	Tentar criar um produto com o campo 'preco' com um valor negativo.	O sistema retorna o status code 400 e a mensagem 'preco deve ser um número positivo'.

Cenário 44: POST Registro com quantidade negativa dinâmico	POST, DINAMICO	Tentar criar um produto com o campo 'quantidade' com um valor negativo.	O sistema retorna o status code 400 e a mensagem 'quantidade deve ser maior ou igual a 0'.
Cenário 45: POST Acesso a rota protegida sem token dinâmico	POST, DINAMICO	Tentar criar um produto sem fornecer um token de autenticação.	O sistema retorna o status code 401 e a mensagem 'Token de acesso ausente, inválido, expirado ou usuário do token não existe mais'.
Cenário 46: GET Listar usuários dinâmico	GET, DINAMICO	Realizar uma requisição GET para o endpoint de usuários.	A requisição é bem-sucedida, retornando o status code 200 e uma lista de usuários.
Cenário 47: GET Obter usuário por ID válido dinâmico	GET, DINAMICO	Criar um usuário dinâmico e, em seguida, buscar por seu ID recém-criado.	O sistema encontra o usuário, retornando o status code 200 e os dados do usuário correspondente.
Cenário 48: GET Obter usuário por ID inválido dinâmico	GET, DINAMICO	Tentar buscar um usuário utilizando um ID inválido (formato incorreto).	A busca falha, retornando o status code 400 e a mensagem 'id deve ter exatamente 16 caracteres alfanuméricos'.
Cenário 49: PUT Atualizar produto com preco negativo dinamico	PUT, DINAMICO	Criar um produto e tentar atualizá-lo com um valor de preço negativo.	A atualização falha, retornando o status code 400 e a mensagem 'preco deve ser um número positivo'.
Cenário 50: PUT Atualizar	PUT, DINAMICO	Criar um produto e tentar atualizá-	A atualização falha, retornando

produto com quantidade negativa dinamico		lo com um valor de quantidade negativo.	o status code 400 e a mensagem 'quantidade deve ser maior ou igual a 0'
Cenário 51: PUT Atualizar produto sem token dinamico	PUT, DINAMICO	Criar um produto e tentar atualizá-lo sem fornecer um token.	A atualização falha, retornando o status code 401 e a mensagem 'Token de acesso ausente, inválido, expirado ou usuário do token não existe mais'.
Cenário 52: PUT Atualizar usuario com email duplicado dinamico	PUT, DINAMICO	Criar dois usuários dinâmicos e tentar atualizar o email do primeiro com o email do segundo.	A atualização falha, retornando o status code 400 e a mensagem 'Este email já está sendo usado'.
Cenário 53: PUT Atualizar produto com nome duplicado dinâmico	PUT, DINAMICO	Criar dois produtos dinâmicos e tentar atualizar o nome do segundo com o nome do primeiro.	A atualização falha, retornando o status code 400 e a mensagem 'Já existe produto com esse nome'.
Cenário 54: DELETE Excluir usuário dinâmico	DELETE, DINAMICO	Criar um usuário, obter um token de administrador e excluir o usuário recém-criado	O usuário é excluído com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.
Cenário 55: DELETE Excluir produto dinâmico	DELETE, DINAMICO	Criar um produto, obter um token de administrador e excluir o produto recém-criado.	O produto é excluído com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.

Cenário 56: DELETE Concluir compra do carrinho dinâmico	DELETE, DINAMICO	Criar um usuário admin, criar um produto, adicionar ao carrinho e concluir a compra.	A compra é concluída com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso'.
Cenário 57: DELETE Cancelar compra do carrinho dinâmico	DELETE, DINAMICO	Criar um usuário admin, criar um produto, adicionar ao carrinho e cancelar a compra.	A compra é cancelada com sucesso, retornando o status code 200 e a mensagem 'Registro excluído com sucesso. Estoque dos produtos reabastecido'.
Cenario 01 banco de dados: Adicionar Novo Usuario ao Banco JSON	JSON, MANIPULACAO	Adicionar um novo usuário ao banco de dados JSON e verificar se ele foi encontrado.	O usuário é adicionado e encontrado com sucesso no banco de dados.
Cenario 02 banco de dados: Contar Usuarios no Banco	JSON, CONSULTA	Contar o número total de usuários no banco JSON.	A contagem de usuários é maior ou igual a 3.
Cenario 03 banco de dados: Validar Estrutura dos Bancos JSON	JSON, VALIDACAO	Validar a estrutura dos bancos de dados JSON de usuários, endpoints e status codes.	Todas as estruturas dos arquivos JSON são validadas com sucesso.
Cenario 04 banco de dados: Testar Busca de Usuario Inexistente	JSON, ERRO	Tentar buscar um usuário por um email que não existe no banco.	Um erro é retornado, indicando que o usuário não foi encontrado.
Cenario 05 banco de dados: Adicionar e	JSON, CRUD	Adicionar um usuário temporário, verificar se a	O usuário é adicionado e removido com sucesso, e a

Remover Usuario Dinamicamente		contagem aumenta e depois remover, verificando se a contagem volta ao normal.	contagem de usuários é restaurada corretamente.
Cenario 06 banco de dados: Testar Todos os Usuarios do Banco Dinamicamente	SON, DINAMICO	Carregar todos os usuários do banco de dados e iterar sobre eles, validando se os campos necessários não estão vazios.	Todos os usuários são validados com sucesso, com seus campos obrigatórios preenchidos e no formato correto.

Cenários de Teste NÃO executados com Robot Framework (CT)

Cenário	Tags	Test Step	Expected Result
Tentar editar um usuário com um ID que não existe	PUT, USUARIOS, NEGATIVO	Tentar realizar um PUT para o endpoint de usuários com um ID inexistente.	A requisição falha, retornando o status code 404 e uma mensagem de erro indicando que o usuário não foi encontrado.
Adicionar produto que não existe	PUT, PRODUTOS, NEGATIVO	Tentar adicionar ao carrinho um produto com um ID que não existe.	A requisição falha, retornando o status code 400 e uma mensagem de erro
Adicionar produto com estoque insuficiente	PUT, PRODUTOS, NEGATIVO	Tentar adicionar uma quantidade maior de produto do que o estoque disponível.	A requisição falha, retornando o status code 400 e uma mensagem de erro sobre o estoque.
Excluir usuário com carrinho ativo	PUT, USUARIOS, NEGATIVO	Tentar excluir um usuário que possui um carrinho com produtos.	A exclusão falha, retornando o status code 400 e a mensagem de que o usuário não pode ser excluído por ter um carrinho ativo.

Finalizar compra com carrinho vazio	DELETE, CARRINHOS, NEGATIVO	Tentar finalizar uma compra com o carrinho vazio.	A requisição falha, retornando o status code 400 e uma mensagem de erro informando que o carrinho está vazio.
Finalizar compra com produto removido	PUT, PRODUTOS, NEGATIVO	Adicionar um produto ao carrinho, excluí-lo e, em seguida, tentar finalizar a compra.	A requisição falha, retornando o status code 400 e uma mensagem de que o produto não está mais disponível.
Remover produto individual do carrinho	PUT, PRODUTOS, NEGATIVO	Tentar excluir um produto específico do carrinho.	O produto é removido com sucesso, retornando o status code 200 e uma mensagem de sucesso.
Tentar editar um produto com ID que não existe	PUT, PRODUTOS, NEGATIVO	Tentar realizar um PUT para o endpoint de produtos com um ID inexistente.	A requisição falha, retornando o status code 404 e uma mensagem de erro indicando que o produto não foi encontrado.

Cobertura de Testes Com Robot Framework

ID do Cenário	**Cenários**	**Cobertura do Requisito**
Usuários	CT-01, CT-02, CT-03, CT-04, CT-05, CT-06, CT-07, CT-08, CT-11, CT-33, CT-34, CT-35, CT-36, CT-37	Cobertura Completa:Os testes de Robot Framework cobrem o cadastro, login, busca, edição e exclusão de usuários
Produtos	CT-12, CT-13, CT-14, CT-15, CT-16, CT-17, CT-18, CT-38, CT-39, CT-40, CT-41, CT-42	Cobertura Parcial: Os testes de Robot Framework cobrem a criação de produtos e validações de dados

		negativos, a edição, exclusão e busca por um produto específico são testadas , mas a cobertura total ainda depende de cenários não mapeados
Carrinhos	CT-19, CT-20, CT-21, CT-22, CT-23, CT-24, CT-25	Cobertura Parcial:Os testes de Robot Framework cobre a adição de produtos e os fluxos de conclusão e cancelamento de compra. A documentação indica a falta de cenários como remoção individual e estoque insuficiente
Segurança	CT-27	Cobertura Básica: Os testes de Robot Framework verifica o acesso não autorizado. Testes mais avançados, como injeção de dados, não estão no escopo da automação.