API - Serverest

Kaio Max Lira Pereira
Trilha Logical Forest - Cypress - Sprint 6

1- Introdução

API ServeRest simula um e-commerce muito utilizado para estudos, foi escolhida para ter algumas das suas rotas candidatas a automação. O plano de teste foi criado com intuito de planejar e orientar todos os testes a serem feitos na *API ServeRest,*

2- Objetivo

Este documento tem como objetivo apresentar as funcionalidades da *API* que estão com funcionamento correto e incorreto.

3- Escopo

A *API ServeRest* passará por alguns testes para que haja a validação das rotas do fluxo prioritário, onde haverá a verificação de testes positivos, negativos e possíveis bugs, assim automatizando os testes do fluxo prioritário.

4- Ferramentas utilizadas

Neste tópico estão listadas todas as ferramentas utilizadas durante todo o processo de teste da *API Serverest.* Para criação de documentos e alocação foi utilizado o *Google Docs* e ferramentas disponibilizadas pelo mesmo, *VSCode* foi o responsável por toda a produção de códigos, junto com a extensão *thunder Client* que foi usada para efetuar testes de unidade. *XMind* utilizado na construção do mapa mental, *Git* e *GitHub* usados no versionamento de códigos e divulgação.

5- Local dos testes

Os teste foram realizados em uma máquina *HP* com sistema operacional *Linux Ubuntu* 20.04

6- Riscos

Neste tópico será apresentado e listado os possíveis riscos que possam interferir, atrasar ou impedir o andamento das tarefas e atividades para a conclusão do projeto:

- Uma possível falta de energia ou queda de internet devido a temporais ou imprevistos.
- A perda inesperada de um equipamento de uso de um dos integrantes da equipe.
- Conflito de datas nas agendas dos integrantes da equipe.
- Problemas de saúde que possam vir a afligir um dos integrantes da equipe.

7- Prioridades

A *API ServeRest* deve ter como prioridade a conclusão de compra dos usuários tornando assim seu fluxo prioritário o login, cadastro, *CRUD* de produtos, criar um carrinho e concluir compra.

8- Teste candidatos a automação

Cadastro dos usuários, login, *CRUD* de produtos, criação de carrinho e permitir a conclusão da compra.

9- Pessoas envolvidas

Papel	**Responsabilidades**	**Integrantes**
Gerente de teste	● Supervisiona o ambiente de teste	● Kaio Max
Analista de teste	● Identifica e prioriza os testes	● Thiago Bento
Testador	● Executar os testes ● Registrar os resultados ● Documentar as solicitações de mudança	● Paulo Henrique

10- Casos de teste

Casos de teste	**Endpoint**	**Descrição**	**Passos**	**Resultado Esperado**
TC01	POST /login	Deve realizar o login do usuário com sucesso.	Enviar uma requisição **POST** no endpoint/login	Retornar uma mensagem “Login realizado com sucesso”
TC02	POST /login	Deve tentar realizar login do usuário.	Enviar uma requisição **POST** no endpoint/login	Retornar uma mensagem “Email e/ou senha inválidos”
TC03	POST /usuarios	Deve cadastrar usuário com sucesso.	Enviar uma requisição **POST** no endpoint/usuario	Retornar uma mensagem “Cadastro realizado com sucesso” junto o id do usuário.

TC04	POST /usuarios	Deve tentar cadastrar usuário.	Enviar uma requisição **POST** no endpoint/usuario	Retornar uma mensagem “Este email já está sendo usado”
TC05	GET /usuarios/{_id}	Deve buscar os usuários por id.	Enviar uma requisição **GET** no endpoint /usuarios/{_id}	Retornar uma mensagem com “nome, email, password, administrador, id”
TC06	GET /produtos	Deve buscar todos os produtos.	Enviar uma requisição **GET** no endpoint /produtos	Retorna uma mensagem com “nome preço, descrição, quantidade, id”
TC07	POST /produtos	Deve realizar o cadastro do produto com sucesso.	Enviar uma requisição **POST** no endpoint/produtos	Retornar uma mensagem “Cadastro realizado com sucesso” junto o id do produto.
TC08	GET /produtos/{_id}	Deve buscar os produtos por id.	Enviar uma requisição **GET** no endpoint /produtos/{_id}	Retornar uma mensagem com “nome preço, descrição, quantidade, id”
TC09	PUT /produtos/{_id}	Deve alterar produto	Enviar uma requisição **PUT** no endpoint /produtos/{_id}	Retornar uma mensagem com “Registro alterado com sucesso”
TC10	DELETE /produtos/{_id}	Deve excluir produtos por id	Enviar uma requisição **DELETE** no endpoint /produtos/{_id}	Retornar uma mensagem com “Registro excluído com sucesso”

TC11	POST /carrinhos	Deve cadastrar um carrinho	Enviar uma requisição **POST** no endpoint /carrinhos	Retornar uma mensagem com “Cadastro realizado com sucesso”
TC12	POST /carrinhos/concluir-compra	Deve excluir carrinho	Enviar uma requisição **DELETE** no endpoint /carrinhos/concluir-compra	Retornar uma mensagem com “Registro excluído com sucesso”
TC13	POST /carrinhos/cancelar-compra	Deve excluir carrinho	Enviar uma requisição **DELETE** no endpoint /carrinhos/cancelar-compra	Retornar uma mensagem com “Registro excluído com sucesso”

11- Relatório de bugs

Id	Descrição	Cenário	Prioridade
TC06	Na documentação temos uma resposta diferente da que é requisitada	Enviar uma requisição **GET** no endpoint /produtos	Baixa