Plano de teste - API Serverest

Thiago Couto

1. Introdução

Este projeto foi realizado tomando como base as funcionalidades da API Serverest, sendo sua principal função disponibilizar a realização de compras online.

2. Objetivos

Este documento tem como objetivo realizar o planejamento dos testes da API Serverest, cuja principal função é a compra de produtos. O motivo para realizar testes nesta API é não só validar o funcionamento correto de suas funcionalidades, mas também apontar suas falhas.

3. Escopo

Neste tópico serão apresentadas as funcionalidades da API Serverest, assim como as possíveis rotas, métodos e respostas (status code).

Método	Endpoint	Descrição	Possíveis status code
POST	/login	Realizar login	200 e 400
GET	/usuarios	Listar usuários cadastrados	200
POST	/usuarios	Cadastrar usuário	201 e 400
GET	/usuarios/{_id}	Buscar usuário por ID	200 e 400
DELETE	/usuarios/{_id}	Excluir usuário	200 e 400
PUT	/usuarios/{_id}	Editar usuário	200, 201 e 400
GET	/produtos	Listar produtos cadastrados	200
POST	/produtos	Cadastrar produto	201, 400, 401 e 403
GET	/produtos/{_id}	Buscar produto por ID	200 e 400
DELETE	/produtos/{_id}	Excluir produto	200, 400, 401 e 403
PUT	/produtos/{_id}	Editar produto	200, 201, 400, 401 e 403
GET	/carrinhos	Listar carrinhos cadastrados	200
POST	/carrinhos	Cadastrar carrinho	201, 400 e 401
GET	/carrinhos/{_id}	Buscar carrinho por	200 e 400

		ID	
DELETE	/carrinhos/concluir-compra	Excluir carrinho	200 e 401
DELETE	/carrinhos/cancelar-compra	Excluir carrinho e retornar produtos para estoque	200 e 401

4. Suíte de casos de teste

Serão realizados testes nas principais funcionalidades da API Serverest, são elas, as rotas "/login", "/usuarios", "/carrinhos" e "/produtos". O tópico em questão irá conter o ID dos testes, descrição, os dados de entrada, resultados esperados, resultados obtidos e se o teste passou ou falhou.

Caso de teste	Descrição	Dados de entrada	Resultado esperado	Resultado obtido	Passou/Falhou
TC01	Realizar uma requisição **POST** à rota **/login** com dados de um usuário **válido**.	{ "email": "fulano@qa.com", "password": "teste" }	Uma resposta com a mensagem: "Login realizado com sucesso"	"Login realizado com sucesso"	Passou
TC02	Realizar uma requisição **POST** à rota **/login** com dados de um usuário **inválido**.	{ "email": "fulano@qa.com", "password": "abc" }	Uma resposta com a mensagem: "Email e/ou senha inválidos"	"Email e/ou senha inválidos"	Passou
TC03	Realizar uma requisição **POST** à rota **/login** com **dados nulos**.	{ "email": null "password": null }	A documentação da API não possui resposta esperada para esta situação.	undefined	Falhou
TC04	Realizar uma requisição **POST** à rota /**login** com **campos vazios**.	{ "email": "", "password": "" }	A documentação da API não possui resposta esperada para esta situação.	undefined	Falhou
TC05	Realizar uma requisição **GET** à rota **/produtos**	Sem entrada de dados	A lista de produtos cadastrados e quantidade de produtos atualmente cadastrados.	Lista de produtos cadastrados e quantidade de produtos cadastrados.	Falhou
TC06	Realizar uma requisição **GET** à rota **/produtos/{_id}**	Neste caso: O dado será o 'ID' do produto, e será passado através da URL.	Uma resposta com os dados do produto cujo ID foi especificado na URL.	Dados do produto encontrado.	Passou
TC07	Realizar uma requisição **GET**	Neste caso: O dado	Uma resposta com	"Produto não	Passou

	à rota “**/produtos/{_id}**” com um **ID inválido**	será o ‘ID’ do produto, e será passado através da URL.	a mensagem: “Produto não encontrado”	encontrado”	

TC08	Realizar uma requisição **POST** à rota **/produtos** com dados de um produto **válido**.	Dados gerados automaticamente. Mas nesta estrutura:
		{
		"nome": "nome",
		"preco": preco,
		"descricao": "descricao",
		"quantidade": quantidade
		}
			Uma resposta com a mensagem: "Cadastro realizado com sucesso", e o ID do produto cadastrado
			"_id": “id”
				{
				"message": "Cadastro realizado com sucesso",
				"_id": "jogfODIlXsqxNFS2"
				}
					Passou

TC09	Realizar uma requisição **POST** à rota **/produtos** com dados de um produto com **nome já existente.**	Dados gerados automaticamente, mas com o campo “nome” com nome de um produto já existente.	Uma resposta com a mensagem: "Já existe produto com esse nome"	"Já existe produto com esse nome"	Passou

TC10	Realizar uma requisição **POST** à rota **/produtos** com dados de um produto válido, mas com **token de acesso ausente.**	Dados gerados automaticamente. Mas nesta estrutura:
		{
		"nome": "nome",
		"preco": preco,
		"descricao": "descricao",
		"quantidade": quantidade
		}
			Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	"Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	Passou

TC11	Realizar uma requisição **POST** à rota **/produtos** com dados de um produto válido, mas logado com usuário {**“administrador”: “false”}.**	Dados gerados automaticamente. Mas nesta estrutura:
		{
		"nome": "nome",
		"preco": preco,
		"descricao": "descricao",
		"quantidade": quantidade
		}
			Uma resposta com a mensagem: "Rota exclusiva para administradores"	"Rota exclusiva para administradores"	Passou

TC12	Realizar uma requisição **DELETE** à rota **/produtos/{_id}** com dados de um **produto válido**.	Neste caso: O dado será o ‘ID’ do produto, e será passado através da URL.	Uma resposta com a mensagem: "Registro excluído com sucesso"	"Registro excluído com sucesso"	Passou
TC13	Realizar uma requisição **DELETE** à rota **/produtos/{_id}** com dados de um produto válido mas com **token de acesso inválido.**	Neste caso: O dado será o ‘ID’ do produto, e será passado através da URL.	Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário	"Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	Passou

			do token não existe mais"		
TC14	Realizar uma requisição **DELETE** à rota **/produtos/{_id}** com dados de um produto válido mas logado com usuário {**“administrador”: “false”}.**	Neste caso: O dado será o ‘ID’ do produto, e será passado através da URL.	Uma resposta com a mensagem: "Rota exclusiva para administradores"	"Rota exclusiva para administradores"	Passou

TC15	Realizar uma requisição **PUT** à rota **/produtos/{_id}** alterando os dados de um **produto válido**.	O dado será o ‘ID’ do produto, e será passado através da URL.
		E também terá uma estrutura em seu body:
		{
		"nome": "nome",
		"preco": 160,
		"descricao": "descricao",
		"quantidade": 1
		}
			Uma resposta com a mensagem: "Registro alterado com sucesso"	"Registro alterado com sucesso"	Passou

TC16	Realizar uma requisição **PUT** à rota **/produtos/{_id}** alterando os dados de um produto válido mas com **token de acesso inválido.**	O dado será o ‘ID’ do produto, e será passado através da URL.
		E também terá uma estrutura em seu body:
		{
		"nome": "nome",
		"preco": 160,
		"descricao": "descricao",
		"quantidade": 1
		}
			Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	"Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	Passou

TC17	Realizar uma requisição **PUT** à rota **/produtos/{_id}** com dados de um produto válido mas logado com usuário {**“administrador”: “false”}.**	O dado será o ‘ID’ do produto, e será passado através da URL.
		E também terá uma estrutura em seu body:
		{
		"nome": "nome",
		"preco": 160,
		"descricao": "descricao",
		"quantidade": 1
		}
			Uma resposta com a mensagem: "Rota exclusiva para administradores"	"Rota exclusiva para administradores"	Passou

TC18	Realizar uma requisição	Dados gerados	Uma resposta com	{	Passou

	POST à rota **/usuarios** com dados de um usuário **válido**.	automaticamente. Mas nesta estrutura:
		{
		"nome": "nome",
		"email": “email”,
		"password": "password",
		"administrador": “true”
		}
			a mensagem: "Cadastro realizado com sucesso", e o ID do usuário cadastrado
			"_id": “id”
				"message": "Cadastro realizado com sucesso",
				"_id": "jogfODIlXsqxNFS2"
				}	

TC19	Realizar uma requisição **POST** à rota **/usuarios** com dados de um usuário **com email já existente**.	Dados gerados automaticamente. Mas nesta estrutura:
		{
		"nome": "nome",
		"email": “email já em uso”,
		"password": "password",
		"administrador": “true”
		}
			Uma resposta com a mensagem: "Este email já está sendo usado”	{
				"message": "Este email já está sendo usado”
				}
					Passou

TC20	Realizar uma requisição **DELETE** à rota **/usuarios** com dados de um usuário **válido**.	O dado será o ‘ID’ do usuário, e será passado através da URL.	Uma resposta com a mensagem: "Registro excluído com sucesso”	{
				"message": "Registro excluído com sucesso”
				}
					Passou

TC21	Realizar uma requisição **POST** à rota **/carrinhos** com dados de um produto válido adicionando ele ao carrinho.	"produtos": [
		{
		"idProduto": id,
		"quantidade": 1
		},
		]
			Uma resposta com a mensagem: "Cadastro realizado com sucesso”	{
				"message": "Cadastro realizado com sucesso”
				}
					Passou

TC22	Realizar uma requisição **POST** à rota **/carrinhos** com dados de um produto válido porém, adicionando o mesmo produto duas vezes ao carrinho.	"produtos": [
		{
		"idProduto": id,
		"quantidade": 1
		},
		{
		"idProduto": id,
		"quantidade": 1
		}
		]
			Uma resposta com a mensagem: "Não é permitido possuir produto duplicado”	{
				"message": "Não é permitido possuir produto duplicado”
				}
					Passou

TC23	Realizar uma requisição **POST** à rota **/carrinhos** com um **usuário com token ausente ou inválido**.	Os dados de entrada neste caso, será:
		"produtos": [
		{
		"idProduto": id,
		"quantidade": 1
		}
		e token de usuário ausente ou inválido.
			Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	{
				"message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
				}
					Passou

TC24	Realizar uma requisição **DELETE** à rota **/carrinhos/concluir-compra** com dados de um **usuário válido**.	Os dados de entrada neste caso, será o token do usuário (estar logado).	Uma resposta com a mensagem: "Registro excluído com sucesso”	{ "message": "Registro excluído com sucesso” }	Passou
TC25	Realizar uma requisição **DELETE** à rota **/carrinhos/concluir-compra** com um **usuário com token ausente ou inválido**.	Os dados de entrada neste caso, será o token de usuário ausente ou inválido.	Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	{ "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais" }	Passou
TC26	Realizar uma requisição **DELETE** à rota **/carrinhos/cancelar-compra** com dados de um **usuário válido**.	Os dados de entrada neste caso, será o token do usuário (estar logado).	Uma resposta com a mensagem: "Registro excluído com sucesso. Estoque dos produtos reabastecido”	{ "message": "Registro excluído com sucesso” }	Falhou
TC27	Realizar uma requisição **DELETE** à rota **/carrinhos/cancelar-compra** com um **usuário com token ausente ou inválido**.	Os dados de entrada neste caso, será o token de usuário ausente ou inválido.	Uma resposta com a mensagem: "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"	{ "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais" }	Passou

5. Local dos testes

Os testes foram realizados em um notebook de sistema operacional Windows 10 Home Single Language baseado em 64 bits.

6. Ferramentas utilizadas

VSCode foi responsável pela implementação da automação dos testes; GoogleDocs para realização dos documentos; XMind para a construção do mapa mental; GitHub para divulgação e versionamento dos testes;

As tecnologias utilizadas para a implementação dos testes foram o Cypress v9.7.0, Faker v5.5.3, Ajv v8.11.0, Mocha v10.0.0, e mochawesome v7.1.3

7. Cronograma

O projeto foi dividido em três fases: planejamento, produção e execução dos testes. Para melhor visualização das atividades, foi desenvolvido um diagrama de Gantt.

X	08/06	09/06	10/06	13/06	14/06	15/06	17/06	20/06	21/06	22/06
Planejamento										

Produção			
Execução			

8. Prioridades

A principal prioridade da API Serverest é a realização de compras, portanto as prioridades desta API são: rota /usuários (cadastrar e deletar usuários), rota /login (realizar login), rota /produtos (CRUD de produtos) e rota /carrinhos (criar carrinhos, concluir e cancelar compras).

9. Candidatos à automação

Os testes candidatos à automação desta API foram todas as funcionalidades que foram priorizadas, são elas: rota /usuários (cadastrar e deletar usuários), rota /login (realizar login) e também seus cenários de exceção, rota /produtos (CRUD de produtos) e rota /carrinhos (criar carrinhos, concluir e cancelar compras).

A tabela a seguir mostra o principal fluxo da API que também foi automatizado o fluxo foi descrito nos testes como “fluxoCompraSucesso.spec.js”

Caso de teste	**Descrição**
TC18	Realizar uma requisição **POST** à rota **/usuarios** com dados de um usuário **válido**. (Criando usuário)
TC01	Realizar uma requisição **POST** à rota **/login** com dados de um usuário **válido**. (Realizando login)
TC08	Realizar uma requisição **POST** à rota **/produtos** com dados de um produto **válido**.(Criando produto)
TC21	Realizar uma requisição **POST** à rota **/carrinhos** com dados de um produto válido adicionando ele ao carrinho.(Criando carrinho e adicionando produtos à ele)
TC24	Realizar uma requisição **DELETE** à rota **/carrinhos/concluir-compra** com dados de um **usuário válido**. (Concluindo compra)
TC12	Realizar uma requisição **DELETE** à rota **/produtos/{_id}** com dados de um **produto válido**. (Deletando produto)

10. Relatório de bugs

Neste tópico serão descritos os bugs encontrados durante a execução dos testes da API Serverest.

ID do	**Descrição do bug**	**Como reproduzir o bug**	**Severidade**	**Prioridade**

teste				

TC02	Erro de documentação.
	
	A documentação da API aponta que o status code desse caso de teste é 400, mas ao realizar a requisição temos o status code 401.
		Realizar uma requisição **POST** à rota **/login** com dados de um usuário **inválido**.
		
		Inserir os dados:
		{
		“email”: “emailincorreto@qa.com”
		“password”: “senhaincorreta”
		}
			Minor	Baixa

TC03	Erro de documentação.
	
	A documentação da API não possui resposta para esta situação.
		Realizar uma requisição **POST** à rota **/login** com **dados nulos**.
		
		Inserir os dados:
		{
		“email”: null
		“password”: null
		}
			Minor	Baixa

TC04	Erro de documentação.
	
	A documentação da API não possui resposta para esta situação.
		Realizar uma requisição **POST** à rota **/login** com **campos vazios**.
		
		Inserir os dados:
		{
		“email”: “”
		“password”: “”
		}
			Minor	Baixa

TC05	Erro na validação do contrato. A documentação da API à rota /produtos com método GET, possui um erro. Onde deveria ser “produtos” está “usuarios”.	Realizar uma requisição **GET** à rota **/produtos, validando seu contrato.**	Minor	Baixa
TC26	Erro de documentação. A documentação da API espera o resultado “Registro excluído com sucesso. Estoque dos produtos reabastecido” mas obtém o resultado "Registro excluído com sucesso”	Realizar uma requisição **DELETE** à rota **/carrinhos/cancelar-compra** com dados de um **usuário válido**.	Minor	Baixa