ServeRest API - Testes

Squad 3 - CapsLock
Maycon Douglas da Silva

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.

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).

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 de Performance

Avaliar tempo de resposta e comportamento sob carga.

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.

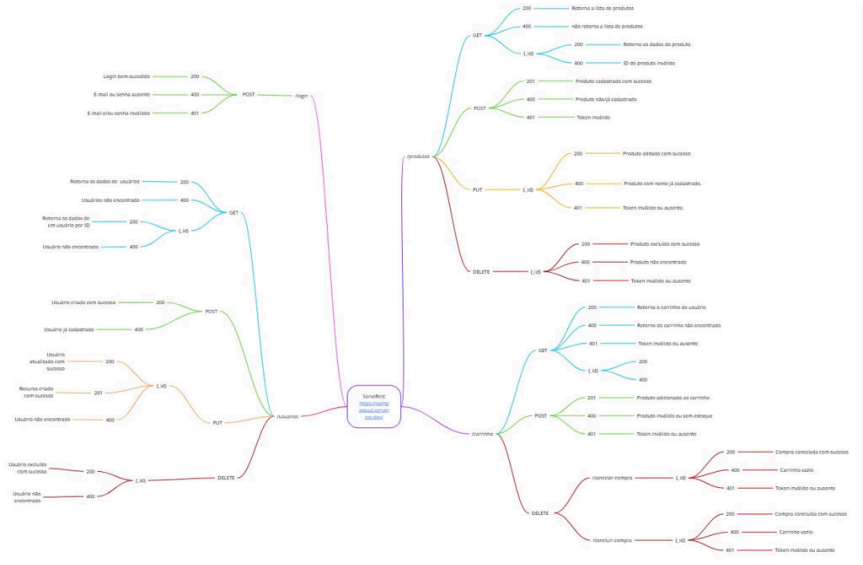
Execução sequencial dos requests em modo Runner, com gravação de logs.

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

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=/

Cenários de Teste (CT)

ID do Cenário	Módulo	Cenário	Ação	Resultado Esperado	Prioridade	Status
CT-01	Usuários	Cadastro de	Enviar POST	201 Created.	Alta	Executado

		usuário com sucesso	para /usuarios com dados válidos (nome, email, senha).	Usuário criado no banco e dados de resposta corretos.		
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 o e-mail em uso.	Alta	Executado
CT-03	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-04	Usuários	Login com credenciais inválidas	Enviar POST para /login com senha errada.	401 Unauthorized. Mensagem de erro apropriada.	Média	Executado
CT-05	Produtos	Cadastro de produto sem autenticação	Enviar POST para /produtos sem o token de autenticação.	401 Unauthorized. Mensagem de erro de autenticação.	Alta	Executado
CT-06	Produtos	Cadastro de produto com sucesso	Enviar POST para /produtos com	201 Created. Produto criado e dados	Alta	Executado

			token e dados válidos.	de resposta corretos.		
CT-07	Carrinhos	Adicionar produto com estoque insuficiente	Enviar POST para /carrinhos solicitando quantidade maior que o estoque.	400 Bad Request. Mensagem de erro de estoque.	Média	Executado
CT-08	Carrinhos	Finalizar compra com sucesso	Enviar DELETE para /carrinhos/concluir-compra com token válido.	200 OK. Estoque do produto reduzido, e carrinho removido.	Alta	Executado
CT-09	Segurança	Acesso a rota protegida sem token	Tentar acessar GET /carrinhos sem enviar um token.	401 Unauthorized. Mensagem de erro de autenticação.	Alta	Executado

Matriz de Risco ServeRest

ID do Cenário	Cenário	Probabilidade	Impacto	Risco	Prioridade de Execução
CT-01	Cadastro de usuário com sucesso	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
CT-02	Cadastro com e-mail já existente	Média (2)	Alto (3)	6 (Crítico)	1 - Crítica
CT-03	Login de usuário	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica

	com credenciais válidas				
CT-04	Login de usuário com credenciais inválidas	Média (2)	Alto (3)	6 (Crítico)	2 - Alta
CT-05	Tentar cadastrar produto sem autenticação	Alta (3)	Média (2)	6 (Crítico)	2 - Alta
CT-06	Cadastro de produto com sucesso	Média (2)	Média (2)	4 (Alta)	2 - Alta
CT-07	Adicionar produto com estoque insuficiente	Média (2)	Média (2)	4 (Alta)	2 - Alta
CT-08	Finalizar compra com sucesso	Alta (3)	Alto (3)	9 (Crítico)	1 - Crítica
CT-09	Acesso a rota protegida sem token	Alta (3)	Baixo (1)	3 (Média/Alta)	3 - Média
CT-10	Excluir usuário	Média (2)	Alto (3)	3 (Média/Alta)	3 - Média

Cobertura de teste

ID do Cenário	Cenários	Cobertura do Requisito

Usuários	CT-01, CT-02, CT-03, CT-04, CT-09,CT-10	Cobertura Completa: Testes do cadastro, login, busca de usuários e a verificação de autenticação (erro e sucesso)
Produtos	CT-05, CT-06	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-07, CT-08	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-09 (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.

Testes de candidatos á automação

CT-01: Cadastro de usuário com sucesso

Ponto de entrada da aplicação. A automação garante que o fluxo de criação de novos usuários está sempre funcionando.

CT-03: Login de usuário com credenciais válidas

Automação garante que o acesso à API está sempre operacional.

CT-08: Finalizar compra com sucesso Não foi possível executar

A automação verifica se a lógica de compra e a atualização de estoque funcionam como esperado.

CT-08: Excluir usuário com sucesso

A automação valida se é possível remover um usuário existente do sistema.

Testes de Fluxos de Negócio

CT-07: Adicionar produto com estoque insuficiente Não foi possível executar.

Essencial para a integridade dos dados e regras de negócio. A automação verifica que a API está protegida contra transações inválidas.

CT-06: Cadastro de produto com sucesso

A automação garante que novos produtos podem ser adicionados ao sistema.

CT-04: Login de usuário com credenciais inválidas

A automação garante que a API está protegida contra acessos não autorizados.

Testes de Validação e Integridade

CT-02: Cadastro com e-mail já existente

A automação verifica que a API impede a duplicação de dados, mantendo a integridade da base de usuários.

CT-05: Tentar cadastrar produto sem autenticação

A automação valida a regra de negócio de que apenas usuários autenticados podem interagir com a API, reforçando a segurança.

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

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	Adicionar uma regra de validação que impeça o nome de usuário de ser o mesmo que o e-mail, garantindo que cada

	problemas em futuras funcionalidades.	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 usuários.	A ausência da funcionalidade de edição de usuário impede que o cliente atualize seus dados pessoais.	Desenvolver a interface de usuário (front-end) 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.

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-07 e CT-08	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.

Testes de automação realizados

- CT-01: Cadastro de usuário com sucesso
- CT-03: Login de usuário com credenciais válidas
- CT-06: Cadastro de produto com sucesso
- CT-04: Login de usuário com credenciais inválidas
- CT-02: Cadastro com e-mail já existente
- CT-05: Tentar cadastrar produto sem autenticação
- CT-08: Excluir usuário com sucesso

Mapeamento de Endpoints e Casos de Teste

- **ENDPOINTS /usarios**

Endpoint	**Método**	**Descrição**	**Retorno do Teste**	**Resultado Esperado**
/login	POST	Realiza login do usuário	`{ "message": "Login realizado com sucesso", "authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InRlc3RlQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiMTIzIiwiaWF0IjoxNzU1MjAxMTAyLCJleHAiOjE3NTUyMDE3MDJ9.zTln6xdLtvfaAI2Mr1YDpVQ8406CYaY7U7ilGGdHuZI" }`	200 OK, JSON , usuario cadastrado
/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,	200 OK, retornando

			"descricao": "Mouse", "quantidade": 276, "_id": "BeeJh5lz3k6kSIzA" },	

/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": [{	200 OK Lista de carrinhos

			"idProduto": "RSECrOSq6PKQvWQr", "quantidade": 1, "precoUnitario": 14 }], "precoTotal": 14, "quantidadeTotal": 1, "idUsuario": "JUcxHKRX1ZyE9THg", "_id": "Ddj67WB2jlew5DLw" },	
/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-	Exclui carrinho	DELETE	{ "message": "Não foi encontrado carrinho para esse usuário" }	200, Registro excluido com sucesso

compra			Retornou 200, porém nenhum registro foi excluido	

/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