**Министр науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 4

Выполнил студент группы № M32071

Реброва Татьяна Ивановна

Проверил:

Бутенко Олег Романович

Санкт-Петербург  
2022

KotikiLabApplication.java  
  
package ru.itmo.kotikilab;  
  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import ru.itmo.kotikilab.entities.\*;  
import ru.itmo.kotikilab.services.MainService;  
  
import java.time.LocalDate;  
  
@SpringBootApplication  
public class KotikiLabApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(KotikiLabApplication.class, args);  
 }  
  
 @Bean  
 CommandLineRunner run(MainService mainService) {  
 return args -> {  
 Owner owner = new Owner("Masha", LocalDate.of(2002, 03, 29), "username", "1234", "user");  
 mainService.saveOwner(owner);  
 Kotik kotik = new Kotik("Rubi", LocalDate.of(2022, 03, 29), "No name", Color.GREY, owner);  
 kotik.setOwnerId(owner);  
 mainService.saveKotik(kotik);  
 //mainService.addOwnerToKotik(owner.getId(), kotik.getId());  
  
 Owner owner2 = new Owner("Tanya", LocalDate.of(2002, 05, 29), "username2", "1234", "admin");  
 mainService.saveOwner(owner2);  
 Kotik kotik2 = new Kotik("Vudi", LocalDate.of(2022, 03, 29), "No name", Color.WHITE, owner);  
 kotik2.setOwnerId(owner2);  
 mainService.saveKotik(kotik2);  
 //mainService.addOwnerToKotik(owner2.getId(), kotik2.getId());  
  
 /\*Friend friend = mainService.addFriends(kotik.getId(), kotik2.getId());  
 mainService.saveFriend(friend);\*/  
 };  
 }  
}  
  
  
  
MainController.java  
  
package ru.itmo.kotikilab.controller;  
  
import lombok.RequiredArgsConstructor;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.core.context.SecurityContext;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.userdetails.UserDetails;  
import ru.itmo.kotikilab.entities.Friend;  
import ru.itmo.kotikilab.entities.Kotik;  
import ru.itmo.kotikilab.entities.Owner;  
import ru.itmo.kotikilab.services.MainService;  
import ru.itmo.kotikilab.wrapper.FriendWrap;  
import ru.itmo.kotikilab.wrapper.KotikWrap;  
import ru.itmo.kotikilab.wrapper.OwnerWrap;  
  
import java.net.URI;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
//@RequestMapping("/api")  
@RequiredArgsConstructor  
public class MainController {  
 private final MainService kotikiService;  
  
 @GetMapping("/")  
 public String hello() {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 Owner owner = kotikiService.findOwnerByUsername(auth.getName());  
 return "Hello " + owner.getName();  
 }  
  
 @GetMapping("/error")  
 public String error() {  
 return "Error";  
 }  
  
 @GetMapping("/user")  
 public String user() {  
 return "User";  
 }  
  
 @GetMapping("/admin")  
 public String admin() {  
 return "Admin";  
 }  
  
 @GetMapping("/admin/owners")  
 public ResponseEntity<List<OwnerWrap>>getOwners(){  
 return new ResponseEntity<>(  
 kotikiService.findAllOwners().stream().  
 map((owner -> owner.getOwnerWrap())).  
 collect(Collectors.toList()), HttpStatus.ACCEPTED);  
 };  
  
 @PostMapping("/admin/owner/createowner")  
 public ResponseEntity<OwnerWrap>createOwner(@RequestParam String name, @RequestParam String date, @RequestParam String username, @RequestParam String password, @RequestParam String role) {  
 kotikiService.createOwner(name, date, username, password, role);  
 return ResponseEntity.ok().build();  
 }  
  
 @PostMapping("/admin/owner/save")  
 public ResponseEntity<OwnerWrap>saveOwner(@RequestBody Owner owner){  
 URI uri = URI.create(ServletUriComponentsBuilder.fromCurrentContextPath().path("/api/owner/save").toUriString());  
 OwnerWrap ownerWrap = owner.getOwnerWrap();  
 return ResponseEntity.created(uri).body(ownerWrap);  
 }  
  
 @GetMapping("/admin/kotiks")  
 public ResponseEntity<List<KotikWrap>>getKotiks(){  
 return new ResponseEntity<>(  
 kotikiService.findAllKotiks().stream().  
 map((kotik -> kotik.getKotikWrap())).  
 collect(Collectors.toList()), HttpStatus.ACCEPTED);  
 };  
  
 @PostMapping("/admin/kotik/createkotik")  
 public ResponseEntity<KotikWrap>createKotik(@RequestParam String name, @RequestParam String date, @RequestParam String breed, @RequestParam int color, @RequestParam int ownerId) throws Exception {  
 kotikiService.createKotik(name, date, breed, color, ownerId);  
 return ResponseEntity.ok().build();  
 }  
  
 @PostMapping("/admin/kotik/save")  
 public ResponseEntity<Kotik>saveKotik(@RequestBody Kotik kotik){  
 URI uri = URI.create(ServletUriComponentsBuilder.fromCurrentContextPath().path("/api/kotik/save").toUriString());  
 return ResponseEntity.created(uri).body(kotikiService.saveKotik(kotik));  
 }  
  
 @GetMapping("/admin/friends")  
 public ResponseEntity<List<FriendWrap>>getFriends(){  
 return new ResponseEntity<>(  
 kotikiService.findAllFriends().stream().  
 map((friend -> friend.getFriendWrap())).  
 collect(Collectors.toList()), HttpStatus.ACCEPTED);  
 };  
  
 @PostMapping("/admin/friend/save")  
 public ResponseEntity<Friend>saveFriend(@RequestBody Friend friend){  
 URI uri = URI.create(ServletUriComponentsBuilder.fromCurrentContextPath().path("/api/friend/save").toUriString());  
 return ResponseEntity.created(uri).body(kotikiService.saveFriend(friend));  
 }  
  
 @PostMapping("/admin/friend/addfriend")  
 public ResponseEntity<FriendWrap>addFriend(@RequestParam int idKotik, @RequestParam int idFriend) {  
 kotikiService.addFriends(idKotik, idFriend);  
 return ResponseEntity.ok().build();  
 }  
  
 @GetMapping("/user/kotiks")  
 public ResponseEntity<List<KotikWrap>>getUserKotiks(){  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 Owner owner = kotikiService.findOwnerByUsername(auth.getName());  
 List<KotikWrap> kotiks = kotikiService.findAllKotiks().stream().  
 map((kotik -> kotik.getKotikWrap())).  
 collect(Collectors.toList());  
 List<KotikWrap> kotiksWrap = new ArrayList<>();  
 for (KotikWrap kotik:kotiks) {  
 if(kotik.getOwner\_id() == owner.getId()) {  
 kotiksWrap.add(kotik);  
 }  
 }  
 return ResponseEntity.ok().body(kotiksWrap);  
  
 };  
  
 @PostMapping("/admin/kotik/findByColor")  
 public ResponseEntity<List<KotikWrap>>findKotikByColor(@RequestParam String color) {  
 List<Kotik> kotiki = kotikiService.findKotikByColor(color);  
 return new ResponseEntity<>(  
 kotiki.stream().  
 map((kotik -> kotik.getKotikWrap())).  
 collect(Collectors.toList()), HttpStatus.ACCEPTED);  
 }  
  
 @PostMapping("/user/kotik/findByColor")  
 public ResponseEntity<List<KotikWrap>>getUserKotiksByColor(@RequestParam String color){  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 Owner owner = kotikiService.findOwnerByUsername(auth.getName());  
 List<KotikWrap> result = new ArrayList<>();  
 List<Kotik> kotiki = kotikiService.findKotikByColor(color);  
 List<KotikWrap> kotiksWrap = new ArrayList<>();  
 for (Kotik kotik:kotiki) {  
 kotiksWrap.add(kotik.getKotikWrap());  
 }  
 for (KotikWrap kotik:kotiksWrap) {  
 if(kotik.getOwner\_id() == owner.getId()) {  
 result.add(kotik);  
 }  
 }  
 return ResponseEntity.ok().body(result);  
 }  
}  
  
  
  
Color.java  
  
package ru.itmo.kotikilab.entities;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public enum Color {  
 BLACK("black"),  
 WHITE("white"),  
 RED("red"),  
 GREY("grey"),  
 GREEN("green"),  
 YELLOW("yellow");  
  
 private final String name;  
  
 Color(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 private static final Map<String, Color> LOOKUP\_MAP = new HashMap<>();  
  
 static {  
 for (Color env : values()) {  
 LOOKUP\_MAP.put(env.getName(), env);  
 }  
 }  
  
 public static Color getTypeByName(String url) {  
 return LOOKUP\_MAP.get(url);  
 }  
}  
  
  
  
Friend.java  
  
package ru.itmo.kotikilab.entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import ru.itmo.kotikilab.wrapper.FriendWrap;  
  
import javax.persistence.\*;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table (name = "kotiki\_friends")  
public class Friend{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "kotik")  
 private Kotik kotik;  
 @OneToOne  
 @JoinColumn(name = "friend")  
 private Kotik friend;  
  
 public Friend(Kotik kotik, Kotik friend) {  
 this.kotik = kotik;  
 this.friend = friend;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Kotik getKotik() {  
 return kotik;  
 }  
  
 public void setKotik(Kotik kotik) {  
 this.kotik = kotik;  
 }  
  
 public Kotik getFriend() {  
 return friend;  
 }  
  
 public void setFriend(Kotik friend) {  
 this.friend = friend;  
 }  
  
 public FriendWrap getFriendWrap(){  
 return new FriendWrap(id, kotik.getId(), friend.getId());  
 }  
}  
  
  
Kotik.java  
  
package ru.itmo.kotikilab.entities;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import ru.itmo.kotikilab.tools.KotikiException;  
import ru.itmo.kotikilab.wrapper.KotikWrap;  
  
import javax.persistence.\*;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table (name = "kotiki")  
public class Kotik{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
  
 private String name;  
 private LocalDate birthday;  
 private String breed;  
  
 @Column(name = "color")  
 private Color color;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "owner")  
 private Owner owner;  
  
 @OneToMany(mappedBy = "kotik", orphanRemoval = true)  
 private final List<Friend> friends = new ArrayList<>();  
  
 public Kotik(String name, LocalDate birthday, String breed,  
 Color color, Owner owner) {  
 this.name = name;  
 this.birthday = birthday;  
 this.breed = breed;  
 this.color = color;  
 this.owner = owner;  
 }  
  
 public Friend addFriend(Kotik kotik) throws KotikiException {  
 if(kotik == null) {  
 throw new KotikiException("Sorry not found kotik");  
 }  
 if(kotik == this) {  
 throw new KotikiException("Sorry you can't be frendship with yourself I must to say your owner about your schiza");  
 }  
 Friend newFriend = new Friend(this, kotik);  
 friends.add(newFriend);  
 return newFriend;  
 }  
  
 public void removeFriends(Friend friend) {  
 friends.remove(friend);  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthDay() {  
 return birthday;  
 }  
  
 public void setBirthDay(LocalDate birthDay) {  
 this.birthday = birthDay;  
 }  
  
 public String getBreed() {  
 return breed;  
 }  
  
 public void setBreed(String breed) {  
 this.breed = breed;  
 }  
  
 public Color getColorId() {  
 return color;  
 }  
  
 public void setColorId(Color color) {  
 this.color = color;  
 }  
  
 public Owner getOwnerId() {  
 return owner;  
 }  
  
 public void setOwnerId(Owner owner) {  
 this.owner = owner;  
 }  
  
 public KotikWrap getKotikWrap(){  
 return new KotikWrap(id, name, birthday, breed, color.ordinal(), owner.getId());  
 }  
  
}  
  
  
  
Owner.java  
  
package ru.itmo.kotikilab.entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import ru.itmo.kotikilab.wrapper.OwnerWrap;  
  
import javax.persistence.\*;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table (name = "owners")  
public class Owner{  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 @Column(name = "name")  
 private String name;  
 private LocalDate birthday;  
 private String username;  
 private String password;  
 private String role;  
 @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL, orphanRemoval = true)  
 private List<Kotik> kotiki;  
  
 public Owner(String name, LocalDate birthday, String username, String password, String role) {  
 this.name = name;  
 this.birthday = birthday;  
 this.username = username;  
 this.password = password;  
 this.role = role;  
 kotiki = new ArrayList<>();  
 }  
  
 public void addKotik(Kotik kotik) {  
 kotik.setOwnerId(this);  
 kotiki.add(kotik);  
 }  
  
 public void removeKotik(Kotik kotik) {  
 kotiki.remove(kotik);  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public LocalDate getBirthDay() {  
 return birthday;  
 }  
  
 public void setBirthDay(LocalDate birthDay) {  
 this.birthday = birthDay;  
 }  
  
 public List<Kotik> getKotiki() {  
 return kotiki;  
 }  
  
 public List<Integer> getKotikiId(){  
 List<Integer> kotikiId = new ArrayList<>();  
 for (Kotik item: kotiki) {  
 kotikiId.add(item.getId());  
 }  
 return kotikiId;  
 }  
 public LocalDate getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(LocalDate birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getRole() {  
 return role;  
 }  
  
 public void setRole(String role) {  
 this.role = role;  
 }  
  
 public void setKotiki(List<Kotik> kotiki) {  
 this.kotiki = kotiki;  
 }  
  
  
 public OwnerWrap getOwnerWrap(){  
 return new OwnerWrap(id, name, birthday, username, password, role, this.getKotikiId());  
 }  
}  
  
  
  
Role.java  
  
package ru.itmo.kotikilab.entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.util.List;  
  
@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table(name = "roles")  
public class Role {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 private String name;  
 //@OneToMany(mappedBy = "role", cascade = CascadeType.ALL, orphanRemoval = true)  
 //private List<Owner> owners;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
// public List<Owner> getOwners() {  
// return owners;  
// }  
//  
// public void setOwners(List<Owner> owners) {  
// this.owners = owners;  
// }  
  
 public Role(String name) {  
 this.name = name;  
 }  
  
  
  
}  
  
  
  
FriendRepository.java  
  
package ru.itmo.kotikilab.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import ru.itmo.kotikilab.entities.Friend;  
  
@Repository  
public interface FriendRepository extends JpaRepository<Friend, Integer> {  
}  
  
  
  
KotikRepository.java  
  
package ru.itmo.kotikilab.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import ru.itmo.kotikilab.entities.Color;  
import ru.itmo.kotikilab.entities.Kotik;  
  
import java.util.List;  
  
@Repository  
public interface KotikRepository extends JpaRepository<Kotik, Integer> {  
 List<Kotik> findByColor(Color color);  
}  
  
  
  
OwnerRepository.java  
  
package ru.itmo.kotikilab.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import ru.itmo.kotikilab.entities.Owner;  
  
@Repository  
public interface OwnerRepository extends JpaRepository<Owner, Integer> {  
 Owner findByUsername(String username);  
}  
  
  
  
RoleRepository.java  
  
package ru.itmo.kotikilab.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import ru.itmo.kotikilab.entities.Role;  
  
@Repository  
public interface RoleRepository extends JpaRepository<Role, Integer> {  
 Role findById(int id);  
}  
  
  
  
CustomAuthencationProvider.java  
  
package ru.itmo.kotikilab.security;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.authentication.AuthenticationProvider;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.stereotype.Component;  
import ru.itmo.kotikilab.entities.Owner;  
import ru.itmo.kotikilab.repository.OwnerRepository;  
import ru.itmo.kotikilab.tools.KotikiException;  
  
@Component  
public class CustomAuthencationProvider implements AuthenticationProvider {  
 @Autowired  
 private OwnerRepository ownerRepo;  
  
 @Override  
 public Authentication authenticate(Authentication authentication) throws AuthenticationException {  
 String username = authentication.getName();  
 String password = authentication.getCredentials().toString();  
  
 Owner owner = ownerRepo.findByUsername(username);  
 if(owner == null) {  
 throw new KotikiException("Unknown owner" + username);  
 }  
 if(!password.equals(owner.getPassword())) {  
 throw new KotikiException("Bad password");  
 }  
 UserDetails principal = User.builder()  
 .username(owner.getUsername())  
 .password(owner.getPassword())  
 .roles(owner.getRole())  
 .build();  
 return new UsernamePasswordAuthenticationToken(principal, password, principal.getAuthorities());  
  
 }  
  
 @Override  
 public boolean supports(Class<?> authentication) {  
 return authentication.equals(UsernamePasswordAuthenticationToken.class);  
 }  
}  
  
  
  
SecurityConfig.java  
  
package ru.itmo.kotikilab.security;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
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.crypto.password.PasswordEncoder;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import ru.itmo.kotikilab.services.CustomUserDetailsService;  
  
@EnableWebSecurity(debug = true)  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Autowired  
 private CustomUserDetailsService userDetailsService;  
  
 @Autowired  
 private CustomAuthencationProvider customAuthencationProvider;  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return NoOpPasswordEncoder.getInstance();  
 }  
  
 @Override  
 public void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userDetailsService);  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests()  
 .antMatchers("/user/\*\*").hasAnyRole("user", "admin")  
 .antMatchers("/admin/\*\*").hasRole("admin")  
 .antMatchers("/").permitAll()  
 .and()  
 .formLogin();  
 http.csrf().disable();  
 }  
  
  
}  
  
  
  
CustomUserDetailsService.java  
  
package ru.itmo.kotikilab.services;  
  
import org.springframework.beans.factory.annotation.Autowired;  
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.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
import ru.itmo.kotikilab.entities.Owner;  
import ru.itmo.kotikilab.repository.OwnerRepository;  
  
@Service  
public class CustomUserDetailsService implements UserDetailsService {  
 @Autowired  
 private OwnerRepository ownerRepo;  
  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 Owner owner = ownerRepo.findByUsername(username);  
 if (owner == null) {  
 throw new UsernameNotFoundException("Unknown user: "+ username);  
 }  
 UserDetails user = User.builder()  
 .username(owner.getUsername())  
 .password(owner.getPassword())  
 .roles(owner.getRole())  
 .build();  
 return user;  
 }  
}  
  
  
  
MainService.java  
  
package ru.itmo.kotikilab.services;  
  
import ru.itmo.kotikilab.entities.\*;  
  
import java.time.LocalDate;  
import java.util.List;  
  
public interface MainService {  
 void createOwner(String name, String date, String username, String password, String role);  
 Owner findOwnerById(int id);  
 Owner findOwnerByUsername(String username);  
 Owner saveOwner(Owner owner);  
 void deleteOwner(Owner owner);  
 List<Owner> findAllOwners();  
 Role saveRole(Role role);  
  
 Color getColor(int color) throws Exception;  
 Kotik createKotik(String name, String date, String breed, int color, int ownerId) throws Exception;  
 Kotik findKotikById(int id);  
 Kotik saveKotik(Kotik kotik);  
 void deleteKotik(Kotik kotik);  
 List<Kotik> findAllKotiks();  
 List<Kotik> findKotikByColor(String color);  
  
 Friend findFriendById(int id);  
 Friend saveFriend(Friend friend);  
 void deleteFriend(Friend friend);  
 List<Friend> findAllFriends();  
  
 void addOwnerToKotik(int idOwner, int idKotik);  
 Friend addFriends(int idKotik, int idFriend);  
}  
  
  
  
MainServiceImpl.java  
  
package ru.itmo.kotikilab.services;  
  
import lombok.Data;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import ru.itmo.kotikilab.entities.\*;  
import ru.itmo.kotikilab.repository.\*;  
import ru.itmo.kotikilab.tools.KotikiException;  
  
import javax.transaction.Transactional;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.List;  
  
@Service  
@RequiredArgsConstructor  
@Transactional  
@Slf4j  
public class MainServiceImpl implements MainService {  
 private final KotikRepository kotikRepo;  
 private final OwnerRepository ownerRepo;  
 private final FriendRepository friendRepo;  
 private final RoleRepository roleRepo;  
  
 @Override  
 public void createOwner(String name, String date, String username, String password, String role){  
 log.info("Create owner");  
 Owner owner = new Owner(name, LocalDate.parse(date), username, password, role);  
 ownerRepo.save(owner);  
 }  
  
 @Override  
 public Owner findOwnerById(int id) {  
 log.info("Finding owner by id");  
 return ownerRepo.getById(id);  
 }  
  
 @Override  
 public Owner findOwnerByUsername(String username) {  
 log.info("Finding owner by username");  
 return ownerRepo.findByUsername(username);  
 }  
  
 @Override  
 public Owner saveOwner(Owner owner) {  
 log.info("Saving new owner");  
 return ownerRepo.save(owner);  
 }  
  
 @Override  
 public void deleteOwner(Owner owner) {  
 log.info("Delete owner");  
 ownerRepo.delete(owner);  
 }  
  
 @Override  
 public List<Owner> findAllOwners() {  
 log.info("Fetching all owners");  
 return ownerRepo.findAll();  
 }  
  
 @Override  
 public Color getColor(int color) throws Exception{  
 for (Color colors: Color.values()) {  
 if (colors.ordinal() == color) {  
 return colors;  
 }  
 }  
 throw new Exception("Error");  
 }  
  
 @Override  
 public Role saveRole(Role role) {  
 log.info("Saving new role");  
 return roleRepo.save(role);  
 }  
  
 @Override  
 public Kotik createKotik(String name, String date, String breed, int color, int ownerId) throws Exception {  
 log.info("Create kotik");  
 Color colorEnum = getColor(color);  
 Owner owner = findOwnerById(ownerId);  
 Kotik kotik = new Kotik(name, LocalDate.parse(date), breed, colorEnum, owner);  
 ownerRepo.save(owner);  
 kotikRepo.save(kotik);  
 return kotik;  
 }  
  
 @Override  
 public Kotik findKotikById(int id) {  
 log.info("Finding kotik by id");  
 return kotikRepo.getById(id);  
 }  
  
 @Override  
 public Kotik saveKotik(Kotik kotik) {  
 log.info("Saving new kotik");  
 return kotikRepo.save(kotik);  
 }  
  
 @Override  
 public void deleteKotik(Kotik kotik) {  
 log.info("Delete kotik");  
 kotikRepo.delete(kotik);  
 }  
  
 @Override  
 public List<Kotik> findAllKotiks() {  
 log.info("Fetching all kotiks");  
 return kotikRepo.findAll();  
 }  
  
 @Override  
 public List<Kotik> findKotikByColor(String color) {  
 log.info("Find kotik by color");  
 Color colorEnum = Color.getTypeByName(color);  
 List<Kotik> kotiki = kotikRepo.findByColor(colorEnum);  
 return kotiki;  
 }  
  
 @Override  
 public Friend findFriendById(int id) {  
 log.info("Finding friend by id");  
 return friendRepo.getById(id);  
 }  
  
 @Override  
 public Friend saveFriend(Friend friend) {  
 log.info("Saving new friend");  
 return friendRepo.save(friend);  
 }  
  
 @Override  
 public void deleteFriend(Friend friend) {  
 log.info("Delete friend");  
 friendRepo.delete(friend);  
 }  
  
 @Override  
 public List<Friend> findAllFriends() {  
 log.info("Fetching all friends");  
 return friendRepo.findAll();  
 }  
  
 @Override  
 public void addOwnerToKotik(int idOwner, int idKotik) {  
 log.info("Adding owner to kotik");  
 Owner owner = findOwnerById(idOwner);  
 Kotik kotik = findKotikById(idKotik);  
 owner.addKotik(kotik);  
 kotik.setOwnerId(owner);  
 ownerRepo.save(owner);  
 kotikRepo.save(kotik);  
 }  
  
 @Override  
 public Friend addFriends(int idKotik, int idFriend) {  
 Kotik kotik = findKotikById(idKotik);  
 Kotik kotikFriend = findKotikById(idFriend);  
 Friend newFriend = kotik.addFriend(kotikFriend);  
 friendRepo.save(newFriend);  
 return newFriend;  
 }  
}  
  
  
  
KotikiException.java  
  
package ru.itmo.kotikilab.tools;  
public class KotikiException extends RuntimeException {  
 public KotikiException() {  
 }  
  
 public KotikiException(String message) {  
 super(message);  
 }  
}  
  
  
  
FriendWrap.java  
  
package ru.itmo.kotikilab.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
  
@Data  
@AllArgsConstructor  
public class FriendWrap {  
 public int id;  
 public int kotikId;  
 public int friendId;  
}  
  
  
  
KotikWrap.java  
  
package ru.itmo.kotikilab.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import ru.itmo.kotikilab.entities.Color;  
import java.time.LocalDate;  
  
@Data  
@AllArgsConstructor  
public class KotikWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String breed;  
 private int colorId;  
 private int owner\_id;  
}  
  
  
  
OwnerWrap.java  
  
package ru.itmo.kotikilab.wrapper;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import ru.itmo.kotikilab.entities.Kotik;  
  
import java.time.LocalDate;  
import java.util.List;  
  
@Data  
@AllArgsConstructor  
public class OwnerWrap {  
 private int id;  
 private String name;  
 private LocalDate birthday;  
 private String username;  
 private String password;  
 private String role;  
 private List<Integer> kotikiId;  
}  
  
  
  
KotikiLabApplicationTests.java  
  
package ru.itmo.kotikilab;  
  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
class KotikiLabApplicationTests {  
  
 @Test  
 void contextLoads() {  
 }  
  
}