ServeRest

PLANO DE TESTES

LogicalForest 2

Felipe Ripplinger Dupont

SUMÁRIO

1.Introdução

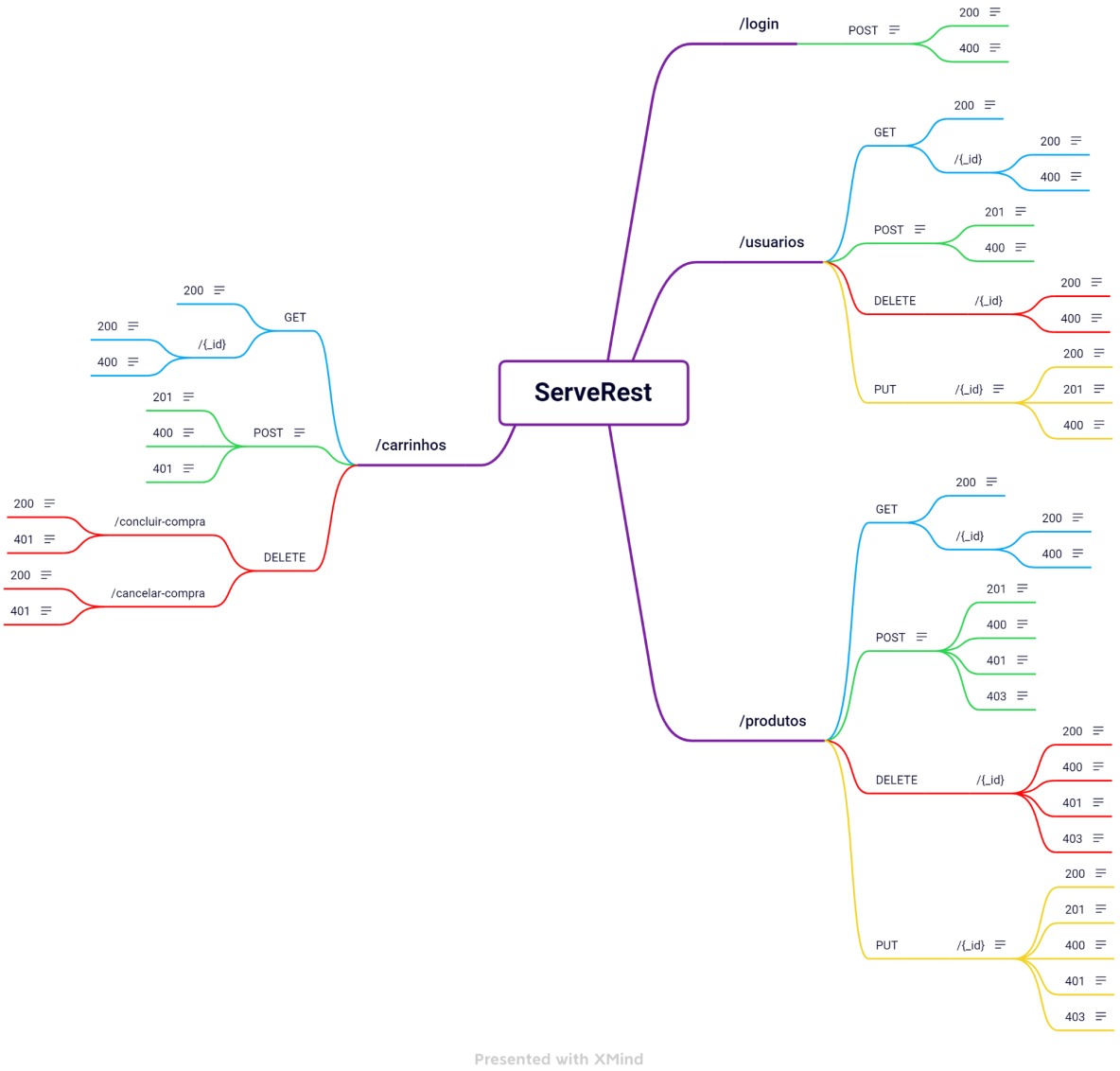
1.1 Objetivos

Bem-vindo ao plano de testes da API ServeRest, os principais objetivos com este plano é selecionar, priorizar e planejar os testes das funcionalidades da ServeRest, uma API gratuita que simula uma loja virtual com intuito de servir de material de estudos de testes.

1.2 Escopo

Os testes serão realizados em todos os endpoints presentes na API ServeRest, almejando um Path Coverage de 100%. Os endpoints são bem definidos e não muito complexos, permitindo um escopo abrangente. Prever os resultados de cada requisição, as mensagens de retorno, status codes e respostas do body fazem parte do planejamento.

1.3 Mapa mental



Link para o [mapa mental na íntegra](#).

2.Técnicas e tipos de testes

2.1 Prioridades

Como já citado, as rotas são bem definidas e não possuem muita complexidade, portanto para garantir o bom funcionamento da API, todas as rotas serão testadas conforme sua funcionalidade. Testes de carga e destrutivos não serão aplicados, focando apenas em testes funcionais.

2.2 Estratégia de teste

A estratégia escolhida para os testes foi a de caixa preta, e apesar de ser uma API pública e ter o código disponível, ela foi tratada como propriedade de terceiros. Os tipos de testes que serão realizados se dividem em dois: regressivos e de fluxo, o primeiro garante individualmente que as funcionalidades continuem a funcionar adequadamente após alterações no sistema e o segundo simula fluxos e comportamentos de usuários do sistema.

2.3 Informações gerais referentes

Status code presentes na documentação

CODE	TEXTO	DESCRIÇÃO
200	OK	Sucesso.
201	Created	Cadastrado com sucesso.
400	Bad Request	Inválido ou não pode ser realizado.
401	Unauthorized	Token ausente, inválido ou expirado.
403	Forbidden	Rota exclusiva para administradores (administrador = true).
405	Method Not Allowed	Não é possível realizar GET em /{}. Acesse https://serverest.dev para ver as rotas disponíveis e como utilizá-las.

Métodos de autenticação presente

Bearer Token

Formato de retorno e envio padrão

JSON

2.4 Suítes de casos de teste

Seção 01 - Testes Positivos

Todos os testes devem retornar um status code 2XX

GET

POST

DELETE

PUT

Login

CTP01 - Realizar login
Descrição: Deve realizar um login com sucesso.
Etapas do Teste: 1.Obter o e-mail e senha de um usuário cadastrado e colocar no body da requisição. 2.Realizar requisição na rota /login com o método POST.

Padrão do Body:
{
"email": "fulano@qa.com",
"password": "teste"
}

Padrão da resposta esperada: 200 - Login realizado com sucesso

{
"message": "Login realizado com sucesso",
"authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImZ1bGFub0BxYS5jb20iLCJwYXNzd29yZCI6InRlc3RlIiwiaWF0IjoxNTg5NzU4NzQ2LCJleHAiOjE1ODk3Njg3NDZ9.B6TASHV8k9xBerz4NSeFBlAZGSDhZlqESt767M0567I"
}

Usuários

CTP02 - Listar usuários cadastrados

Descrição:
Deve buscar todos os usuários cadastrados.

Etapas do Teste:
1.Realizar requisição na rota /usuarios com o método GET.

Padrão da resposta esperada:
200 - Lista de usuários


```
{
  "quantidade": 1,
  "usuarios": [
    {
      "nome": "Fulano da Silva",
      "email": "beltrano@qa.com.br",
      "password": "teste",
      "administrador": "true",
      "_id": "0uxuPY0cbmQhpEz1"
    }
  ]
}
```


CTP03 - Cadastrar usuário

Descrição:
Deve cadastrar um novo usuário com sucesso.

Etapas do Teste:
1.Obter um nome, e-mail não cadastrado e uma senha, colocar isso no body da requisição.
2.Realizar requisição na rota /usuarios com o método POST.

Padrão do Body:

```
{
  "nome": "Fulano da Silva",
  "email": "beltrano@qa.com.br",
  "password": "teste",
  "administrador": "false"
}
```

Padrão da resposta esperada:
201 - Cadastro com sucesso


```
{
  "message": "Cadastro realizado com sucesso",
  "_id": "jogfODIIXsqxNFS2"
}
```

CTP04 - Cadastrar usuário administrador

Descrição:
Deve cadastrar um novo usuário administrador com sucesso.

Etapas do Teste:
1.Obter um nome, e-mail não cadastrado, uma senha e garantir que a chave administrador tenha o valor "true", colocar isso no body da requisição.
2.Realizar requisição na rota /usuarios com o método POST.

Padrão do Body:

```
{
  "nome": "Fulano da Silva",
  "email": "beltrano@qa.com.br",
  "password": "teste",
  "administrador": "true"
}
```

Padrão da resposta esperada:
201 - Cadastro com sucesso

```
{
  "message": "Cadastro realizado com sucesso",
  "_id": "jogfODIlXsqxNFS2"
}
```

CTP05 - Buscar usuário por ID

Descrição:
Deve buscar um usuário com sucesso.

Etapas do Teste:
1.Obter o ID de um usuário cadastrado.
2.Realizar requisição na rota /usuarios/{_id} com o método GET.

Padrão da resposta esperada:
200 - Usuário encontrado

```
{
  "nome": "Fulano da Silva",
  "email": "beltrano@qa.com.br",
  "password": "teste",
  "administrador": "true",
  "_id": "0uxuPY0cbmQhpEz1"
}
```

CTP06 - Editar usuário

Descrição:
Deve editar um usuário com sucesso.

Etapas do Teste:
1.Obter o ID de um usuário cadastrado.

2.Alterar um ou mais valores de chaves pertencentes ao usuário do ID e colocar isto no body.
2.Realizar requisição na rota /usuarios/{_id} com o método POST.

Padrão do Body:

```
{
  "nome": "Fulano da Silva Sauro",
  "email": "beltranociclano@qa.com.br",
  "password": "teste123",
  "administrador": "true"
}
```

Padrão da resposta esperada:
200 - Alterado com sucesso

```
{
  "message": "Registro alterado com sucesso"
}
```

CTP07 - Excluir usuário

Descrição:
Deve excluir um usuário com sucesso.

Etapas do Teste:
1.Obter o ID de um usuário cadastrado.
2.Realizar requisição na rota /usuarios/{_id} com o método DELETE.

Padrão da resposta esperada:
200 - Registro excluído com sucesso

```
{
  "message": "Registro excluído com sucesso"
}
```

CTP08 - Salvar um usuário em um arquivo JSON

Descrição:
Deve buscar e salvar um usuário em um arquivo JSON

Etapas do Teste:
1.Realizar requisição na rota /usuarios com o método GET.
2.Selecionar o primeiro usuário da lista e salvar suas informações em um arquivo JSON.

Padrão da resposta esperada:
200 - Lista de usuários

```
{
  "nome": "Fulano da Silva",
  "email": "beltrano@qa.com.br",
  "password": "teste",
  "administrador": "true",
```

```
  "_id": "0uxuPY0cbmQhpEz1"
}
```


CTP09 - Logar com um usuário em um arquivo JSON

Descrição:
Deve logar com um usuário de um arquivo JSON.

Etapas do Teste:
1.Obter o e-mail e senha de um usuário do usuário salvo previamente em um arquivo JSON..
2.Realizar requisição na rota /login com o método POST.

Padrão do Body:


```
{
  "email": "fulano@qa.com",
  "password": "teste"
}
```


Padrão da resposta esperada:
200 - Login realizado com sucesso


```
{
"message": "Login realizado com sucesso",
"authorization": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImZ1bGFub0BxYS5jb20iLCJwYX
Nzd29yZCI6InRlc3RlIiwiaWF0IjoxNTg5NzU4NzQ2LCJleHAiOjE1ODk3Njg3NDZ9.B6TASH
V8k9xBerz4NSeFBlAZGSDhZlqESt767M0567I"
}
```


Produtos

CTP10 - Listar produtos cadastrados

Descrição:
Deve buscar todos os produtos cadastrados.

Etapas do Teste:
1.Realizar requisição na rota /produtos com o método GET.

Padrão da resposta esperada:
200 - Lista de produtos

Resultado esperado:


```
{
  "quantidade": 2,
  "produtos": [
    {
      "nome": "Logitech MX Vertical",
      "preco": 470,
      "descricao": "Mouse",
      "quantidade": 381,
```

```
    "_id": "BeeJh5lz3k6kSIzA"
   },
   {
    "nome": "Samsung 60 polegadas",
    "preco": 5240,
    "descricao": "TV",
    "quantidade": 49977,
    "_id": "K6leHdftCeOJj8BJ"
   }
  ]
}
```


CTP11 - Cadastrar produto

Descrição:
Deve cadastrar um novo produto com sucesso.

Etapas do Teste:
1.Logar com sucesso(Usuário administrador).
2.Obter um nome não cadastrado, preço, descrição e quantidade, colocar isso no body da requisição.
3.Realizar requisição na rota /produtos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:


```
{
  "nome": "Logitech MX Vertical",
  "preco": 470,
  "descricao": "Mouse",
  "quantidade": 381
}
```


Padrão da resposta esperada:
201 - Cadastro com sucesso


```
{
  "message": "Cadastro realizado com sucesso",
  "_id": "jogfODIIXsqxNFS2"
}
```


CTP12 - Buscar produto por ID

Descrição:
Deve buscar um produto com sucesso.

Etapas do Teste:
1.Obter o ID de um usuário cadastrado.
2.Realizar requisição na rota /produtos/{_id} com o método GET.

Padrão da resposta esperada:
200 - Produto encontrado

Resultado esperado:

{
  "nome": "Logitech MX Vertical",
  "preco": 470,
  "descricao": "Mouse",
  "quantidade": 381,
  "_id": "BeeJh5lz3k6kSIzA"
}

CTP13 - Editar produto

Descrição:
Deve editar um produto com sucesso.

Etapas do Teste:
1.Logar com sucesso(Usuário administrador).
2.Obter o ID de um produto cadastrado.
3.Alterar um ou mais valores de chaves pertencentes ao produto do ID e colocar isto no body.
4.Realizar requisição na rota /produto /{_id} com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

{
  "nome": "Logitech MX Horizontal",
  "preco": 250,
  "descricao": "Mouse",
  "quantidade": 220
}

Padrão da resposta esperada:
200 - Alterado com sucesso

{
  "message": "Registro alterado com sucesso"
}

CTP14 - Excluir produto

Descrição:
Deve excluir um produto com sucesso.

Etapas do Teste:
1.Logar com sucesso(Usuário administrador).
2.Obter o ID de um produto cadastrado que não esteja em um carrinho.
3.Realizar requisição na rota /produto /{_id} com o método DELETE.

Método de Autenticação:
Bearer Token.

Padrão da resposta esperada:
200 - Registro excluído com sucesso


```
{
  "message": "Registro excluído com sucesso"
}
```


Carrinhos

CTP15 - Listar carrinhos cadastrados

Descrição:
Deve buscar todos os carrinhos cadastrados.

Etapas do Teste:
1.Realizar requisição na rota /carrinhos com o método GET.

Padrão da resposta esperada:
200 - Lista de carrinhos

Resultado esperado:


```
{
  "quantidade": 1,
  "carrinhos": [
    {
      "produtos": [
          {
            "idProduto": "BeeJh5lz3k6kSIzA",
            "quantidade": 1,
            "precoUnitario": 470
          },
          {
            "idProduto": "K6leHdftCeOJj8BJ",
            "quantidade": 2,
            "precoUnitario": 5240
          }
      ],
      "precoTotal": 10950,
      "quantidadeTotal": 3,
      "idUsuario": "oUb7aGkMtSEPf6BZ",
      "_id": "aFOUqntef4iaOwWfg"
    }
  ]
}
```


CTP16 - Cadastrar carrinho

Descrição:
Deve cadastrar um novo carrinho com sucesso.

Etapas do Teste:

1.Logar com sucesso em um usuário que não tenha um carrinho cadastrado(ou excluir carrinho antes de seguir as próximas etapas).
2.Obter um ou mais IDs de produtos cadastrados com quantidade suficiente e colocar isso no body da requisição.
3.Realizar requisição na rota /carrinhos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "produtos": [
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
    },
    {
      "idProduto": "YaeJ455lz3k6kSIzA",
      "quantidade": 3
    }
  ]
}
```

Padrão da resposta esperada:
201 - Cadastro com sucesso

```
{
  "message": "Cadastro realizado com sucesso",
  "_id": "jogfODIIXsqxNFS2"
}
```

CTP17 - Buscar carrinho por ID

Descrição:
Deve buscar um carrinho com sucesso.

Etapas do Teste:
1.Obter o ID de um carrinho cadastrado.
2.Realizar requisição na rota /carrinhos /{_id} com o método GET.

Padrão da resposta esperada:
200 - Produto encontrado

Resultado esperado:

```
{
  "produtos": [
    [
      {
        "idProduto": "BeeJh5lz3k6kSIzA",
        "quantidade": 2,
        "precoUnitario": 470
      },
      {
        "idProduto": "K6leHdftCeOJj8BJ",
```

```
        "quantidade": 1,
        "precoUnitario": 5240
      }
    ]
  ],
  "precoTotal": 6180,
  "quantidadeTotal": 3,
  "idUsuario": "oUb7aGkMtSEPf6BZ",
  "_id": "qbMqntef4iTOwWfg"
}
```


CTP18 - Excluir carrinho e retornar produtos para estoque/Cancelar compra

Descrição:
Deve excluir um carrinho e cancelar a compra com sucesso.

Etapas do Teste:
1.Logar com sucesso em um usuário com um carrinho cadastrado (ou cadastrar um carrinho antes de seguir a próxima etapa).
2.Realizar requisição na rota /carrinhos/cancelar-compra com o método DELETE.

Método de Autenticação:
Bearer Token.

Padrão da resposta esperada:
200 - Registro excluído com sucesso


```
{
  "message": "Registro excluído com sucesso"
}
```


CTP19 - Excluir carrinhos/Concluir Compra

Descrição:
Deve excluir um carrinho e concluir a compra com sucesso.

Etapas do Teste:
1.Logar com sucesso em um usuário com um carrinho cadastrado (ou cadastrar um carrinho antes de seguir a próxima etapa).
2.Realizar requisição na rota /carrinhos/concluir-compra com o método DELETE.

Método de Autenticação:
Bearer Token.

Padrão da resposta esperada:
200 - Registro excluído com sucesso


```
{
  "message": "Registro excluído com sucesso"
}
```

Seção 02 - Testes Negativos

GET
POST
DELETE
PUT

Login

CTN01 - Login sem sucesso
Descrição: O login não deve ser bem sucedido caso o email e/ou senha sejam inválidos.
Etapas do Teste: 1.Obter o e-mail e/ou senha permitidos mas não válidos e colocar no body da requisição . 2.Realizar requisição na rota /login com o método POST.

Padrão do Body:
{
"email": "xxxxxxx@qa.com",
"password": "naologin"
}

Status code: 401 - Unauthorized

Resultado esperado:
{
"message": "Email e/ou senha inválidos"
}

Usuários

CTN02 - Cadastrar usuário sem sucesso
Descrição: Não deve cadastrar um novo usuário caso o email já esteja cadastrado.
Etapas do Teste: 1.Obter um nome, e-mail já cadastrado e uma senha, colocar isso no body da requisição. 2.Realizar requisição na rota /usuarios com o método POST.
Padrão do Body: { "nome": "Fulano da Silva", "email": "beltrano@qa.com.br", "password": "teste",

"administrador": "false" }
Padrão da resposta esperada: 400 - Bad Request

{
"message": "Este email já está sendo usado"
}

CTN03 - Buscar usuário por ID sem sucesso
Descrição: Não deve buscar um usuário com sucesso se o ID enviado não existir.
Etapas do Teste: 1.Obter um ID de usuário permitido, mas não válido. 2.Realizar requisição na rota /usuarios/{_id} com o método GET.
Padrão da resposta esperada: 400- Bad Request

{
"message": "Usuário não encontrado"
}

CTN04 - Excluir usuário sem sucesso
Descrição: Não deve excluir um usuário se o mesmo tiver um carrinho cadastrado.
Etapas do Teste: 1.Obter o ID de um usuário cadastrado e com carrinho cadastrado. 2.Realizar requisição na rota /usuarios/{_id} com o método DELETE.
Padrão da resposta esperada: 400- Bad Request

{
"message": "Não é permitido excluir usuário com carrinho cadastrado",
"idCarrinho": "qbMqntef4iTOwWfg"
}

Produtos

CTN05 - Cadastrar produto sem sucesso (Token ausente)
Descrição: Não deve tentar cadastrar um novo produto caso o token esteja ausente.

Etapas do Teste:
1.Obter um nome não cadastrado, preço, descrição e quantidade, colocar isso no body da requisição.
2.Realizar requisição na rota /produtos com o método POST.

Padrão do Body:
```
{
  "nome": "Logitech MX Vertical",
  "preco": 470,
  "descricao": "Mouse",
  "quantidade": 381
}
```

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN06 - Excluir produto sem sucesso (Token ausente)

Descrição:
Não deve tentar excluir um produto caso o token esteja ausente.

Etapas do Teste:
1.Obter o ID de um produto cadastrado que não esteja em um carrinho.
2.Realizar requisição na rota /produto /{_id} com o método DELETE.

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN07 - Editar produto sem sucesso (Token ausente)

Descrição:
Não deve tentar editar um produto caso o token esteja ausente.

Etapas do Teste:
1.Obter o ID de um produto cadastrado.
2.Alterar um ou mais valores de chaves pertencentes ao produto do ID e colocar isto no body.
3.Realizar requisição na rota /produto /{_id} com o método POST.

Padrão do Body:
```
{
  "nome": "Logitech MX Horizontal",
  "preco": 250,
```

```
  "descricao": "Mouse",
  "quantidade": 220
}
```

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN08 - Cadastrar produto sem sucesso (Já cadastrado)

Descrição:
Não deve cadastrar um novo produto caso o nome do produto já esteja cadastrado.

Etapas do Teste:
1.Logar com sucesso(Usuário administrador).
2.Obter um nome já cadastrado, preço, descrição e quantidade, colocar isso no body da requisição.
3.Realizar requisição na rota /produtos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "nome": "Logitech MX Vertical",
  "preco": 470,
  "descricao": "Mouse",
  "quantidade": 381
}
```

Padrão da resposta esperada:
400- Bad Request

```
{
  "message": "Já existe produto com esse nome"
}
```

CTN09 - Buscar produto por ID sem sucesso

Descrição:
Não deve buscar um produto com sucesso se o ID enviado não existir.

Etapas do Teste:
1.Obter um ID de produto permitido, mas não válido.
2.Realizar requisição na rota /produto /{_id} com o método GET.

Padrão da resposta esperada:
400- Bad Request

```
{
  "message": "Produto  não encontrado"
}
```

CTN10 - Excluir produto sem sucesso (Em carrinho)

Descrição:
Não deve excluir um produto caso o mesmo esteja em um carrinho.

Etapas do Teste:
1.Logar com sucesso(Usuário administrador).
2.Obter o ID de um produto cadastrado que esteja em um carrinho.
3.Realizar requisição na rota /produto /{_id} com o método DELETE.

Método de Autenticação:
Bearer Token.

Padrão da resposta esperada:
400- Bad Request

```
{
  "message": "Não é permitido excluir produto que faz parte de carrinho",
  "idCarrinho": [
    "qbMqntef4iTOwWfg, ylLJY1eaAUC6hyRc"
  ]
}
```

CTN11 - Cadastrar produto sem sucesso (Usuário não administrador)

Descrição:
Não deve cadastrar um novo produto caso o usuário logado não seja administrador.

Etapas do Teste:
1.Logar com sucesso(Usuário não administrador).
2.Obter um nome não cadastrado, preço, descrição e quantidade, colocar isso no body da requisição.
3.Realizar requisição na rota /produtos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "nome": "Logitech MX Vertical",
  "preco": 470,
  "descricao": "Mouse",
  "quantidade": 381
}
```

Padrão da resposta esperada:
403- Forbidden

```
{
```

```
  "message": "Rota exclusiva para administradores"
}
```

CTN12 - Excluir produto sem sucesso (Usuário não administrador)

Descrição:
Não deve excluir um produto caso o usuário logado não seja administrador.

Etapas do Teste:
1.Logar com sucesso(Usuário não administrador).
2.Obter o ID de um produto cadastrado que não esteja em um carrinho.
3.Realizar requisição na rota /produto /{_id} com o método DELETE.

Método de Autenticação:
Bearer Token.

Padrão da resposta esperada:
403- Forbidden

```
{
  "message": "Rota exclusiva para administradores"
}
```

CTN13 - Editar produto sem sucesso (Usuário não administrador)

Descrição:
Não deve editar um produto caso o usuário logado não seja administrador.

Etapas do Teste:
1.Logar com sucesso(Usuário não administrador).
2.Obter o ID de um produto cadastrado.
3.Alterar um ou mais valores de chaves pertencentes ao produto do ID e colocar isto no body.
4.Realizar requisição na rota /produto /{_id} com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "nome": "Logitech MX Horizontal",
  "preco": 250,
  "descricao": "Mouse",
  "quantidade": 220
}
```

Padrão da resposta esperada:
403- Forbidden

```
{
  "message": "Rota exclusiva para administradores"
}
```

Carrinhos

CTN14 - Cadastrar carrinho sem sucesso (Token ausente)

Descrição:
Não deve cadastrar um novo carrinho caso o token esteja ausente.

Etapas do Teste:
1.Obter um ou mais IDs de produtos cadastrados com quantidade suficiente e colocar isso no body da requisição.
2.Realizar requisição na rota /carrinhos com o método POST.

Padrão do Body:

```
{
  "produtos": [
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
    },
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
    }
  ]
}
```

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN15 - Excluir carrinhos/Concluir Compra sem sucesso (Token ausente)

Descrição:
Não deve tentar excluir carrinho e concluir a compra caso o token esteja ausente.

Etapas do Teste:
1.Realizar requisição na rota /carrinhos/concluir-compra com o método DELETE.

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN16 - Excluir carrinho e retornar produtos para estoque/Cancelar compra sem sucesso (Token ausente)

Descrição:
Não deve excluir carrinho e cancelar a compra caso o token esteja ausente.

Etapas do Teste:
1.Realizar requisição na rota /carrinhos/cancelar-compra com o método DELETE.

Padrão da resposta esperada:
401- Unauthorized

```
{
  "message": "Token de acesso ausente, inválido, expirado ou usuário do token não existe mais"
}
```

CTN17 - Buscar carrinho por ID sem sucesso

Descrição:
Não deve buscar um carrinho com sucesso se o ID enviado não existir.

Etapas do Teste:
1.Obter um ID de carrinho permitido, mas não válido.
2.Realizar requisição na rota /carrinhos/{_id} com o método GET.

Padrão da resposta esperada:
400- Bad Request

Resultado esperado:

```
{
  "message": "Carrinho não encontrado"
}
```

CTN18 - Cadastrar carrinho sem sucesso (Produto duplicado)

Descrição:
Não deve cadastrar um novo carrinho caso os produtos incluídos sejam duplicados.

Etapas do Teste:
1.Logar com sucesso em um usuário que não tenha um carrinho cadastrado(ou excluir carrinho antes de seguir as próximas etapas).
2.Obter um ID de produtos cadastrados com quantidade suficiente e colocar isso no body da requisição duas vezes.
3.Realizar requisição na rota /carrinhos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "produtos": [
    {
```

```
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
    },
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
    }
  ]
}
```

Padrão da resposta esperada:
400- Bad request

```
{
  "message": "Não é permitido possuir produto duplicado"
}
```

CTN19 - Cadastrar carrinho sem sucesso (Produto não cadastrado)

Descrição:
Não deve cadastrar um novo carrinho caso os produtos incluídos não estejam cadastrados.

Etapas do Teste:
1.Logar com sucesso em um usuário que não tenha um carrinho cadastrado(ou excluir carrinho antes de seguir as próximas etapas).
2.Obter um ID permitido mas não válido de produtos e colocar isso no body da requisição.
3.Realizar requisição na rota /carrinhos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:

```
{
  "produtos": [
    {
      "idProduto": "fdgssdgdgsgsd4",
      "quantidade": 1
    }
  ]
}
```

Padrão da resposta esperada:
400- Bad request

```
{
  "message": "Produto não encontrado"
}
```

CTN20 - Cadastrar carrinho sem sucesso (Produto sem quantidade o suficiente)

Descrição:
Não deve cadastrar um novo carrinho caso o produto cadastrado não tenha quantidade

suficiente em estoque conforme o pedido.

Etapas do Teste:
1.Logar com sucesso em um usuário que tenha um carrinho cadastrado(ou cadastrar um carrinho antes de seguir as próximas etapas).
2.Obter um ou mais IDs de produtos cadastrados e informar uma quantidade maior do que a em estoque, colocar isso no body da requisição.
3.Realizar requisição na rota /carrinhos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:
```
{
  "produtos": [
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 5000
    },
    {
      "idProduto": "YaeJ455lz3k6kSIzA",
      "quantidade": 15000
    }
  ]
}
```

Padrão da resposta esperada:
400- Bad request

```
{
  "message": "Produto não possui quantidade suficiente"
}
```

CTN21 - Cadastrar carrinho sem sucesso (Usuário com carrinho)

Descrição:
Não deve cadastrar um novo carrinho caso o usuário logado já possua um carrinho.

Etapas do Teste:
1.Logar com sucesso em um usuário que tenha um carrinho cadastrado(ou cadastrar um carrinho antes de seguir as próximas etapas).
2.Obter um ou mais IDs de produtos cadastrados com quantidade suficiente e colocar isso no body da requisição.
3.Realizar requisição na rota /carrinhos com o método POST.

Método de Autenticação:
Bearer Token.

Padrão do Body:
```
{
  "produtos": [
    {
      "idProduto": "BeeJh5lz3k6kSIzA",
      "quantidade": 1
```

```
    },
    {
      "idProduto": "YaeJ455lz3k6kSIzA",
      "quantidade": 3
    }
  ]
}
```


Padrão da resposta esperada:
400- Bad request


```
{
  "message": "Não é permitido ter mais de 1 carrinho"
}
```


Seção 03 - Fluxos de Teste

CFT01 - Fluxo de compra

Descrição:
Simular o fluxo de compra de um novo usuário.

Etapas do Fluxo:
1.Cadastrar.
2.Logar.
3.Olhar os produtos disponíveis.
4.Adicionar um dos produtos cadastrados ao carrinho.
5.Concluir a compra e excluir carrinho.

Status codes:
1. 201
2. 200
3. 200
4. 201
5. 200

CFT02 - Fluxo de usuário administrador

Descrição:
Simular as atividades que um novo usuário administrador pode realizar.

Etapas do Fluxo:
1.Cadastrar como administrador.
2.Logar.
3.Olhar os produtos cadastrados.
4.Cadastrar um novo produto.
5.Editar um produto cadastrado.

6.Excluir um produto cadastrado.
Status codes: 1. 201 2. 200 3. 200 4. 201 5. 200 6. 200

Link para a [suíte de testes na íntegra](#).

2.5 Candidatos para automação

Devido ao amplo tempo fornecido para o desenvolvimento deste projeto, todos os testes presentes na suíte serão automatizados e irão gerar um relatório, também automatizado.

3. Ferramentas

Ferramenta	Função
Google Docs	Documentação do projeto.
Google Slides	Apresentação em slides do projeto.
Postman	Realização de um mapeamento básico das rotas e para conferir defeitos manualmente.
XMind	Criação do Mapa Mental das rotas e seus respectivos status code.
VSCode(JS,Cypress & Mocha)	Desenvolvimento do código dos testes e report automatizado.
Git	Versionamento do código.
GitHub	Hospedagem de código e arquivos.

.