



Projeto Shop Style

O Shop Style é uma loja física que vende roupas de todos os tipos e estilos. Os fundadores do Shop Style desejam agora abrir uma loja virtual e contrataram uma equipe para implementar. Os arquitetos já desenharam a solução e agora cabe a você implementar essa solução.

O projeto usará uma arquitetura de micro-serviços. Foi definindo a criação de cinco micro-serviços: customer, catalog, checkout, history e o bff-shop. Todos os micro-serviços devem ter testes unitários com cobertura de **pelo menos** 80%.

MS Customer

O MS customer tem a responsabilidade de armazenar e gerenciar os dados de usuário.

O MS customer possui os seguintes endpoints:

```
POST - /v1/login
POST - /v1/users
GET - /v1/users/:id
PUT - /v1/users/:id
```

Exemplo de um payload para cadastrar um usuário:

```
{
  "firstName": "Maria",
  "lastName": "Oliveira",
  "sex": "Feminino",
  "cpf": "000.000.000-00",
  "birthdate": "00/00/0000",
  "email": "maria@email.com",
  "password": "12345678",
  "active": true
}
```

Validações necessárias:


- Os campos firstName e lastName precisam ter no mínimo 3 caracteres.
- O campo sex só pode ter duas opções disponíveis Masculino e Feminino, caso contrário informar um erro ao usuário.
- O campo email precisa estar no formato de um email válido.
- O campo cpf precisa seguir o seguinte padrão (xxx-xxx-xxx-xx).
- O campo password precisa ter no mínimo 8 caracteres.
- O campo birthdate precisa ser salvo no formato brasileiro de datas (00/00/0000).

Observação:

- A senha deve estar encriptografada.
- Não deve permitir emails duplicados.
- O login será feito a partir do email e senha.
- Usar o PostgreSQL.

MS Catalog

O MS catalog é o responsável por armazenar os produtos e categorias que vão estar disponíveis na aplicação. Um produto tem uma ou mais variações, as variações têm os atributos cor, tamanho, preço e quantidade. O produto também está vinculado a uma ou mais categorias e uma categoria pode ter zero ou mais produtos. O MS catalog possui os seguintes endpoints:



```

POST - /v1/products
GET - /v1/products
GET - /v1/products/:id
PUT - /v1/products/:id
DELETE - /v1/products/:id

POST - /v1/variations
PUT - /v1/variations/:id
DELETE - /v1/variations/:id

POST - /v1/categories
GET - /v1/categories
GET - /v1/categories/:id/products
PUT - /v1/categories/:id
DELETE - /v1/categories/:id

```

Exemplo de um payload para cadastrar um produto:

```
{
  "name": "Camisa Oficial do Fluminense",
  "description": "A camisa pra você que é tricolor de coração",
  "active": true,
  "category_ids": [1,2,3]
}
```

Exemplo de um payload para cadastrar uma variação de um produto:

```
{
  "color": "tricolor",
  "size": "M",
  "price": 249.99,
  "quantity": 10,
  "product_id": 1
}
```

Exemplo de um payload para cadastrar uma categoria:

```
{
  "name": "Camisas de Futebol",
  "active": true
}
```

Validações necessárias:

- Os campos name, description, active e category_id são obrigatórios para salvar um produto.
- As categorias têm que estar ativa para um produto ser salvo.
- Todos os campos são necessários para cadastrar uma variação e categoria.

Observações:

- Usar o MongoDB.
- O Get de produtos deve retonar além das informações do produto, todas as suas variações.

MS Checkout

O MS Checkout é o responsável por armazenar as formas de pagamento, os descontos que podem ser dados e o ato de compra dos produtos. O MS checkout possui os seguintes endpoints:

```
POST - /v1/payments
GET - /v1/payments
GET - /v1/payments/:id
PUT - /v1/payments/:id
DELETE - /v1/payments/:id

POST - /v1/purchases
```

Exemplo de um payload para cadastrar um método de pagamento:

```
{
  "type": "credit card",
  "discount": 5,
  "status": true
}
```

Exemplo de um payload para efetuar uma compra:

```
{
  "user_id": 1,
  "payment_id": 1
  "cart": [
    {
      "variant_id": 1,
      "quantity": 1
    },
    {
      "variant_id": 2,
      "quantity": 5
    }
  ]
}
```

Validações necessárias:

- Todos os campos para salvar um pagamento são obrigatórios.

- Todos os campos para efetuar uma compra são obrigatórios.

Observações:

- O endpoint purchases precisa consultar se o usuário informado, o método de pagamento e o produto está ativo.
- Após efetuar a compra deve ser enviado uma mensagem para o ms history, com todas as informações de quantidade e total para ser salvo no history e também uma mensagem para o ms catalog diminuir o estoque do produto.
- As mensagens devem ser enviados usando o RabbitMQ.
- Usar o PostgreSQL.

MS History

O MS History tem a responsabilidade de armazenar todo o histórico de compras realizadas. O MS history possui o seguinte endpoint:

```
GET - /v1/historic/user/:idUser
```

O payload que esse endpoint deve retornar é o seguinte:

```
{
  "user": {
    "firstName": "Maria",
    "lastName": "Oliveira",
    "sex": "Feminino",
    "cpf": "000.000.000-00",
    "birthdate": "00/00/0000",
    "email": "maria@email.com"
  },
  "purchases": [
    {
      "paymentMethod": {
        "type": "credit card",
        "discount": 5,
        "status": true
      },
      "products": [
        {
          "name": "Camisa Oficial do Fluminense",
          "description": "A camisa pra você que é tricolor de coração",
          "color": "tricolor",
```

```

        "size": "M",
        "price": 249.99,
        "quantity": 1,
      },
      {
        "name": "Camisa Oficial do Fluminense",
        "description": "A camisa pra você que é tricolor de coração",
        "color": "tricolor",
        "size": "P",
        "price": 499.99,
        "quantity": 2,
      }
    ],
    "total": 749.99,
    "date": "00/00/0000"
  }
]
}

```

Observações:

- Usar o MongoDB.

MS BFF-Shop

Todos os micro-serviços serão de uso interno para os funcionários da empresa. Então precisa ser disponibilizado um ponto de entrada para que os clientes possam se comunicar com as funcionalidades. O MS bff-shop tem os seguintes endpoints:

```

POST - /v1/users
POST - /v1/login
GET - /v1/users/:id
PUT - /v1/users/:id
GET - /v1/products/:id
GET - /v1/categories/:id/products
GET - /v1/payments
POST - /v1/purchases
GET - /v1/historic/user/:idUser

```

Observação:

- Todos os endpoints precisam ser autenticados e autorizados via token JWT.