**DOCUMENTAÇÃO**

**FOUR STORE**

Autor:

Thales Borba

1. **INTRODUÇÃO**

Este documento especifica os requisitos do sistema a ser desenvolvido pelo estagiário Thales Borba, fornecendo aos usuários as informações necessárias para o entendimento do projeto e manuseio diário do sistema

1. **DESCRIÇÃO**

A API em questão trata-se de o desenvolvimento de um sistema de uma loja de roupas com controle e gerenciamento de estoque e processamento de vendas. Utilizando como tecnologia principal a linguagem JAVA.

1. **REPOSITORIO**

GitHub: <https://github.com/ThalesBorba/FourStore>

1. **FUNCIONALIDADES**
2. Modo de acesso

O sistema será acessado pela interface Swagger, tendo como endereço padrão localhost:8080 no navegador, após o programa ser inicializado. Os métodos receberão requisições de consulta, quando for inserida a respectiva id no path, e de cadastramento através de Json no corpo daquela. Para que os dados sejam salvos, é necessário possuir o banco MySql instalado com o banco ‘fourstore’.

2. Stock-Controller

Essa funcionalidade tem como objetivo criar, atualizar, pesquisar e apagar estoques de produto. Estoque e produto possuem uma relação um-para-um, portanto devem ser criados em única requisição. Não será possível criar mais de um estoque com um produto de mesma Sku. O produto jamais poderá ter um preço de venda que não seja pelo menos 25% maior que o preço de compra.

GET /stock/{sku}

Essa função receberá a Sku do produto no path, e receberá um Json contendo a id do estoque, a descrição e Sku do produto, e o quantidade atual desse no estoque.

PUT /stock/{sku}

Essa função receberá a Sku do produto a ser modificado no path e no corpo da requisição receberá novamente a Sku (que deve ser igual), a descrição, o preço de venda e compra e quantidade do produto. A regra dos 25% permanece, e quantidade deve ser um número positivo, que será **adicionado** ao já presente no estoque. Caso queira alterar somente a descrição ou preço, colocar a quantidade como 0.

DELETE /stock/{sku}

Essa função removerá o estoque cujo Sku for oferecido no path, contanto que este exista.

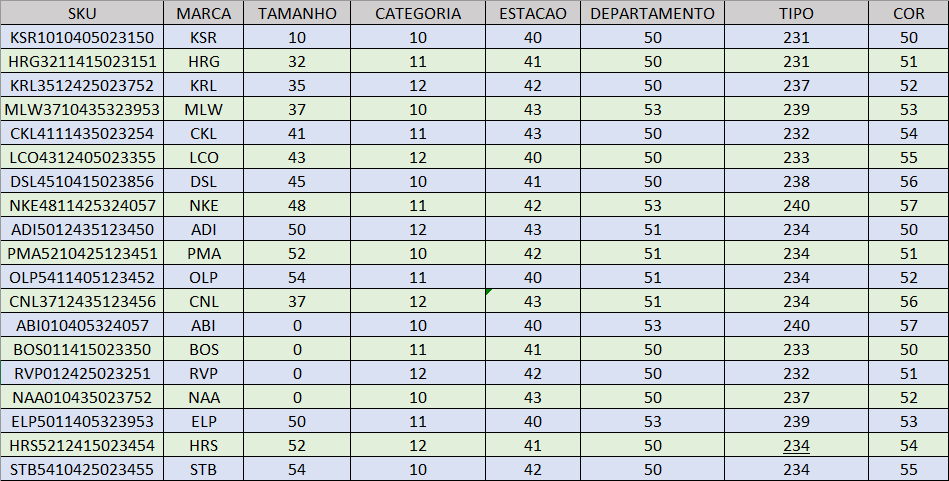
POST /stock/

Essa função irá criar um novo estoque, contanto que seja oferecida uma Sku condizente com o padrão apresentado na planilha “Sku - Examples.xlsx”, e que não tenha sido anteriormente cadastrada. Deverão ser oferecidas as informações iniciais em formato Json no corpo da requisição

GET /stock

Essa função irá retornar uma lista de todos os estoques já cadastrados em formato Json

Recorte de “Sku - Examples.xlsx”:



1. Client-Controller

Essa funcionalidade tem como objetivo criar, atualizar, pesquisar e apagar clientes no banco. Será feita uma verificação de cpf, que pode ter ou não pontos e traços, mas precisa ter 11 números. Também deverá ser escolhido uma forma de pagamento entre:

1 – Boleto à vista, 2 – Boleto parcelado, 3 – Cartão de crédito, 4 – Cartão de débito, 5 – Pix ou 6 – Cash

Os números de cartão de crédito, débito e Pix serão validados, e dependendo da forma de pagamento, será aplicado desconto no valor da venda.

GET /client/{cpf}

Essa função receberá um cpf no path, e receberá um Json contendo o cpf, o nome e as transações do cliente, caso esse já esteja cadastrado.

PUT /client/{cpf}

Essa função receberá a cpf do cliente a ser modificado no path e no corpo da requisição receberá novamente o cpf (que deve ser igual), o nome, o método de pagamento e o identificador de cartão/pix, caso esse seja o método escolhido. Caso sejam respeitadas as validações, o cliente será atualizado.

DELETE /client/{cpf}

Essa função removerá o cliente portador do cpf oferecido no path, contanto que este exista no banco de dados.

POST /stock/

Essa função irá cadastrar um novo cliente através de um Json que recebe o cpf, nome, método de pagamento e o identificador de cartão/pix , caso esse seja o método escolhido. Caso sejam respeitadas as validações, o cliente será criado.

GET /client

Essa função irá retornar uma lista de todos os clientes já cadastrados em formato Json

1. Transaction-Controller

GET /transaction

Essa função irá retornar uma lista de todas as transações registradas no banco em formato Json

POST /transaction

Essa função irá receber uma lista de Sku(s) e suas respectivas quantidades, e o cpf de um cliente (que deve ter sido previamente cadastrado) em formato Json. Serão verificados se o banco possui o cliente, se existe um estoque do produto com esta Sku, e se a quantidade deste é suficiente para realizar a transação. Caso a operação seja bem-sucedida, a quantidade vendida será subtraída do estoque, e será registrada a transação com a respectiva id, o nome e cpf do cliente que realizou a compra, e o lucro da venda de todos os produtos escolhidos, após aplicado o desconto que depende do método de pagamento que está cadastrado no perfil deste cliente.

GET /transaction/{id}

Essa função irá retornar a transação cujo id seja oferecido no path, com seus respectivos id e lucro, nome e cpf do cliente que a realizou.

Obs.: Transações representam registros que por questões de ética não devem ser alterados, portanto não terá métodos PUT ou DELETE

1. Product-Controller

GET /product

Essa função retornará uma lista com todos os produtos cadastrados no banco e seus respectivos Sku, descrição, preço de compra e preço de venda.

GET /product/{sku}

Essa função irá receber uma Sku no path, e retornará o respectivo produto com os detalhes contidos na sku, como: marca, tamanho, categoria, estação, departamento, tipo e cor, além das informações padrão.

Obs.: Todo produto deve estar atrelado a um estoque, e portanto, caso o usuário deseje realizar alterações, isso deve ser feito no endpoint de estoque.

1. **CONSIDERAÇÕES FINAIS**

Sistema FourStore versão 2.0 minimamente revisado, testado, excludente de erros e totalmente modularizado o que proporciona uma maior facilidade de manutenção e futuras atualizações.