**Dependências:**

Este projeto utiliza a ultima versão estável do spring-boot e gerenciador de dependências Dalston.SR1

**Build.plugins:**

**spring-boot-maven-plugin** – é utilizado para melhor gerencias as dependências do spring-boot

**maven-compiler-plugin** – é utilizado para melhor organizar as dependências para compilação e versão do Java

**maven-surefire-plugin** – é utilizado para executar todos os testes do junit durante o build

**docker-maven-plugin** – é utilizado para gerar o dockerfile e imagem da aplicação

**Outros frameworks**

**spring-boot-starter-web** – é utilizado para trabalhar com webapi, faço a remoção do tomcat e do logger pois irei configurá-lo mais a frente

**spring-boot-starter-undertow** – é utilizado para subir a aplicação em um servidor de aplicação embarcado undertow no lucar do tomcat que foi excluído

**spring-boot-starter-log4j** – é utilizado para configurar o log4j

**spring-boot-starter-test** – é utilizado para realização de testes unitários

**mockito-core** – é utilizado para trabalhar com mock de dados

**htmlunit** – é utilizado para realização de testes de integração

**commons-collections** – é um core para melhor trabalhar com collections

**commons-io** – é um core para trabalhar melhor com io

**commons-lang3** – é utilizado para melhor trabalhar com objetos

**gson** – é utilizado para trabalhar com serialização e deserialização de JSON

**guava** – e um core para trabalhar com objetos e outras funcionalidades

**Descrição**

Este projeto foi configurado para um ambiente de integração contínua, todo o fluxo configurado em seu pom funciona automaticamente da seguinte maneira:

* Build do projeto
* Testes Unitários
* Teste de integração
* Geração do dockerfile
* Build da imagem

O projeto foi configurado em um projeto principal que contém os módulos de front-end e back-end

As variáveis de configuração do front-end ficam no arquivo home-app.js (src/main/resources/static/javascript/app/home-app.js) na variável host\_app fica o apontamento do host para o back-end, outra configuração importante é dentro do .pom existem duas propriedades docker.image.name (que corresponde ao nome da imagem que será gerada) e docker.host (que corresponde ao ip do servidor com o Docker). Outra configuração importante é no teste de integração IntegrationTest (br.com.jonathan.test.junit) apenas a variável host\_test\_app\_back que aponta para o back-end da aplicação

As variáveis de configuração do back-end ficam apenas no .pom, docker.image.name (que corresponde ao nome da imagem que será gerada) e docker.host (que corresponde ao ip do servidor com o Docker)

**Build**

Para iniciar a aplicação basta seguir os itens abaixo:

1. Configurar as variáveis docker.image.name, docker.host dos poms do back-end e front-end
2. Executar o maven no back-end (mvn clean install -e)
3. Iniciar a apliação back-end no Docker (docker run -d --net=host snackshopbackend)
4. Apontar as variáveis de home\_app.js (host\_app) e IntegrationTest (host\_test\_app\_back) para o back-end inicializado no Docker
5. Executar o maven no front-end (mvn clean install -e)
6. Iniciar a aplicação front-end no Docker (docker run -d --net=host snackshopfrontend)

A aplicação web estará disponível na porta 8080 (front-end) e 8888 (back-end), a alteração das portas podem ser alteradas no application.yml

Para alterar o fluxo do build com o Docker basta comentar o docker-maven-plugin, após isto executar o comando do maven (mvn) copiar o dockerfile disponível na raiz do módulo e fazer o build da imagem (docker build -t snackshopbackend . ou docker build -t snackshopfrontend .)