API TESTE DESAFIO NEXASS BACK-END

Este projeto tem como finaliade a criação de uma api JSON REST para gerenciamento de estoque de loja

Configuração/Ferramentas

- Rails 6.0.3
- Ruby 2.7.1
- Banco de dados postgres
- Docker and Postgres container
- Comando abaixo, configura uma imagem e cria um container com docker configurado:

```
sudo docker run -d
--name=postgres
-v /etc/localtime:/etc/localtime:ro
-e POSTGRES_USER=root
-e POSTGRES_PASSWORD=root
-v /storage/pgdata:/var/lib/postgresql/data
-p 5432:5432
--restart=always
postgres
```

- O projeto pode ser iniciado local com comando rails s.
- No database.yml do projeto configura host aponta para o docker por fim usuario root e senha root.
- Postman Canary utilizado para requisições REST.
- Caso não opite por esta configuração do banco de dados, basta altera o database.yml para forma que preferi.

Operações sobre produto

- Parametros enviados no formato json
- Definir o headers:
  - key: Content-Type
  - value: application/json

CADASTRA NOVO PRODUTO

- URL
  - POST http://localhost:3000/api/v1/products

```
{
  "product":
  {
    "name":"Suporte para 2 monitor MDBT",
    "price": 250.00
  }
}
```

- resposta

```
{
  "msg": "Produto registrado!",
  "status": "created"
}
```

ALTERA PRODUTO

- Parametro id=1
- URL
  - PUT http://localhost:3000/api/v1/products/:id
- exemplo http://localhost:3000/api/v1/products/1
- json enviado via body:

```
{
  "product":
  {
    "name":"Suporte para 2 monitor MDBT no maximo 37 polegadas",
    "price": 255.00
  }
}
```

- resposta

```
{
  "msg": "Produto atualizado!",
  "status": "created"
}
```

EXCLUIR UM PRODUTO

- parametro id=6
- URL
  - DELETE http://localhost:3000/api/v1/products/:id
- exemplo http://localhost:3000/api/v1/products/6
- retorno

```
{
  "msg": "Produto removido!"
}
```

- retorno em caso de erro

```
{
  "msg": "Produto não existe!"
}
```

PESQUISAR PRODUTO

- parametro id=5
- URL
  - GET http://localhost:3000/api/v1/products/:id
- exemplo http://localhost:3000/api/v1/products/6
- exemplo de retorno

```
{
  "id": 5,
  "name": "HyperX Alloy Core",
  "price": "400.0",
  "created_at": "2020-08-07T18:49:16.622Z",
  "updated_at": "2020-08-07T18:49:16.622Z"
}
```

- exemplo de erro

```
{
  "msg": "Produto não encontrado!"
}
```

OPERAÇÕES SOBRE LOJAS

- Parametros enviados no formato json

- Definir o headers:
  - key: Content-Type
  - value: application/json

CADASTRA LOJA

- Parametros enviados no formato json
- URL
  - POST http://localhost:3000/api/v1/stores

```
{
  "store":
  {
    "name":"Loja1",
    "street": "Algodoeiro",
    "neighborhood": "Eletronorte",
    "city":"Porto Velho",
    "cep": 76808518
  }
}
```

- resposta

```
{
  "msg": "Loja registrada!",
  "status": "created"
}
```

ALTERA UMA LOJA

- Parametro id=1
- URL
  PUT http://localhost:3000/api/v1/stores/:id
- exemplo http://localhost:3000/api/v1/stores/1
- json enviado via body:

```
{
  "store":
  {
    "name":"Loja1000",
    "street": "Algodoeiro teste",
    "neighborhood": "Eletronorte teste",
    "city":"Porto Velho teste",
    "cep": 76808518
  }
}
```

- resposta

```
{
  "msg": "Produto atualizado!",
  "status": "created"
}
```

EXCLUIR UMA LOJA

- parametro id=1
- URL
  DELETE http://localhost:3000/api/v1/stores/:id
- exemplo http://localhost:3000/api/v1/stores/1
- retorno

```
{
  "msg": "Loja removida!"
}
```

- retorno em caso de erro

```
{
  "msg": "Loja não existe!"
}
```

PESQUISAR LOJA

- parametro id=1
- URL
  GET http://localhost:3000/api/v1/stores/:id
- exemplo http://localhost:3000/api/v1/stores/1
- exemplo de retorno

```
{
  "id": 1,
  "name": "Loja1000",
  "street": "Algodoeiro teste",
  "neighborhood": "Eletronorte teste",
  "city": "Porto Velho teste",
  "cep": 76808518,
  "created_at": "2020-08-07T19:42:50.699Z",
  "updated_at": "2020-08-08T03:12:36.935Z"
}
```


- exemplo de erro


```
{
  "msg": "Loja não encontrada!"
}
```


OPERAÇÕES EM ESTOQUE

- Parametros enviados no formato json
- Definir o headers:
  - key: Content-Type
  - value: application/json

INSERINDO ITENS AO ESTOQUE

- requisitos via body:
  - id de um produto
  - id de uma loja
  - quantidade
- URL
  - POST http://localhost:3000/api/v1/stockitems
- exemplo json enviado


```
{
  "stockitem":
  {
    "product_id": 5,
    "store_id": 2,
    "quantities": 25
  }
}
```

- exemplo de sucesso

```
{
  "msg": "Item inserido ao estoque!",
  "status": "created"
}
```

- exemplo erros caso produto não existe ou loja

```
{
  "msg": "Product must exist and Store must exist"
}
```

ADICIONAR ITENS DE UM PRODUTO AO ESTOQUE

- requisitos via body:
  - quantidade
- requisitos via url
  - produto_id
- URL
  - PUT http://localhost:3000/api/v1/stockitems/add_quantities_of_product/:product_id
- exemplo de envio

```
{
  "stockitem":
  {
    "quantities": 1
  }
}
```

- exemplo de retorno apos atualizar o estoque
  - Lista de novas quantidades

```
[
    {
        "id": 1,
        "quantities": 42,
        "product_id": 5,
        "store_id": 2,
        "created_at": "2020-08-08T03:50:39.040Z",
        "updated_at": "2020-08-08T05:04:05.671Z"
    },
    {
        "id": 2,
        "quantities": 34,
        "product_id": 5,
        "store_id": 4,
        "created_at": "2020-08-08T04:24:20.252Z",
        "updated_at": "2020-08-08T05:04:05.678Z"
    }
]
```


- exemplo de erro se não tiver produto


```
{
  "msg": "Produto não existe em estoque"
}
```


REMOVE ITENS DE UM PRODUTO AO ESTOQUE

- requisitos body:
  - quantidade
- requisitos via url:
  - produto_id
- URL
  - PUT http://localhost:3000/api/v1/stockitems/lower/:product_id
  - exemplo de envio


```
{
  "stockitem":
  {
    "quantities": 1
  }
}
```


- exemplo de retorno apos atualizar o estoque
  - Lista as novas quantidades retiradas

```
[
    {
        "id": 1,
        "quantities": 42,
        "product_id": 5,
        "store_id": 2,
        "created_at": "2020-08-08T03:50:39.040Z",
        "updated_at": "2020-08-08T05:04:05.671Z"
    },
    {
        "id": 2,
        "quantities": 34,
        "product_id": 5,
        "store_id": 4,
        "created_at": "2020-08-08T04:24:20.252Z",
        "updated_at": "2020-08-08T05:04:05.678Z"
    }
]
```


- exemplo de erro se não tiver produto


```
{
  "msg": "Produto não existe em estoque"
}
```


- exemplo de erro se o produto tive a baixo do estoque
    - retorno com o estoque atualizado e com os que não foram atualizado


```
[
  {
    "id": 2,
    "quantities": 15,
    "product_id": 5,
    "store_id": 4,
    "created_at": "2020-08-08T04:24:20.252Z",
    "updated_at": "2020-08-08T06:06:44.033Z"
  },
  {
    "id": 1,
    "product_id": 5,
    "current_quantities": 0,
    "msg": "Estoque baixo!"
  }
]
```