МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3  
по дисциплине: Технологии Web-программирования

Выполнил: ст. гр. ПВ-41

Мишина Ю.А.

Проверил: Картамышев С.В.

Белгород 2020г.

**Цель**: познакомиться с основами backend разработки web-приложений. Научится писать программы на языке Java. Познакомиться с основами работы docker. Познакомиться с фреймворком Spring Boot и научиться разворачивать проект, производить его настройку. Научится работать с API в приложении Postman.

**Задание к лабораторной работе:**

1. Развернуть базовое приложение Spring Boot App Basic.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

**Выполнение**

В данной лабораторной работе мы будем реализовывать простой Rest сервис, с двумя endpoints.

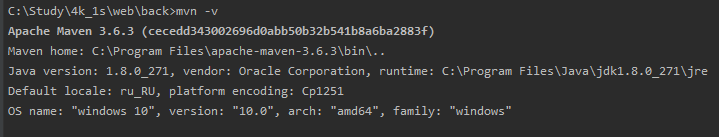
Язык программирования: Java 8

Фреймворк: Spring, Spring Boot

Система сборки: Maven

Контейниризатор: Docker

После установки java8 и maven. Проверим работу maven, узнав его версию.



DockerFile – файл, в котом будет находится конфигурация нашего Docker-образа

FROM openjdk:8  
ADD target/back\_lab3\_web-1.0-SNAPSHOT.jar back\_lab3\_web-1.0-SNAPSHOT.jar  
EXPOSE 8080  
ENTRYPOINT ["java", "-jar", "back\_lab3\_web-1.0-SNAPSHOT.jar"]

pom.xml – файл в котором будут находится все необходимые нам зависимости.

Нам понадобиться библиотека для реализации контролеров, поднятия tomcat.

<plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
</plugin>

**Создаём точку запуска нашего приложения**

package com;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class App {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(App.class, args);  
 }  
  
}

@SpringBootApplication – аннотация фреймворка, которая делает очень кучу всего. Но нам важно, что она возьмёт свою базовую конфигурацию и настроит зависимости, заинжектид, запустит tomcat на 8080 порту.

**Создаём контроллеры**

package com.web;  
  
import com.model.News;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.annotation.PostConstruct;  
import java.util.ArrayList;  
import java.util.List;  
  
@RestController  
@RequestMapping("/api")  
public class Controller {  
  
 private List<News> inf = new ArrayList<>();  
  
 @PostConstruct  
 private void init() {  
 for (int i = 0; i < 5; i++) {  
 inf.add(News.*builder*()  
 .title("My title" + i)  
 .description("My description" + i)  
 .build());  
 }  
 }  
  
 @GetMapping("/news")  
 public List<News> getLists() {  
 return inf;  
 }  
  
}

@RestController – аннотация, которая говорит, что этот класс является компонентом и так же является REST контроллером

@RequestMapping("/test") – url этого контроллера начнётся с /test

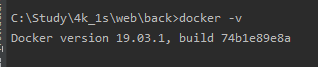
Над методами находится аннотация @GetMapping("/lists") – которая говорит, какой будет тип запроса (GET, POST, DEL и.т.д.) и так же часть его url.

В нашем случае методы будут отдавать json в виде строки.

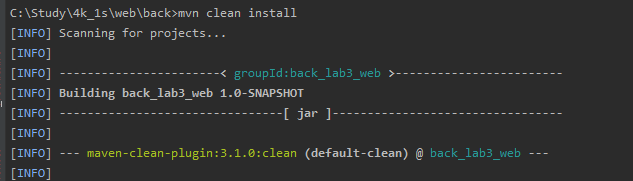
**Добавление приложения в Docker**

Docker был установлен на ОС Windows10.

Проверим работу докер командой docker -v

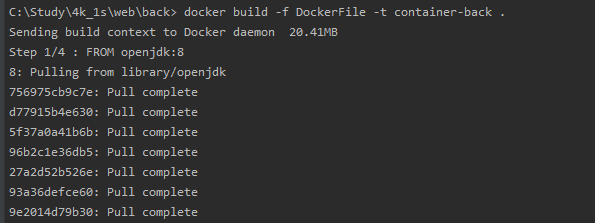


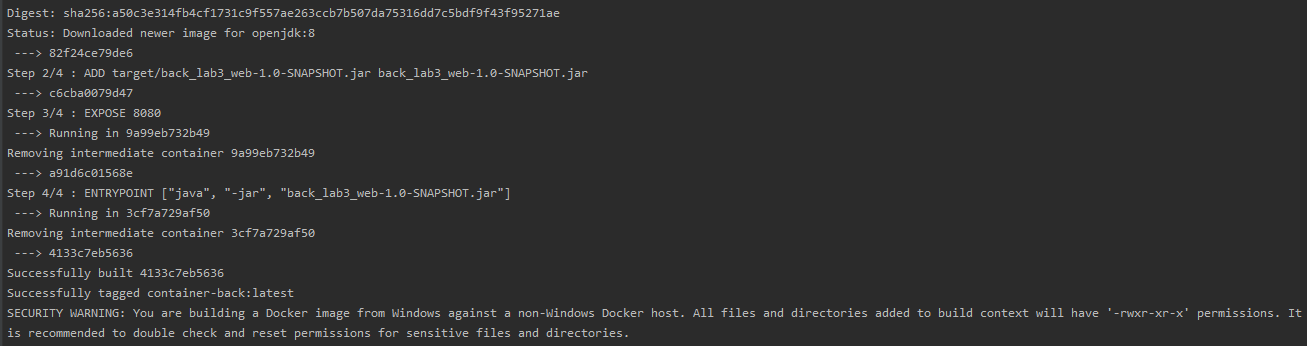
Компилируем приложение и создаём из него jar



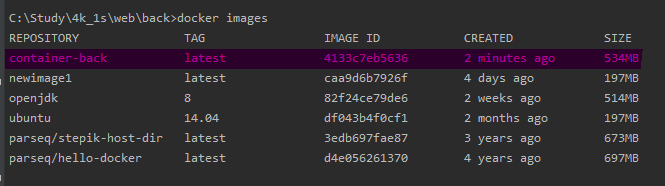
…

Сборка образа с именем container-back





После загрузки всего необходимого докер образа, он должен появиться в списке образов. Проверим это командой docker images



Для запуска образа выполнить кодманду

docker run -p 8080:8080 container-back

