**PLANO DE TESTE REFERENTE A API SERVEREST**

Gustavo Henrique Siqueira da Silva

1. **Introdução**

Esse plano de teste descreve quais os cenários de testes que estarão sendo levados em consideração para a API ServeRest, que representa um E-commerce. A abordagem adotada irá cobrir os seguintes fluxos: compra, desistência de compra e cadastro de produto. Além disso, cobrirá também os métodos GET, PUT, POST e DELETE.

1. **Objetivos**

A API ServeRest simula um e-commerce com o objetivo de servir de estudos de testes de API. Nela existem rotas relacionadas a login, usuários, produtos e carrinhos.

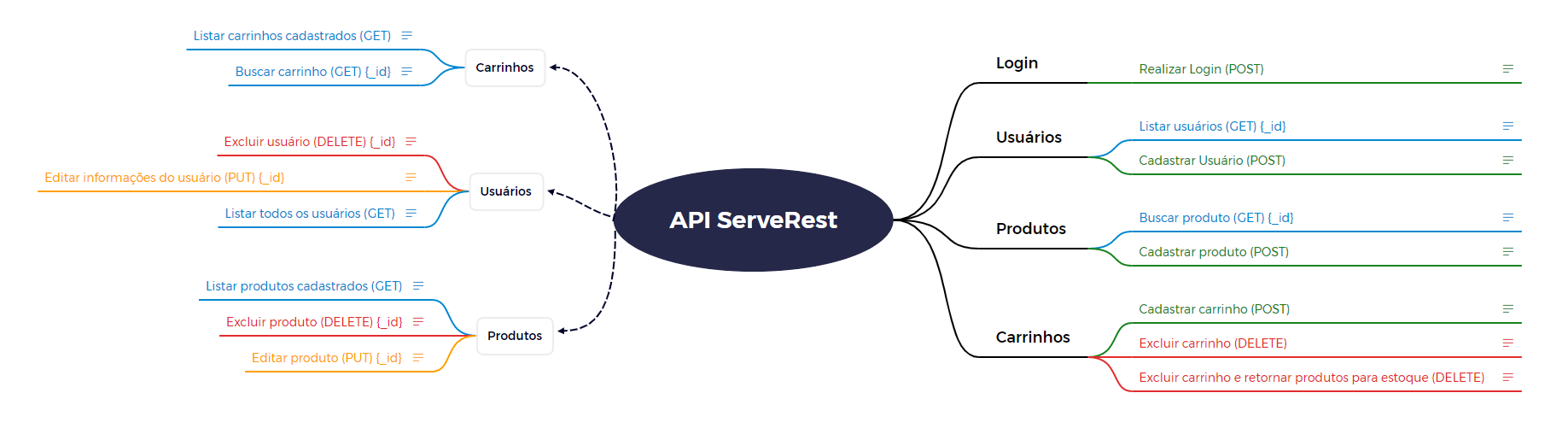
1. **Escopo**

Nesse documento serão apresentados os seguintes pontos:

* Mapa mental da API;
* Descrição das estratégias de testes que serão utilizadas;
* Identificação do que será prioridade;
* Identificação do que será automatizado;
* Os endpoints que serão testados são: Login, usuários, produtos e carrinhos;
* Teste das principais funções da API, sendo divididas em duas pastas, uma para as operações positivas, que são as que durante os testes espera-se que as requisições sejam realizadas e outra para as operações negativas, onde espera-se que as operações não sejam realizadas ou seja executada alguma outra ação;

Responsável pela execução dos testes: Gustavo Henrique Siqueira da Silva.

1. **Mapa mental**



**\*\*OBS: para melhor visualização do mapa mental, favor dar dois cliques no arquivo abaixo\*\***



1. **Suíte de casos de teste**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URL API (Web):** | [**https://serverest.dev**](https://serverest.dev/) | **Informação que vem antes de cada endpoint.** | | |
| **URL API (Local):** | [**http://localhost:3000**](http://localhost:3000/) |
| **Positivo** | | | | |
| **Caso de teste** | **Pré-requisitos** | **Procedimento** | **Resultados esperados** | **Pós-condição** |
| **CT 1 - Cadastrar usuário** | Ter um e-mail que não esteja sendo utilizado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/usuarios**, informando no corpo da requisição o nome, email, password e o booleano true para o campo administrador caso seja um usuário administrador. | É retornada uma mensagem informando a realização do cadastro, junto com o id do usuário. | Conclusão do cadastro. |
| **CT 2 - Login** | Ter realizado o cadastro. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/Login**, informando no corpo da requisição o email e o password. | É retornada uma mensagem informando sucesso ao realizar login, juntamente com o campo Authorization, que vai conter um token do tipo Bearer. | Definir na aba Authorization do Postman o tipo de autorização como Bearer e inserir o token que foi gerado na response (sem o nome Bearer). |
| **CT 3 - Cadastrar produto** | Ter realizado o cadastro de usuário, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/produtos**, preenchendo no corpo da requisição os campos: Nome, preco, descricao e quantidade. Sendo os campos preco e quantidade do tipo Number e os demais do tipo String. | É retornada uma mensagem de sucesso no cadastro, juntamente com o ID do produto cadastrado. | Conclusão do cadastro do produto e é gerado o ID. |
| **CT 4 - Cadastrar Carrinho** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/carrinhos**, preenchendo no corpo da requisição dentro do array chamado produtos, uma lista contendo os ID's dos produtos selecionados e suas respectivas quantidades. | É retornada uma mensagem de sucesso no cadastro, junto com o ID do carrinho. | Conclusão do cadastro do carrinho sendo gerado o ID. |
| **CT 5 - Excluir carrinho - Concluir compra** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter um produto e um carrinho cadastrados. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **DELETE**, para o endpoint **/carrinhos/concluir-compra.** | **Positivo:** Mensagem informando que o registro foi excluído com sucesso. **Negativo:** Mensagem informando que não foi encontrado carrinho para o usuário, ou, token de acesso ausente, inválido ou expirado. | Exclusão do carrinho e redução do estoque de produtos que estava no carrinho. |
| **CT 6 - Excluir carrinho - Devolver estoque** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter um produto e um carrinho cadastrados. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **DELETE**, para o endpoint **/carrinhos/cancelar-compra.** | **Positivo:** Mensagem informando que o registro foi excluído com sucesso. **Negativo:** Mensagem informando que não foi encontrado carrinho para o usuário, ou, token de acesso ausente, inválido ou expirado. | Exclusão do carrinho e devolução dos produtos que estavam no carrinho ao estoque. |
| **CT 7 - Editar produto {\_ID}** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado e possuir o ID. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **PUT**, para o endpoint **/produtos/{\_ID}**, preenchendo o ID na URL e no corpo da requisição as informações a serem alteradas nos respectivos campos. Nome, preco, descricao, quantidade, sendo os campos preco e quantidade do tipo INT e os demais do tipo string. Os campos que não forem ser alterados, basta repetir a informação que já estava. | Mensagem informando que o registro foi alterado com sucesso. Ou, que foi cadastrado com sucesso, no caso do produto não estar cadastrado. | Alteração realizada com sucesso. |
| **CT 8 - Listar produtos cadastrados** | Não possui | Envia uma requisição com método **GET**, para o endpoint **/produtos**. | É retornada uma lista com a quantidade de produtos cadastrados e as informações referente a cada um, nome, preço, descrição, quantidade e id. | Consulta realizada com sucesso. |
| **CT 9 - Excluir produto {\_ID}** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado. | Envia uma requisição com método **DELETE**, para o endpoint **/Produtos/{\_ID}**, informando na url o ID do produto a ser excluído. | É retornada uma mensagem informando que o produto foi excluído. | Exclusão realizada. |
| **CT 10 - Editar usuário {\_ID}** | É preciso que o usuário esteja cadastrado e ter o ID do usuário. | Envia uma requisição com método **PUT**, para o endpoint **/usuarios**, preenchendo no corpo da requisição as novas informações nos campos, nome, email, password, administrador. Administrador é um valor booleano, os demais são do tipo String. | É retornada uma mensagem informando que o registro foi alterado com sucesso. Ou, uma mensagem informando que o cadastro foi realizado com sucesso, caso seja um usuário que não esteja cadastrado na documentação. | Alteração realizada com sucesso. |
| **CT 11 - Listar usuários cadastrados** | Não possui | Envia uma requisição com método **GET**, para o endpoint **/usuarios**. | É retornada uma lista com a quantidade de usuários cadastrados e as informações referente a cada um, nome, email, senha, nível de autorização e id. | Consulta realizada com sucesso. |
| **CT 12 - Excluir Usuário {\_ID}** | Ter o ID do usuário a ser excluído | Envia uma requisição com método **DELETE**, para o endpoint **/usuarios/{\_ID}**, informando na URL da API o ID do usuário. | É retornada uma mensagem informando que o registro foi excluído. | Exclusão do usuário. |
| **Negativo** | | | | |
| **Caso de teste** | **Pré-requisitos** | **Procedimento** | **Resultados esperados** | **Pós-condição** |
| **CT 13 - Cadastrar usuário com e-mail inválido** | O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/usuarios**, informando no corpo da requisição o nome, email em um formato desconhecido, password e o booleano true para o campo administrador caso seja um usuário administrador. | É retornada uma mensagem informando que o email está inválido. | Operação não realizada. |
| **CT 14 - Cadastrar usuário sem informar senha** | Ter um e-mail que não esteja sendo utilizado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/usuarios**, informando no corpo da requisição o nome, email o campo password como uma string vazia e o booleano true para o campo administrador caso seja um usuário administrador. | É retornada uma mensagem informando que o campo password não pode ficar em branco. | Operação não realizada. |
| **CT 15 - Cadastrar usuário (e-mail e senha em branco)** | O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/usuarios**, informando no corpo da requisição o nome, email como string vazia, password como uma string vazia e o booleano true para o campo administrador caso seja um usuário administrador. | É retornada uma mensagem informando que o campo password e e-mail não podem ficar em branco. | Operação não realizada. |
| **CT 16 - Login com usuário inválido** | O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/login**, informando no corpo da requisição o email que estará no formato inválido, sem @ ou ponto, por exemplo, e o password. | É retornada uma mensagem informando que o email deve ser um email válido. | Login não realizado. |
| **CT 17 - Login sem usuário e senha** | O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/login**, informando no corpo da requisição o email e o password como strings vazias. | É retornada uma mensagem informando que o campo password e e-mail não podem ficar em branco. | Login não realizado. |
| **CT 18 - Login sem senha** | O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/login**, informando no corpo da requisição o email e o password, sendo que somente o campo password ficará como uma string vazia. | É retornada uma mensagem informando que o campo password não pode ficar em branco. | Login não realizado. |
| **CT 19 - Cadastrar produto (preço negativo)** | Ter realizado o cadastro de usuário, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/produtos**, preenchendo no corpo da requisição os campos: Nome, preco, que deve ser informado um valor negativo, descricao e quantidade. Sendo os campos preco e quantidade do tipo Number e os demais do tipo String. | É retornada uma mensagem informando que o preço deve ser um número positivo. | Operação não concluída. |
| **CT 20 - Cadastrar produto (quantidade negativa)** | Ter realizado o cadastro de usuário, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/produtos**, preenchendo no corpo da requisição os campos: Nome, preco, descricao e quantidade, que deve ser informado um valor negativo. Sendo os campos preco e quantidade do tipo Number e os demais do tipo String. | É retornada uma mensagem informando que a quantidade deve ser um número maior ou igual a zero. | Operação não concluída. |
| **CT 21 - Cadastrar produto (usuário sem autorização)** | Ter realizado o cadastro de usuário, estar logado na plataforma e com o token já definido como Bearer. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/produtos**, preenchendo no corpo da requisição os campos: Nome, preco, descricao e quantidade. Sendo os campos preco e quantidade do tipo Number e os demais do tipo String. | É retornada uma mensagem informando que a rota é exclusiva para administradores | Operação não concluída. |
| **CT 22 - Cadastrar carrinho ID de produto inválido** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/carrinhos**, preenchendo no corpo da requisição dentro do array chamado produtos, uma lista contendo os ID's dos produtos selecionados, que não estarão cadastrados no sistema e suas respectivas quantidades. | É retornada uma mensagem informando que o produto não foi encontrado, junto com o ID do produto que foi inserido e a quantidade do mesmo no carrinho. | Operação não concluída. |
| **CT 23 - Cadastrar carrinho com quantidade de produtos insuficiente no estoque** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/carrinhos**, preenchendo no corpo da requisição dentro do array chamado produtos, uma lista contendo os ID's dos produtos selecionados e uma quantidade maior que a disponível em estoque. | É retornada uma mensagem informando que o produto não possui quantidade suficiente e informa a quantidade disponível em estoque e a quantidade solicitada. | Operação não concluída. |
| **CT 24 - Cadastrar mais de um carrinho** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e já ter um carrinho cadastrado no usuário que está logado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **POST**, para o endpoint **/carrinhos**, preenchendo no corpo da requisição dentro do array chamado produtos, uma lista contendo os ID's dos produtos selecionados e suas respectivas quantidades. | É retornada uma mensagem informando que não é possível ter mais de um carrinho cadastrado. | Operação não concluída. |
| **CT 25 - Editar usuário {\_ID} com e-mail inválido** | É preciso ter o ID do usuário. | Envia uma requisição com método **PUT**, para o endpoint **/usuarios**, preenchendo no corpo da requisição as novas informações nos campos, nome, email, que estará no formato inválido, sem @ ou ponto, por exemplo, password, administrador. Administrador é um valor booleano, os demais são do tipo String. | É retornada uma mensagem informando que o email deve ser válido. | Alteração não realizada. |
| **CT 26 - Editar usuário {\_ID} sem informar senha** | É preciso ter o ID do usuário. | Envia uma requisição com método **PUT**, para o endpoint **/usuarios**, preenchendo no corpo da requisição as novas informações nos campos, nome, email, password, que será uma string vazia e administrador, que é um valor booleano, os demais são do tipo String. | É retornada uma mensagem informando que o campo password não pode ficar em branco. | Alteração não realizada. |
| **CT 27 - Editar usuário {\_ID} e-mail e senha em branco** | É preciso ter o ID do usuário. | Envia uma requisição com método **PUT**, para o endpoint **/usuarios**, preenchendo no corpo da requisição as novas informações nos campos, nome, email, que será uma string vazia, password, que também será uma string vazia e administrador, que é um valor booleano, os demais são do tipo String. | É retornada uma mensagem informando que os campos de email e senha não podem ficar em branco. | Alteração não realizada. |
| **CT 28 - Editar produto {\_ID} (quantidade negativa)** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado e possuir o ID. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **PUT**, para o endpoint **/produtos/{\_ID}**, preenchendo o ID na URL e no corpo da requisição as informações a serem alteradas nos respectivos campos. Nome, preco, descricao, quantidade, sendo os campos preco e quantidade do tipo INT e os demais do tipo string. Os campos que não forem ser alterados, basta repetir a informação que já estava. | É retornada uma mensagem informando que aquantidade deve ser maior ou igual a 0. | Alteração não realizada. |
| **CT 29 - Editar produto {\_ID} (ID inexistente)** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **PUT**, para o endpoint **/produtos/{\_ID}**, preenchendo um ID que não exista na URL e no corpo da requisição as informações a serem alteradas nos respectivos campos. Nome, preco, descricao, quantidade, sendo os campos preco e quantidade do tipo INT e os demais do tipo string. Os campos que não forem ser alterados, basta repetir a informação que já estava. | É retornada uma mensagem de realização de cadastro do produto, junto com o ID. | É realizado um cadastro ao invés de alteração. |
| **CT 30 - Editar produto {\_ID} (preço negativo)** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **PUT**, para o endpoint **/produtos/{\_ID}**, preenchendo um ID na URL e no corpo da requisição as informações a serem alteradas nos respectivos campos. Nome, preco, que deverá ser um valor negativo, descricao, quantidade, sendo os campos preco e quantidade do tipo INT e os demais do tipo string. Os campos que não forem ser alterados, basta repetir a informação que já estava. | É retornada uma mensagem informando que o preço deve ser um número positivo. | Alteração não realizada. |
| **CT 31 - Editar produto {\_ID} (sem nome)** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado. O corpo da request deve ser do tipo Json. | Envia uma requisição com método **PUT**, para o endpoint **/produtos/{\_ID}**, preenchendo um ID na URL e no corpo da requisição as informações a serem alteradas nos respectivos campos. Nome, que deve ser uma string em branco, preco, descricao, quantidade, sendo os campos preco e quantidade do tipo INT e os demais do tipo string. Os campos que não forem ser alterados, basta repetir a informação que já estava. | É retornada uma mensagem informando que o nome não pode ficar em branco. | Alteração não realizada. |
| **CT 32 - Excluir produto {\_ID} (usuário sem autorização)** | Ter realizado o cadastro, estar logado na plataforma, com o token já definido como Bearer e ter usuário que seja administrador do sistema. Ter um produto cadastrado. | Envia uma requisição com método **DELETE**, para o endpoint **/Produtos/{\_ID}**, informando na url o ID do produto a ser excluído. | É retornada uma mensagem informando que a rota é exclusiva para administradores | Exclusão não realizada. |

1. **Estratégia de testes**

* **Teste de Funcionalidade**

**Objetivo** 🡪 Garantir que o login seja realizado pelo usuário e realizar validações das diferentes funções.

**Técnica 🡪** Executar testes com várias requisições ou em fluxo, preenchendo informações válidas e inválidas para verificar se ao inserir dados válidos o resultado esperado seja atingido e se ao inserir dados inválidos serão retornadas as respectivas mensagens de erro.

**Critério de encerramento 🡪** Todos os testes terem sido executados e validados, conforme esperado.

* **Teste de integração**

**Objetivo 🡪** Garantir o funcionamento dos métodos das requisições expostos no Swagger, além de verificar se o seu comportamento está dentro do esperado.

**Técnica** 🡪 Verificar se as requisições estão gerando resposta normalmente e se o comportamento está de acordo com as respostas.

**Critério de encerramento 🡪** Validar se houve alterações após a requisição ter sido executada.

* **Teste de controle de acesso**

**Objetivo 🡪** Garantirque o usuário faça somente aquilo que lhe foi autorizado.

**Técnica** 🡪 Criação de usuário administrador para realização de requisições que só o administrador pode executar e através de pré-requests, criação de usuário não administrador verificando se as suas funções disponíveis estão sendo executadas corretamente e verificando também os cenários em que ele não tem acesso.

**Critério de encerramento** 🡪 Cada usuário estiver autorizado a executar somente aquilo que lhe foi permitido.

1. **Candidatos a automação**

Para essa API, foram automatizados todos os casos de teste, visando atingir o objetivo do programa de bolsas.

* **CT 1 - Cadastrar usuário**
* **CT 2 - Login**
* **CT 3 - Cadastrar produto**
* **CT 4 - Cadastrar Carrinho**
* **CT 5 - Excluir carrinho - Concluir compra**
* **CT 6 - Excluir carrinho - Devolver estoque**
* **CT 7 - Editar produto {\_ID}**
* **CT 8 - Listar produtos cadastrados**
* **CT 9 - Excluir produto {\_ID}**
* **CT 10 - Editar usuário**
* **CT 11 - Listar usuários cadastrados**
* **CT 12 - Excluir Usuário {\_ID}**
* **CT 13 - Cadastrar usuário com e-mail inválido**
* **CT 14 - Cadastrar usuário sem informar senha**
* **CT 15 - Cadastrar usuário (e-mail e senha em branco)**
* **CT 16 - Login com usuário inválido**
* **CT 17 - Login sem usuário e senha**
* **CT 18 - Login sem senha**
* **CT 19 - Cadastrar produto (preço negativo)**
* **CT 20 - Cadastrar produto (quantidade negativa)**
* **CT 21 - Cadastrar produto (usuário sem autorização)**
* **CT 22 - Cadastrar carrinho ID de produto inválido**
* **CT 23 - Cadastrar carrinho com quantidade de produtos insuficiente no estoque**
* **CT 24 - Cadastrar mais de um carrinho**
* **CT 25 - Editar usuário {\_ID} com e-mail inválido**
* **CT 26 - Editar usuário {\_ID} sem informar senha**
* **CT 27 - Editar usuário {\_ID} e-mail e senha em branco**
* **CT 28 - Editar produto {\_ID} (quantidade negativa)**
* **CT 29 - Editar produto {\_ID} (ID inexistente)**
* **CT 30 - Editar produto {\_ID} (preço negativo)**
* **CT 31 - Editar produto {\_ID} (sem nome)**
* **CT 32 - Excluir produto {\_ID} (usuário sem autorização)**

1. **Ferramentas**

|  |  |  |
| --- | --- | --- |
| **FERRAMENTA** | **FINALIDADE** | **VERSÃO** |
| **Microsoft Word  Microsoft Excel** | **Gerenciamento do projeto** | **1804 (16.0.9226.2156)** |
| **Postman** | **Teste de API  Automação de testes** | **9.19.3 Desktop Platform Version 9.19.0** |
| **Xmind** | **Desenvolvimento de mapa mental** | **202204260729** |
| **Notion** | **Organização do cronograma de montagem do plano de testes** | **2.0.28** |
| **Jira** | **Acompanhamento dos testes/Gerenciamento das tarefas** | **8.22.2** |
| **NPM** | **Executar API localmente** | **8.11.0** |
| **NodeJS** | **Executar API localmente** | **16.15.1** |

1. **Observações complementares**

* O token da API funciona somente por 10 minutos, para gerar um novo basta realizar um novo login.
* Cada usuário só pode ter no máximo 1 carrinho cadastrado.
* Para elaboração do objetivo, escopo e mapa mental, foram consultados:
  + Marcelo Joaquim;
  + Danilo da Silva;
* Para elaboração da suíte de casos de teste, foi consultado:
  + Ramon Gabriel;