Разработка информационной системы онлайн выбора тем по Выпускной Квалификационной Работе.

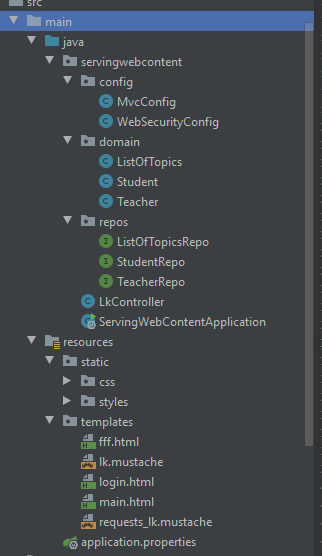
В программе должны быть реализованы такие функции, как: авторизация, просмотр тем и отправление заявки преподавателю (со стороны студента), просмотр тем и отправление решения о согласии (со стороны преподавателя), отслеживание статуса.

Код программы выполнен на языке Java JDK 1.8 с использованием системы контроля версии Maven, Spring-фреймворка – подспорья для запуска (Spring Boot) и работы с базой данных (Crud Repository, Hibernate), в качестве базы данных использовался MySQL. HTML – диалекты mustache, thymeleaf (для доступа к базе данных и полям классов, описывающих сущности), а также бесплатный UI Kit.

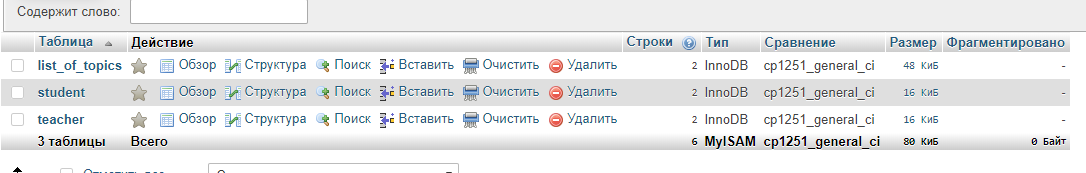
Документ разделен на три части – Детальный обзор, Запуск, Интерфейс

* Детальный обзор

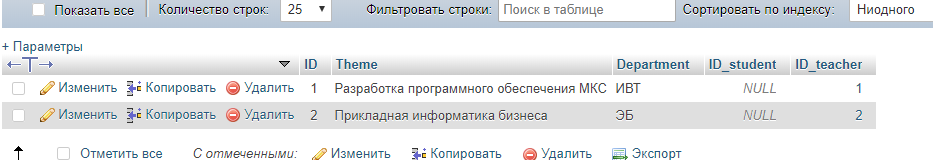
1. Структура программы



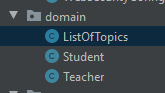
1. База данных



Преподаватели, студенты и список тем – в них записаны id студента и преподавателя, а также название темы. Поля id могут быть null.



Для описания сущностей были созданы классы: Student, Teacher, ListOfTopics



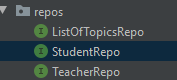
Далее – необходимо создать поля, которые будут ассоциироваться с конкретными атрибутами таблиц.

На примере класса Student



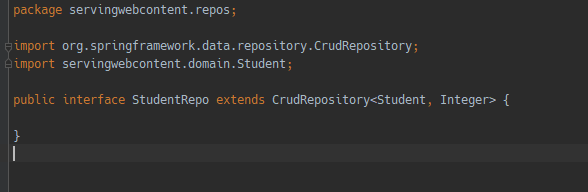
Здесь – указывается название таблицы, названия столбцов. В поле с Id используется стратегия автоматического определения – автоинкремент происходит силами MySQL. Также Getter и Setter для чтения и изменения данных.

Далее необходимо создать репозитории под каждый из классов, описывающих сущности:



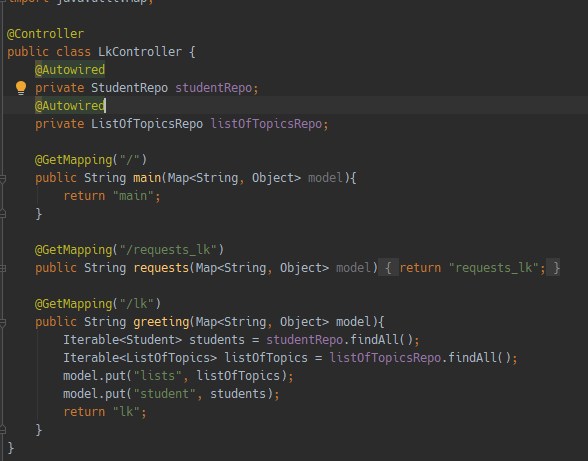
Это нужно для того, чтобы не приходилось писать тривиальные запросы SQL вручную, а брать сразу готовые шаблоны из Crud Repository

Репозиторий представляет собой интерфейс, который наследует Spring интерфейс CrudRepository. В качестве параметров указывается класс, описывающий таблицу и тип первичного ключа.

Пример репозитория для студента:  


Тем не менее, в процессе дальнейшей разработки так или иначе придется дописывать собственные запросы.

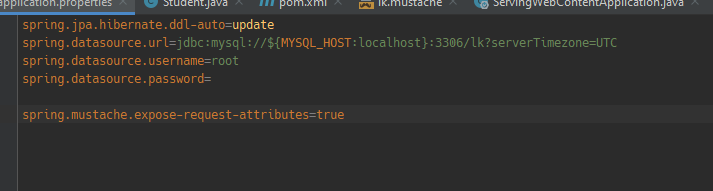
Затем необходимо создать контроллер – это класс, с помощью которого будут осуществляться GET и POST запросы.



Тип запроса определяется аннотацией. В скобках и после return – указывается название веб-странички. В методе greeting мы получаем информацию из базы данных. Заносим в Itetable<Student> используя команду репозитория findAll();

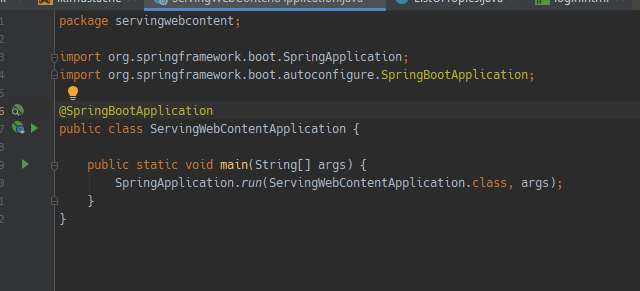
После чего, делаем put в мапу – заносим данные, которые потом считаем в web.

Для того, чтобы подключиться к базе данных, в файл application.properties были внесены следующие данные:



Во второй строчке указывается название базы данных. В данном случае “lk”. 

Затем указываются логин и пароль для доступа.

Основной класс для запуска приложения - ServingWebContentApplication

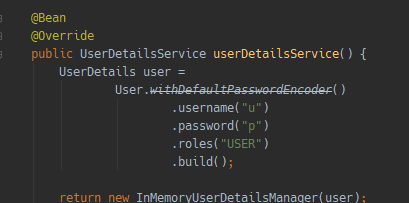
Далее – классы, использующиеся для создания авторизации.



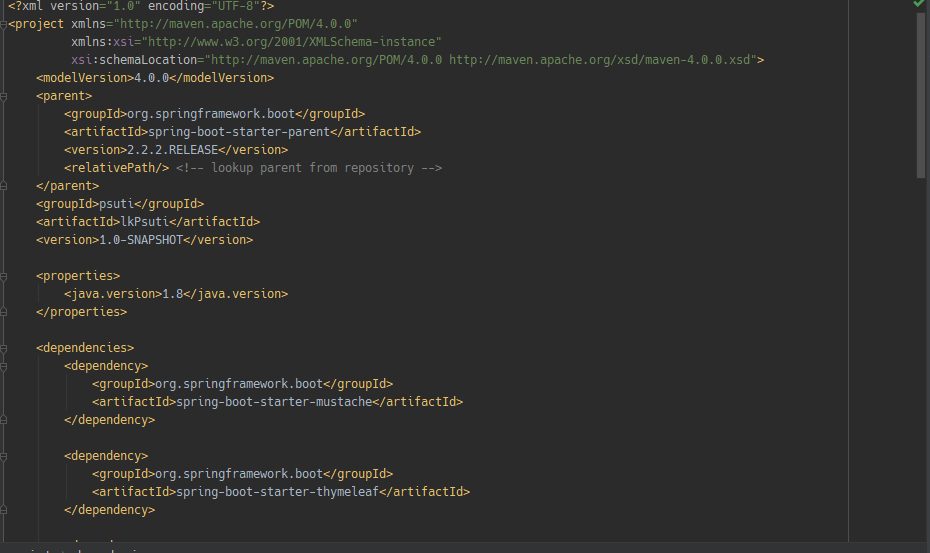
Этот класс (MvcConfig) предназначен для задания отображения контроллера – мы объявляем, что приложение будет иметь систему авторизации.

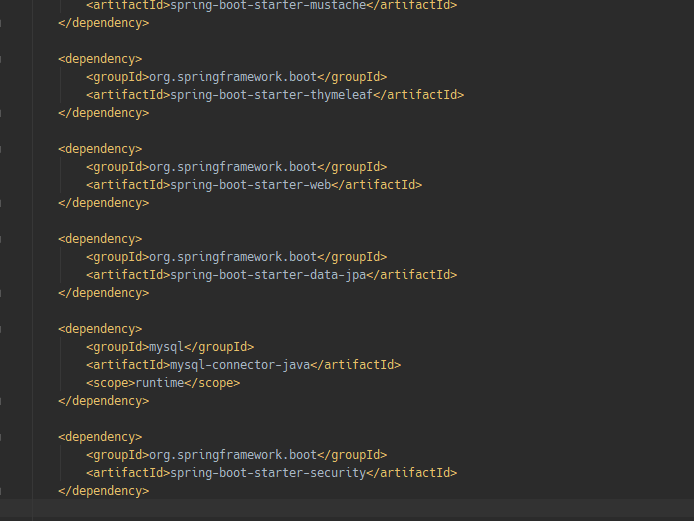
Затем, в классе WebSecurityConfig мы указываем всю необходимую конфигурацию, а именно: права доступа к страницам, файлам, ссылка на страницу входа, ссылка на страницу при успешной авторизации.

Для отладки программы использовался метод userDetailsService(); в котором при каждом запуске создавался один и тот же пользователь с одним и тем же логин\паролем.



Также, работая в системе контроля версий Maven, необходимо было задать соответствующие зависимости в файле Pom.xml

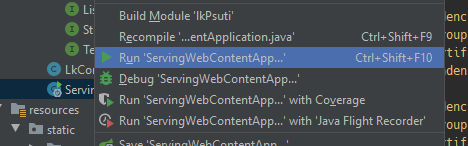




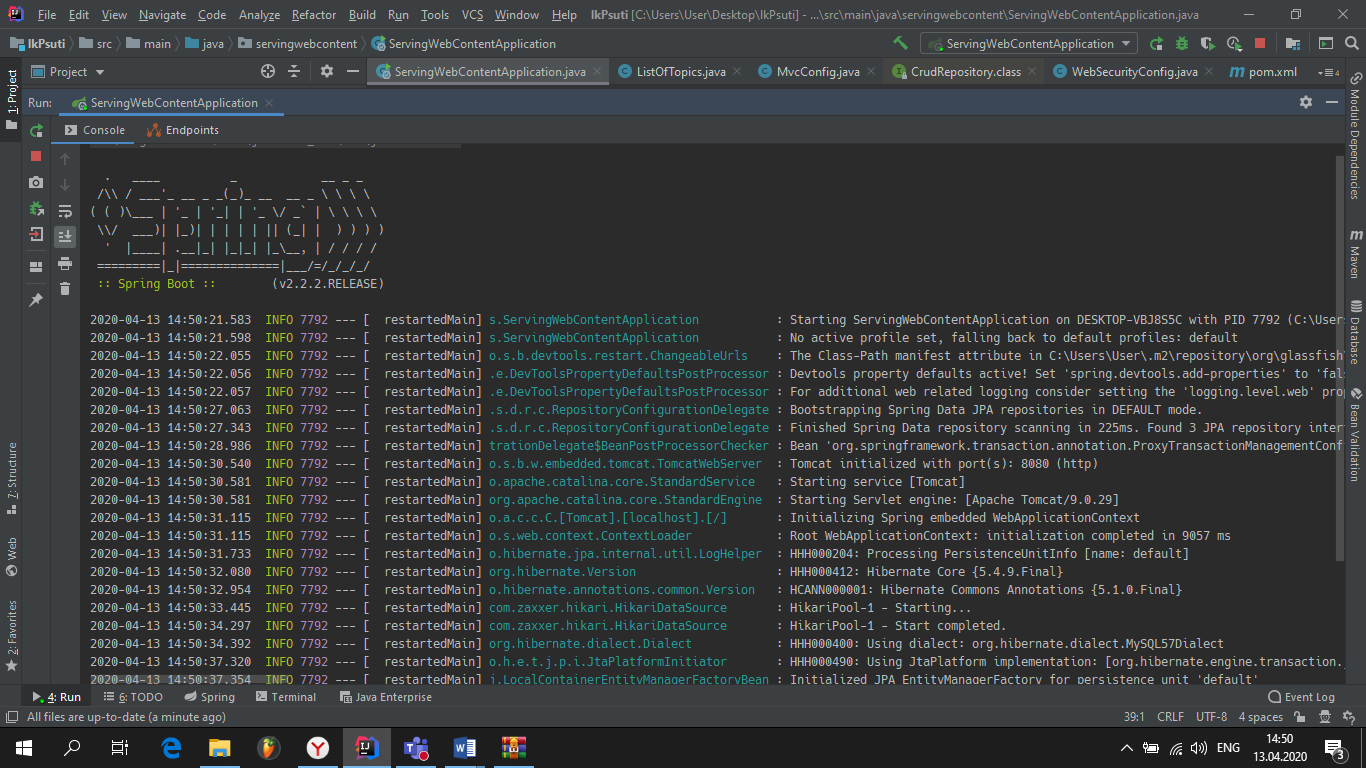
Они необходимы для работы с бд, диалектами html и т.д.

* Запуск программы

Для запуска программы необходимо нажать правой кнопкой мыши по классу ServingWebContentApp и нажать Run



После этого мы увидим, что пошла загрузка



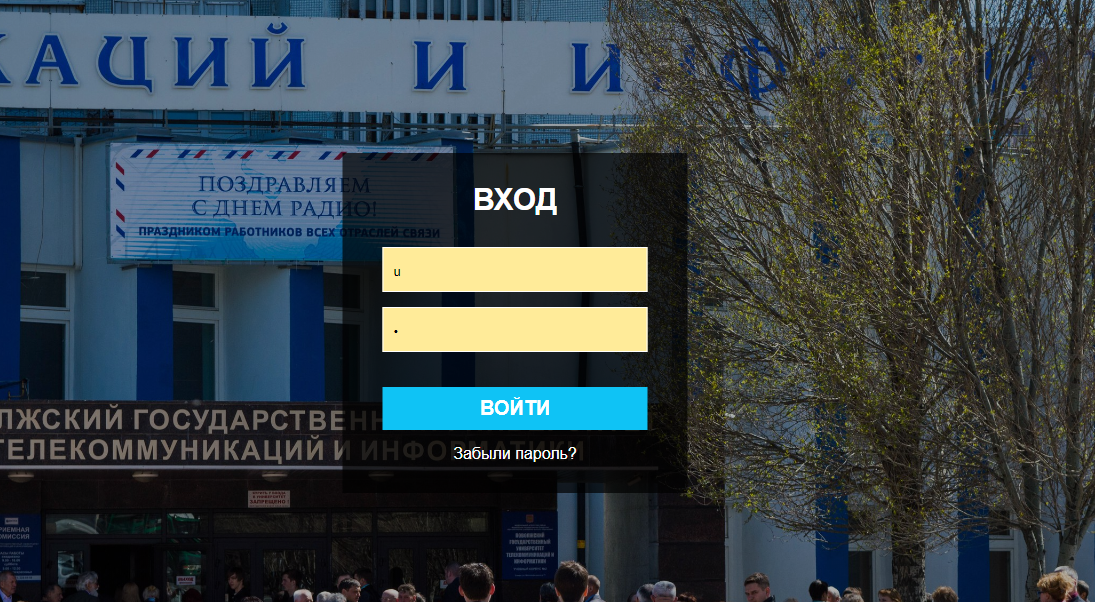
После появления строчки об успешном запуске, можно переходить к веб



Для этого нужно открыть браузер и в адресной строке вписать localhost:8080/login

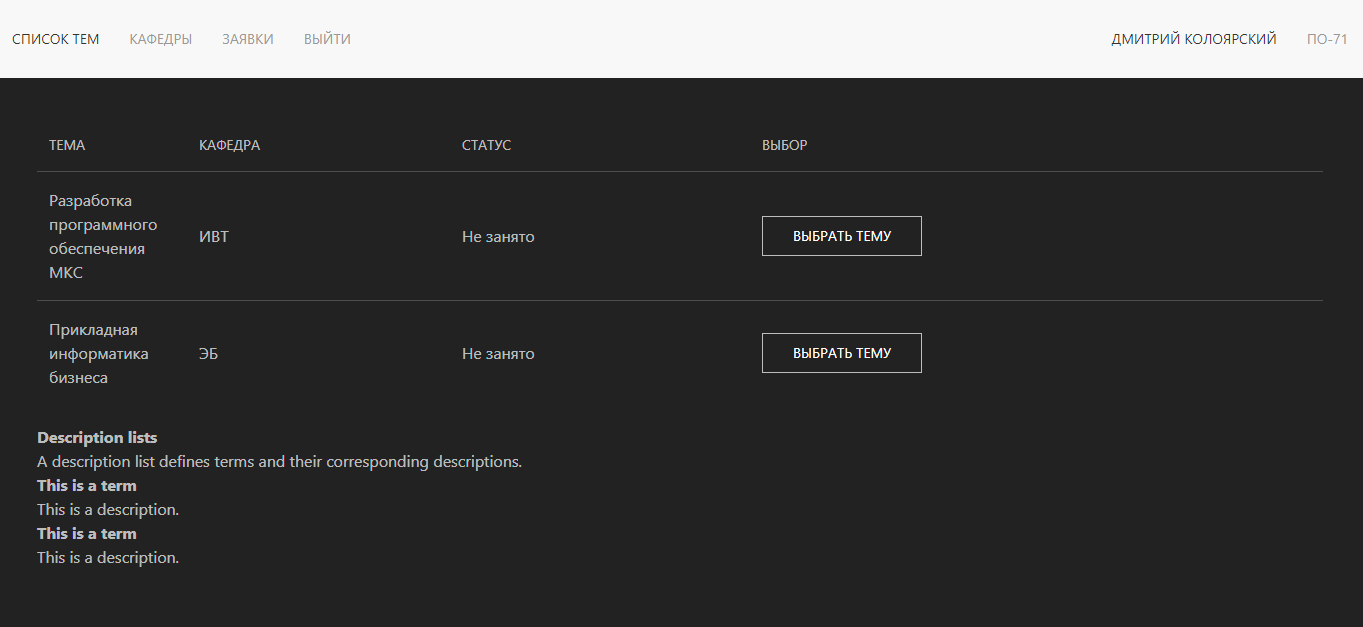
* Интерфейс

Интерфейс авторизации выглядит следующим образом:



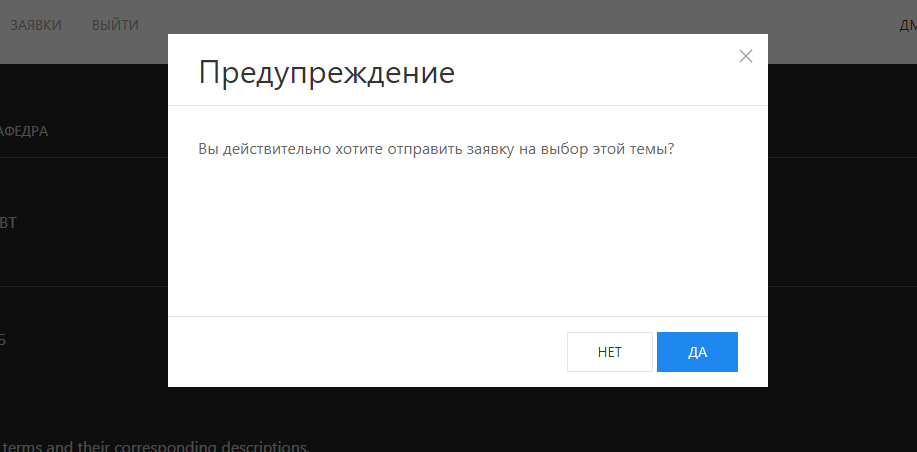
Если неправильно ввести логин и пароль, а также попытаться по прямой ссылке перейти на страницу личного кабинета – то ничего не выйдет.

После авторизации видим следующее:

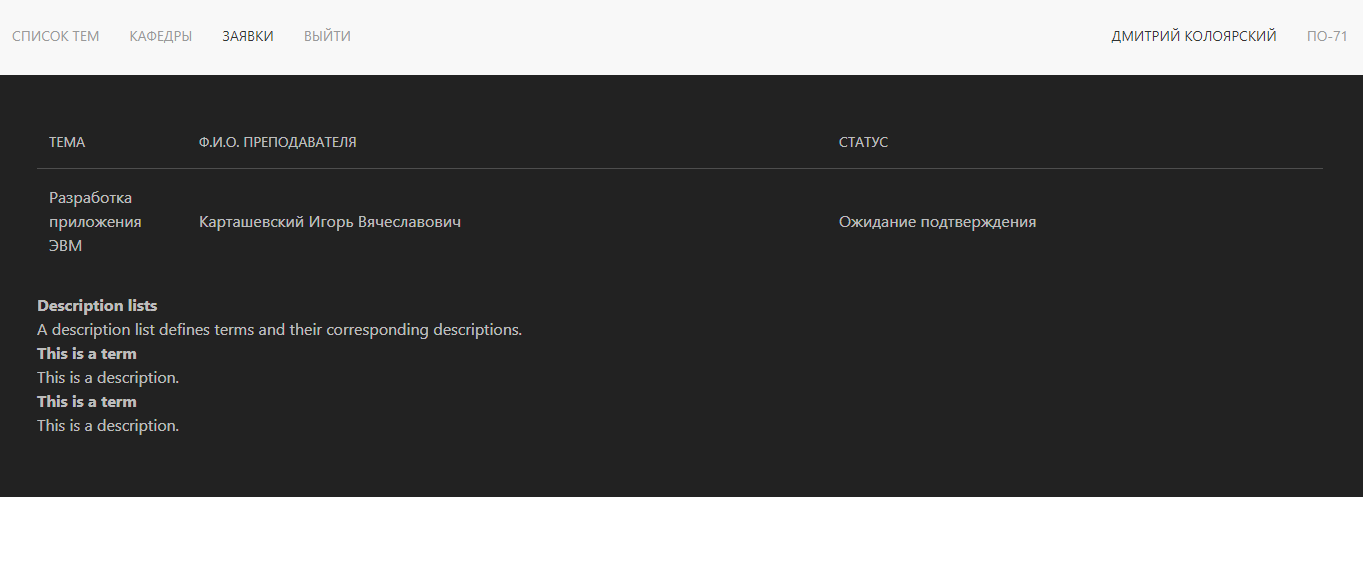


Сверху навбар с меню, справа сверху в углу – имя и фамилия того, кто зашел в учетную запись.

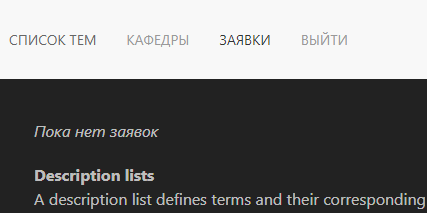
При нажатии на кнопку «Выбрать тему» выскакивает подтверждение выбора



Текущие заявки можно будет отслеживать в пункте «Заявки»

Если имеются, то можно наблюдать табличку:  


Если нет, то будет такой текст:



После завершения работы – нужно нажать кнопку «Выйти», после чего нас вновь вернет на страницу авторизации.

Исходный код:

**Класс MvcConfig.java**

package servingwebcontent.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class MvcConfig implements WebMvcConfigurer {  
  
 public void addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController("/login").setViewName("login.html");  
 }  
  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("/static/\*\*").addResourceLocations("/static/");  
 }  
}

**Класс WebSecurityConfig.java**

package servingwebcontent.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
  
@Configuration  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers(  
 "/",  
 "/js/\*\*",  
 "/css/\*\*",  
 "/img/\*\*",  
 "/styles/\*\*",  
 "/webjars/\*\*").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .defaultSuccessUrl("/lk", true)  
 .permitAll()  
 .and()  
 .logout()  
 .logoutUrl("/logout")  
 .permitAll();  
 }  
  
 @Bean  
 @Override  
 public UserDetailsService userDetailsService() {  
 UserDetails user =  
 User.*withDefaultPasswordEncoder*()  
 .username("u")  
 .password("p")  
 .roles("USER")  
 .build();  
  
 return new InMemoryUserDetailsManager(user);  
 }  
}

**Класс ListOfTopics.java**

package servingwebcontent.domain;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "list\_of\_topics")  
public class ListOfTopics {  
  
 @Id  
 @Column(name = "ID")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id\_list;  
  
 @Column(name = "Theme")  
 private String theme;  
 @Column(name = "Department")  
 private String department;  
 @Column(name = "ID\_student")  
 private Integer id\_student;  
 @Column(name = "ID\_teacher")  
 private Integer id\_teacher;  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 public Integer getId\_list() {  
 return id\_list;  
 }  
  
 public void setId\_list(Integer id\_list) {  
 this.id\_list = id\_list;  
 }  
  
 public String getTheme() {  
 return theme;  
 }  
  
 public void setTheme(String theme) {  
 this.theme = theme;  
 }  
  
 public Integer getId\_student() {  
 return id\_student;  
 }  
  
 public void setId\_student(Integer id\_student) {  
 this.id\_student = id\_student;  
 }  
  
 public Integer getId\_teacher() {  
 return id\_teacher;  
 }  
  
 public void setId\_teacher(Integer id\_teacher) {  
 this.id\_teacher = id\_teacher;  
 }  
}

**Класс Teacher.java**

package servingwebcontent.domain;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "teacher")  
public class Teacher {  
 @Id  
 @Column(name = "ID\_teacher")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id\_teacher;  
  
 @Column(name = "Name")  
 private String name;  
 @Column(name = "Surname")  
 private String surname;  
 @Column(name = "Lastname")  
 private String lastname;  
 @Column(name = "Department")  
 private String department;  
  
 public Integer getId\_teacher() {  
 return id\_teacher;  
 }  
  
 public void setId\_teacher(Integer id\_teacher) {  
 this.id\_teacher = id\_teacher;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 }  
  
 public String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
}

**Класс Student.java**

package servingwebcontent.domain;  
  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "student")  
public class Student {  
  
 @Id  
 @Column(name = "ID\_student")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id\_student;  
  
 @Column(name = "Name")  
 private String name;  
 @Column(name = "Surname")  
 private String surname;  
 @Column(name = "Lastname")  
 private String lastname;  
  
 public Integer getId\_student() {  
 return id\_student;  
 }  
  
 public void setId\_student(Integer id\_student) {  
 this.id\_student = id\_student;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getLastname() {  
 return lastname;  
 }  
  
 public void setLastname(String lastname) {  
 this.lastname = lastname;  
 } **}**

**Класс ListOfTopicsRepo**

package servingwebcontent.repos;  
  
import org.springframework.data.repository.CrudRepository;  
import servingwebcontent.domain.ListOfTopics;  
  
public interface ListOfTopicsRepo extends CrudRepository<ListOfTopics, Integer> {  
  
}

**Класс TeacherRepo**

package servingwebcontent.repos;  
  
import org.springframework.data.repository.CrudRepository;  
import servingwebcontent.domain.Teacher;  
  
public interface TeacherRepo extends CrudRepository<Teacher, Integer> {  
  
}

**Класс StudentRepo**

package servingwebcontent.repos;  
  
import org.springframework.data.repository.CrudRepository;  
import servingwebcontent.domain.Student;  
  
public interface StudentRepo extends CrudRepository<Student, Integer> {  
  
}

**Класс LkController**

package servingwebcontent;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import servingwebcontent.domain.ListOfTopics;  
import servingwebcontent.domain.Student;  
import servingwebcontent.repos.ListOfTopicsRepo;  
import servingwebcontent.repos.StudentRepo;  
  
import java.util.Map;  
  
@Controller  
public class LkController {  
 @Autowired  
 private StudentRepo studentRepo;  
 @Autowired  
 private ListOfTopicsRepo listOfTopicsRepo;  
  
 @GetMapping("/")  
 public String main(Map<String, Object> model){  
 return "main";  
 }  
  
 @GetMapping("/requests\_lk")  
 public String requests(Map<String, Object> model){  
 return "requests\_lk";  
 }  
  
 @GetMapping("/lk")  
 public String greeting(Map<String, Object> model){  
 Iterable<Student> students = studentRepo.findAll();  
 Iterable<ListOfTopics> listOfTopics = listOfTopicsRepo.findAll();  
 model.put("lists", listOfTopics);  
 model.put("student", students);  
 return "lk";  
 }  
}

**Класс ServingWebContentApplication**

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