Projeto Snack-bar

Autor: Enry Gonçalves
Github: github.com/enrylucas

# 1 Introdução

Este projeto foi desenvolvido para um processo seletivo de desenvolvedor, em que o candidato deveria cumprir com os entregáveis exigidos. O projeto foi desenvolvido na linguagem Java, utilizando a tecnologia Spring Boot para gerar o projeto base. Foi gerado uma aplicação Maven com as dependências Web, JPA e PostgreSQL, através do Spring Initializr (`https://start.spring.io/`).

# 2 Apresentação do problema

Somos uma startup do ramo de alimentos e precisamos de uma aplicação web para gerir nosso negócio. Nossa especialidade é a venda de lanches, de modo que alguns lanches são opções de cardápio e outros podem conter ingredientes personalizados.

A seguir, apresentamos a lista de ingredientes disponíveis:

Tabela 1: Tabela de ingredientes.

| INGREDIENTE | VALOR |
|---|---|
| Alface | R$ 0,40 |
| Bacon | R$ 2,00 |
| Hambúrguer de carne | R$ 3,00 |
| Ovo | R$ 0,80 |
| Queijo | R$ 1,50 |

Segue as opções de cardápio e seus respectivos ingredientes:

O valor de cada opção do cardápio é dado pela soma dos ingredientes que compõe o lanche. Além destas opções, o cliente pode personalizar seu lanche e escolher os ingredientes que desejar. Nesse caso, o preço do lanche também será calculado pela soma dos ingredientes.

Tabela 2: Tabela de lanches.

| LANCHE | INGREDIENTES |
|---|---|
| X-Bacon | Bacon, hambúrguer de carne e queijo |
| X-Burguer | Hambúrguer de carne e queijo |
| X-Egg | Ovo, hambúrguer de carne e queijo |
| X-Egg Bacon | Ovo, bacon, hambúrguer de carne e queijo |

Existe uma exceção à regra para o cálculo de preço, quando o lanche pertencer à uma promoção. A seguir, apresentamos a lista de promoções e suas respectivas regras de negócio:

- Light: Se o lanche tem alface e não tem bacon, ganha 10% de desconto.
- Muita carne: A cada 3 porções de carne o cliente só paga 2. Se o lanche tiver 6 porções, o cliente pagará 4. Assim por diante...
- Muito queijo: A cada 3 porções de queijo o cliente só paga 2. Se o lanche tiver 6 porções, o cliente pagará 4. Assim por diante...
- Inflação: Os valores dos ingredientes são alterados com frequência e não gastaríamos que isso influenciasse nos testes automatizados.

# 3 O projeto

Este projeto consiste em duas partes: o back-end, que consiste deste repositório; e o front-end, que está presente no repositório `https://github.com/enrylucas/snack-bar-front`. Aqui será discutido somente o que se refere ao back-end to projeto como um todo.

## 3.1 Estruturação

A estrutura dos códigos-fonte do projeto foi dividida em 4 subdiretórios:

- enry.snackbar;
- enry.snackbar.controller;

- enry.snackbar.model;
- enry.snackbar.repository.

Além desses principais diretórios, temos no diretório de teste o pacote enry.snackbar, que consiste da execução dos testes unitários. Por fim, temos os arquivos de configuração do projeto, sendo eles as dependências, o arquivo de configuração do banco de dados ( application.properties), o pom.xml e nbactions.xml. Na Figura 1 é ilustrado esse diretório, através da IDE NetBeans:

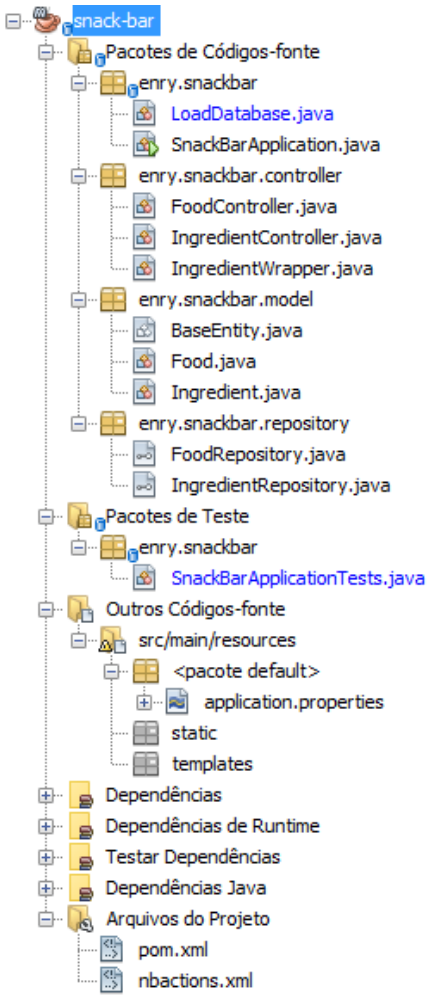


Figura 1: Estrutura do projeto.

Uma vez que o problema consiste essencialmente em calcular lanches pré-montados ou customizados, foi instanciado um banco de dados que armazena em uma tabela todos os ingredientes, onde a mesma possui uma relação muitos-para-muitos com uma outra tabela, a de lanches prontos.

Dessa forma, foi desenvolvido um modelo e um controlador para cada tabela, de forma que através do acesso de seus respectivos repositórios, fosse possível executar o CRUD de cada um deles. Este CRUD não é necessário para a realização da solução do problema. Sua implementação se justifica caso a empresa que utilize a aplicação pretenda escalar a quantidade de lanches(contando que seus ingredientes não sejam repetidos) e de ingredientes. Salvo a função UPDATE de ingredientes, esta funcionalidade é necessária para se comprovar que a regra de **negócio da inflação** (mudança de preço de ingrediente) não influenciará no resultado da aplicação.

Uma vez que o problema já nos diz o universo de ingredientes e lanches a tratar, foi definido também uma classe para pré-inserir estes dados no banco (LoadDatabase.java).

## 3.2 Funcionamento

O funcionamento da aplicação consiste essencialmente de chamadas HTTP, que são validadas e executadas de forma a acessar alguma informação no banco de dados. O acesso ao banco se dá através dos repositórios FoodRepository.java e IngredientRepository.java, que são classes filhas de JpaRepository. Estas classes que possibilitam o acesso ao banco de dados, e as mesmas tem seus métodos utilizados através de seus respectivos controladores (FoodController.java e IngredientController.java).

A aplicação em si consiste de, dado uma lista de ingredientes e suas respectivas quantidades, o usuário solicita ao back-end quanto será o custo do lanche, considerando todas as promoções. Para o caso de lanches pré-montados, a lista de ingredientes já é pré-definida, e por padrão, cada um terá uma quantidade unitária. Para o caso de lanches montados pelo cliente, a lista de ingredientes e suas respectivas quantidades será em função do input passado pelo front, sendo esse a interface que possibilitará o cliente a fazer seu pedido.

De fato, somente três de todas as rotas implementadas neste projeto são utilizadas pelo cliente, de forma indireta: getFoods(); getIngredients(); e calculatePrice(). Esta última, por sua vez, é a rota mais importante, visto que é a partir dela que temos acesso ao valor do lanche montado. As promoções **light**, **mais carne** e **mais queijo** se encontram implementadas em IngredientController.java e são chamadas

pela rota calculatePrice(). Aqui, vale ressaltar que **o desconto percentual do lanche light é calculado ao final**, de forma que ele recaia em cima dos outros descontos.

Uma observação importante é o fato de existir duas flags, isMeat e isCheese, nos ingredientes. Esta implementação se basea no fato de, caso haja mais ingredientes sendo incluidos e se deseje calcular as promoções em função de diferentes carnes e queijos, todos somados, estas flags possibilitarão um cálculo mais genérico da promoção. Vale ressaltar que, caso dois queijos ou duas carnes possuam preços diferentes, a primeira a ser validada terá seu preço como o desconto a ser feito.

# 4 Front-end

Este documento trata das informações pertinentes ao back-end da aplicação snack-bar. Para mais informações referentes ao front-end, favor consultar o doc do mesmo, em seu repositório.