Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабороторная работа №4

Выполнил студент:

Моторина Евгения Викторовна
группа: M32051

Проверил:

Чикишев Константин Максимович

Санкт-Петербург,
2022 г.

1.1. Текст задания

4 лабораторная
Владельцы недовольны, что информацию о котиках может получить кто угодно. В этой лабораторной мы добавим авторизацию к сервису.
Добавляется роль администратора. Он имеет доступ ко всем методам и может создавать новых пользователей. Пользователь связан с владельцем в соотношении 1:1.
Методы по получению информации и котиках и владельцах должны быть защищены Spring Security. Доступ к соответствующим endpoint'ам имеют только владельцы котиков и администраторы. Доступ к методам для фильтрации имеют все авторизованные пользователи, но на выходе получают только данные о своих котиках.
Внимание: эндпоинты, созданные на предыдущем этапе, не должны быть удалены.

1.2. Решение

Листинг 1.1: KotikiApplication.java

```
1  package ru.itmo;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.ImportResource;
7  import org.springframework.security.crypto.bcrypt.
       BCryptPasswordEncoder;
8
9  @SpringBootApplication
10 public class KotikiApplication {
11
12     public static void main(String[] args) {
13         SpringApplication.run(KotikiApplication.class, args);
14         //new BCryptPasswordEncoder().encode("123");
15     }
16
17
18 }
```

Листинг 1.2: AdminController.java

```java
package ru.itmo.api;

import lombok.AllArgsConstructor;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import ru.itmo.models.Owner;
import ru.itmo.models.Role;
import ru.itmo.service.OwnerService;
import ru.itmo.service.RoleService;
import ru.itmo.service.UserService;

@RestController
@RequestMapping("/admin")
@AllArgsConstructor
public class AdminController {
    private final UserService userService;
    private final OwnerService ownerService;
    private final RoleService roleService;

    @PostMapping("/add")
    public ResponseEntity<?> addUser(@RequestParam String username,
   @RequestParam String password,
                                     @RequestParam int
   roleId, @RequestParam int ownerId) {
        Role role = roleService.getRoleById(roleId);
        Owner owner = ownerService.getOwnerById(ownerId);
        try {
            return ResponseEntity.ok().body(userService.createUser(
   username, password, role, owner).getWrapper());
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(
   e.getMessage());
        }
    }

}
```

Листинг 1.3: CatController.java

```java
package ru.itmo.api;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PostAuthorize;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.*;
import ru.itmo.models.Cat;
import ru.itmo.models.CustomUser;
import ru.itmo.models.Owner;
import ru.itmo.serviceImpl.CatServiceImpl;
import ru.itmo.serviceImpl.OwnerServiceImpl;
import ru.itmo.serviceImpl.UserServiceImpl;
import ru.itmo.wrapper.CatWrapper;

import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/cat")
@RequiredArgsConstructor
public class CatController {
    private final CatServiceImpl catService;
    private final OwnerServiceImpl ownerService;
    private final UserServiceImpl userService;

    @PostMapping("/create")
    public ResponseEntity<?> createCat(@RequestParam String name, @RequestParam String birthday,
                                       @RequestParam String breed, @RequestParam int colorId,
                                       @RequestParam int ownerId, Authentication authentication) {
        Owner owner = ownerService.getOwnerById(ownerId);
        CustomUser user = userService.getUserByUsername(authentication.getName());
        if (user.getOwner().getId() == ownerId) {
            return ResponseEntity.ok().body(catService.createCat(name, birthday, breed, colorId, owner).getWrapper());
        }
        return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("You can not create cat that is owned by another user.");
    }

    @GetMapping("/getById")
```

```
43      public ResponseEntity<?> getCat(@RequestParam int catId,
    Authentication authentication) {
44          int ownerId = userService.getUserByUsername(authentication.
    getName()).getOwner().getId();
45          Cat cat = catService.getCatById(catId);
46          if (ownerId == cat.getOwnerId()) {
47              return ResponseEntity.ok().body(cat.getWrapper());
48          }
49          return ResponseEntity.status(HttpStatus.BAD_REQUEST)
50                  .body("You can not get info about stranger cat.");
51      }
52
53      @GetMapping("/all")
54      public ResponseEntity<List<CatWrapper>> getCats(Authentication
    authentication) {
55          int ownerId = userService.getUserByUsername(authentication.
    getName()).getOwner().getId();
56          return new ResponseEntity<>(
57                  catService.getCats()
58                          .stream()
59                          .filter(cat -> cat.getOwnerId() == ownerId)
60                          .map((Cat::getWrapper)).
61                          collect(Collectors.toList()),
62                  HttpStatus.ACCEPTED);
63      }
64
65      @DeleteMapping("/delete")
66      public ResponseEntity<?> deleteCat(@RequestParam int catId,
    Authentication authentication) {
67          CustomUser user = userService.getUserByUsername(authentication
    .getName());
68          Cat cat = catService.getCatById(catId);
69          if (user.getOwner().getId() == cat.getOwnerId()) {
70              return ResponseEntity.ok().body(catService.deleteCat(cat).
    getWrapper());
71          }
72          return ResponseEntity.status(HttpStatus.BAD_REQUEST)
73                  .body("You can not delete stranger cat.");
74      }
75  }
```

Листинг 1.4: FriendshipController.java

```java
package ru.itmo.api;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import ru.itmo.models.Friendship;
import ru.itmo.serviceImpl.FriendshipServiceImpl;
import ru.itmo.wrapper.FriendshipWrapper;

import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/friendship")
@RequiredArgsConstructor
public class FriendshipController {
    private final FriendshipServiceImpl friendshipService;

    @PostMapping("/create")
    public ResponseEntity<FriendshipWrapper> createFriendship(
    @RequestParam int idFirstCat, @RequestParam int idSecondCat) {
        return ResponseEntity.ok().body(friendshipService.
    createFriendship(idFirstCat, idSecondCat).getWrapper());
    }

    @GetMapping("/getById")
    public ResponseEntity<FriendshipWrapper> getFriendship(
    @RequestParam int friendshipId) {
        return ResponseEntity.ok().body(friendshipService.
    getFriendshipById(friendshipId).getWrapper());
    }

    @GetMapping("/all")
    public ResponseEntity<List<FriendshipWrapper>> getFriendships() {
        return new ResponseEntity<>(
                friendshipService.getFriendships().stream().map((
    Friendship::getWrapper)).
                        collect(Collectors.toList()), HttpStatus.
    ACCEPTED);
    }

    @DeleteMapping("/delete")
    public ResponseEntity<FriendshipWrapper> deleteFriendship(
    @RequestParam int friendshipId) {
        Friendship friendship = friendshipService.getFriendshipById(
    friendshipId);
        return ResponseEntity.ok().body(friendshipService.
    deleteFriendship(friendship).getWrapper());
```

```
41         }
42 }
```

Листинг 1.5: OwnerController.java

```java
package ru.itmo.api;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import ru.itmo.models.Owner;
import ru.itmo.serviceImpl.CatServiceImpl;
import ru.itmo.serviceImpl.OwnerServiceImpl;
import ru.itmo.wrapper.OwnerWrapper;

import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/owner")
@RequiredArgsConstructor
public class OwnerController {
    private final OwnerServiceImpl ownerService;
    private final CatServiceImpl catService;

    @PostMapping("/create")
    public ResponseEntity<OwnerWrapper> createOwner(@RequestParam String name, @RequestParam String birthday) {
        return ResponseEntity.ok().body(ownerService.createOwner(name, birthday).getWrapper());
    }

    @PostMapping("/addCatToOwner")
    public ResponseEntity<?> addCatToOwner(@RequestParam int catId, @RequestParam int ownerId) {
        ownerService.addCatToOwner(catId, ownerId);
        return ResponseEntity.ok().build();
    }

    @GetMapping("/getById")
    public ResponseEntity<OwnerWrapper> getOwner(@RequestParam int ownerId) {
        return ResponseEntity.ok().body(ownerService.getOwnerById(ownerId).getWrapper());
    }

    @GetMapping("/all")
    public ResponseEntity<List<OwnerWrapper>> getOwners() {
        return new ResponseEntity<>(
                ownerService.getOwners().stream().map((Owner::getWrapper)).
                        collect(Collectors.toList()), HttpStatus.ACCEPTED);
```

```
43      }
44
45      @DeleteMapping("/delete")
46      public ResponseEntity<OwnerWrapper> deleteOwner(@RequestParam int
    ownerId) {
47          Owner owner = ownerService.getOwnerById(ownerId);
48          return ResponseEntity.ok().body(ownerService.deleteOwner(owner
    ).getWrapper());
49      }
50
51  }
```

Листинг 1.6: UserController.java

```java
package ru.itmo.api;

import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.*;
import ru.itmo.models.Cat;
import ru.itmo.models.CustomUser;
import ru.itmo.models.Owner;
import ru.itmo.models.Role;
import ru.itmo.repository.UserRepository;
import ru.itmo.service.OwnerService;
import ru.itmo.service.UserService;
import ru.itmo.wrapper.CatWrapper;
import ru.itmo.wrapper.UserWrapper;

import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/user")
@RequiredArgsConstructor
public class UserController {
    @Autowired
    private UserService userService;
    @Autowired
    private OwnerService ownerService;
    @Autowired
    private UserRepository userRepository;

    @GetMapping("/")
    public String greeting() {
        return "You have successfully logged in.";
    }

    @GetMapping("/getCats")
    public ResponseEntity<List<CatWrapper>> getCats(int userId) {
        Owner owner = ownerService.getOwnerById(userService.getUserById(userId).getOwner().getId());
        return new ResponseEntity<>(owner.getCats().stream().map((Cat::getWrapper)).
                collect(Collectors.toList()), HttpStatus.ACCEPTED);
    }

}
```

Листинг 1.7: Cat.java

```
1  package ru.itmo.models;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import ru.itmo.wrapper.CatWrapper;
7
8  import javax.persistence.*;
9  import java.time.LocalDate;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 @Entity
14 @Data
15 @NoArgsConstructor
16 @AllArgsConstructor
17 @Table(name = "cats")
18 public class Cat {
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     private int id;
22
23     @Column(name = "name")
24     private String name;
25
26     @Column(name = "birthday")
27     private LocalDate birthday;
28
29     @Column(name = "breed")
30     private String breed;
31
32     @Column(name = "color_id")
33     private int colorId;
34
35     @ManyToOne(fetch = FetchType.LAZY)
36     @JoinColumn(name = "owner_id")
37     private Owner owner;
38
39     @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
40     @JoinTable(name = "friends", joinColumns = @JoinColumn(name =
       "id_first_cat"), inverseJoinColumns = @JoinColumn(name =
       "id_second_cat"))
41     private List<Cat> friends;
42
43     public Cat(String name, LocalDate birthday, String breed, int
       colorId, Owner owner) {
44         this.name = name;
45         this.birthday = birthday;
46         this.breed = breed;
```

```
47          this.colorId = colorId;
48          this.owner = owner;
49          this.friends = new ArrayList<>();
50      }
51
52      public int getId() {
53          return id;
54      }
55
56      public void setId(int id) {
57          this.id = id;
58      }
59
60      public String getName() {
61          return name;
62      }
63
64      public void setName(String name) {
65          this.name = name;
66      }
67
68      public LocalDate getBirthday() {
69          return birthday;
70      }
71
72      public void setBirthday(LocalDate birthday) {
73          this.birthday = birthday;
74      }
75
76      public String getBreed() {
77          return breed;
78      }
79
80      public void setBreed(String breed) {
81          this.breed = breed;
82      }
83
84      public int getColorId() {
85          return colorId;
86      }
87
88      public void setColorId(int colorId) {
89          this.colorId = colorId;
90      }
91
92      public Owner getOwner() {
93          return owner;
94      }
95
96      public void setOwner(Owner owner) {
```

```
97          this.owner = owner;
98      }
99
100     public int getOwnerId() {
101         return owner.getId();
102     }
103
104     public List<Integer> getFriendsId() {
105         List<Integer> friendsId = new ArrayList<>();
106         for (Cat friend: friends) {
107             friendsId.add(friend.id);
108         }
109         return friendsId;
110     }
111
112     public CatWrapper getWrapper() {
113         return new CatWrapper(id, name, birthday, breed, colorId, this
    .getOwnerId(), this.getFriendsId());
114     }
115 }
```

Листинг 1.8: Color.java

```
package ru.itmo.models;

public enum Color {
    BLACK,
    WHITE,
    GRAY,
    CALICO;

    public Color getColorById(int id) throws Exception {
        for (Color color: Color.values()) {
            if (color.ordinal() == id) {
                return color;
            }
        }
        throw new Exception("No color with this id");
    }
}
```

Листинг 1.9: CustomUser.java

```
1  package ru.itmo.models;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import org.springframework.security.core.GrantedAuthority;
6  import org.springframework.security.core.userdetails.UserDetails;
7  import ru.itmo.wrapper.UserWrapper;
8
9  import javax.persistence.*;
10 import java.util.*;
11
12 @Entity
13 @Data
14 @AllArgsConstructor
15 @Table(name = "users")
16 public class CustomUser implements UserDetails {
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private int id;
20
21     @Column(name = "username")
22     private String username;
23
24     @Column(name = "password")
25     private String password;
26
27     @ManyToOne(fetch = FetchType.EAGER)
28     @JoinColumn(name = "role_id")
29     private Role role;
30
31     @OneToOne(fetch = FetchType.LAZY)
32     @JoinColumn(name = "owner_id")
33     private Owner owner;
34
35     public CustomUser(String username, String password, Role role,
       Owner owner) {
36         this.username = username;
37         this.password = password;
38         this.role = role;
39         this.owner = owner;
40     }
41
42     public CustomUser() {
43
44     }
45
46     public String getUsername() {
47         return username;
48     }
```

```
49
50      @Override
51      public boolean isAccountNonExpired() {
52          return true;
53      }
54
55      @Override
56      public boolean isAccountNonLocked() {
57          return true;
58      }
59
60      @Override
61      public boolean isCredentialsNonExpired() {
62          return true;
63      }
64
65      @Override
66      public boolean isEnabled() {
67          return true;
68      }
69
70      @Override
71      public Collection<? extends GrantedAuthority> getAuthorities() {
72          return null;
73      }
74
75      public String getPassword() {
76          return password;
77      }
78
79      public Role getRole() {
80          return role;
81      }
82
83      public Owner getOwner() {
84          return owner;
85      }
86
87      public UserWrapper getWrapper() {
88          return new UserWrapper(id, username, password, role.getId(),
    owner.getId());
89      }
90
91  }
```

Листинг 1.10: Friendship.java

```
package ru.itmo.models;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import ru.itmo.wrapper.CatWrapper;
import ru.itmo.wrapper.FriendshipWrapper;

import javax.persistence.*;
import java.time.LocalDate;
import java.util.List;

@Entity @Data
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "friends")
public class Friendship {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "id_first_cat")
    private int idFirstCat;

    @Column(name = "id_second_cat")
    private int idSecondCat;

    public Friendship(int idFirstCat, int idSecondCat) {
        this.idFirstCat = idFirstCat;
        this.idSecondCat = idSecondCat;
    }
//@ManyToMany(mappedBy = "friends")
    //private List<Cat> cats;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public FriendshipWrapper getWrapper() {
        return new FriendshipWrapper(id, idFirstCat, idSecondCat);
    }

}
```

Листинг 1.11: Owner.java

```
1  package ru.itmo.models;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import ru.itmo.wrapper.CatWrapper;
7  import ru.itmo.wrapper.OwnerWrapper;
8
9  import javax.persistence.*;
10 import java.time.LocalDate;
11 import java.util.ArrayList;
12 import java.util.HashSet;
13 import java.util.List;
14 import java.util.Set;
15
16 @Entity
17 @Data
18 @NoArgsConstructor
19 @AllArgsConstructor
20 @Table (name = "owners")
21 public class Owner {
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     private int id;
25
26     @Column
27     private String name;
28
29     @Column
30     private LocalDate birthday;
31
32     @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL,
       orphanRemoval = true)
33     private List<Cat> cats;
34
35     public void addCat(Cat cat) {
36         cat.setOwner(this);
37         cats.add(cat);
38     }
39
40     public Owner(String name, LocalDate birthday) {
41         this.name = name;
42         this.birthday = birthday;
43         this.cats = new ArrayList<>() {
44         };
45     }
46
47     public int getId() {
48         return id;
```

```
49      }
50
51      public String getName() {
52          return name;
53      }
54
55      public void setName(String name) {
56          this.name = name;
57      }
58
59      public LocalDate getBirthday() {
60          return birthday;
61      }
62
63      public void setBirthday(LocalDate birthday) {
64          this.birthday = birthday;
65      }
66
67      public List<Cat> getCats() {
68          return cats;
69      }
70
71      public List<Integer> getCatsId() {
72          List<Integer> catsId = new ArrayList<>();
73          for (Cat cat: cats) {
74              catsId.add(cat.getId());
75          }
76          return catsId;
77      }
78
79      public OwnerWrapper getWrapper() {
80          return new OwnerWrapper(id, name, birthday, this.getCatsId());
81      }
82  }
```

Листинг 1.12: Role.java

```
package ru.itmo.models;

import lombok.AllArgsConstructor;
import lombok.Data;
import org.springframework.security.core.GrantedAuthority;

import javax.persistence.*;
import java.util.List;
import java.util.Set;

@Entity
@Data
@AllArgsConstructor
@Table(name = "roles")
public class Role implements GrantedAuthority {
    @Id
    private int id;

    private String name;

    @OneToMany(mappedBy = "role", cascade = CascadeType.ALL,
   orphanRemoval = true)
    private List<CustomUser> users;

    public Role() {
    }

    public Role(String name) {
        this.name = name;
    }

    public int getId() {
        return id;
    }


    public String getName() {
        return name;
    }

    public List<CustomUser> getUsers() {
        return users;
    }
    @Override
    public String getAuthority() {
        return getName();
    }

}
```

Листинг 1.13: CatRepository.java

```
1  package ru.itmo.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.models.Cat;
5
6  public interface CatRepository extends JpaRepository<Cat, Integer> {
7      Cat save(Cat cat);
8      Cat findById(int id);
9      void delete(Cat cat);
10 }
```

Листинг 1.14: FriendshipRepository.java

```
1  package ru.itmo.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.models.Friendship;
5
6  import java.util.List;
7
8  public interface FriendshipRepository extends JpaRepository<Friendship
       , Integer> {
9      Friendship save(Friendship friendship);
10     Friendship findById(int id);
11     List<Friendship> findAll();
12     void delete(Friendship friendship);
13 }
```

Листинг 1.15: OwnerRepository.java

```
1  package ru.itmo.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.models.Owner;
5
6  import java.util.List;
7
8  public interface OwnerRepository extends JpaRepository<Owner, Integer>
       {
9      Owner save(Owner owner);
10     Owner findById(int id);
11     List<Owner> findAll();
12     void delete(Owner owner);
13 }
```

Листинг 1.16: RoleRepository.java

```
1 package ru.itmo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import ru.itmo.models.CustomUser;
5 import ru.itmo.models.Role;
6
7 public interface RoleRepository extends JpaRepository<Role, Integer> {
8     Role findById(int id);
9 }
```

Листинг 1.17: UserRepository.java

```
1 package ru.itmo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import ru.itmo.models.CustomUser;
5
6 public interface UserRepository extends JpaRepository<CustomUser,
    Integer> {
7     CustomUser findByUsername(String username);
8     CustomUser findById(int userId);
9 }
```

Листинг 1.18: SecurityConfig.java

```
1  package ru.itmo.security;
2
3  import org.springframework.beans.factory.annotation.Autowired;
4  import org.springframework.context.annotation.Bean;
5  import org.springframework.security.authentication.dao.
       DaoAuthenticationProvider;
6  import org.springframework.security.config.annotation.authentication.
       builders.AuthenticationManagerBuilder;
7  import org.springframework.security.config.annotation.web.builders.
       HttpSecurity;
8  import org.springframework.security.config.annotation.web.
       configuration.EnableWebSecurity;
9  import org.springframework.security.config.annotation.web.
       configuration.WebSecurityConfigurerAdapter;
10 import org.springframework.security.crypto.bcrypt.
       BCryptPasswordEncoder;
11 import org.springframework.security.crypto.password.PasswordEncoder;
12 import ru.itmo.service.UserService;
13
14 @EnableWebSecurity(debug = true)
15 public class SecurityConfig extends WebSecurityConfigurerAdapter {
16     @Autowired
17     private UserService userService;
18     @Bean
19     public PasswordEncoder encoder() {
20         return new BCryptPasswordEncoder();
21     }
22
23     @Override
24     public void configure(AuthenticationManagerBuilder auth) throws
       Exception {
25         auth.userDetailsService(userService).passwordEncoder(new
       BCryptPasswordEncoder());
26     }
27
28     @Bean
29     public DaoAuthenticationProvider authProvider() {
30         DaoAuthenticationProvider authProvider = new
       DaoAuthenticationProvider();
31         authProvider.setUserDetailsService(userService);
32         authProvider.setPasswordEncoder(encoder());
33         return authProvider;
34     }
35
36     @Override
37     protected void configure(HttpSecurity http) throws Exception {
38         http
39                 .csrf().disable()
40                 .authorizeRequests()
```

```
41                      .antMatchers("/admin/**").hasAuthority("ROLE_ADMIN")
42                      .antMatchers("/user/**").hasAnyAuthority("ROLE_ADMIN",
     "ROLE_USER")
43                      .antMatchers("/**").authenticated()
44                      .and().formLogin().loginPage("/login").
    defaultSuccessUrl("/user/");
45      }
46  }
```

Листинг 1.19: CatService.java

```
1  package ru.itmo.service;
2
3  import ru.itmo.models.Cat;
4  import ru.itmo.models.Owner;
5
6  import java.util.List;
7
8  public interface CatService {
9      Cat createCat(String name, String birthday, String breed, int
       colorId, Owner owner);
10     Cat saveCat(Cat cat);
11     Cat getCatById(int id);
12     List<Cat> getCats();
13     Cat deleteCat(Cat cat);
14 }
```

Листинг 1.20: FriendshipService.java

```
1  package ru.itmo.service;
2
3  import ru.itmo.models.Friendship;
4
5  import java.util.List;
6
7  public interface FriendshipService {
8      Friendship createFriendship(int idFirstCat, int idSecondCat);
9      Friendship saveFriendship(Friendship friendship);
10     Friendship getFriendshipById(int id);
11     List<Friendship> getFriendships();
12     Friendship deleteFriendship(Friendship friendship);
13 }
```

Листинг 1.21: OwnerService.java

```
1  package ru.itmo.service;
2
3  import ru.itmo.models.Owner;
4
5  import java.util.List;
6
7  public interface OwnerService {
8      Owner createOwner(String name, String birthday);
9      Owner saveOwner(Owner owner);
10     void addCatToOwner(int catId, int ownerId);
11     Owner getOwnerById(int ownerId);
12     List<Owner> getOwners();
13     Owner deleteOwner(Owner owner);
14 }
```

Листинг 1.22: RoleService.java

```
1 package ru.itmo.service;
2
3 import ru.itmo.models.Role;
4
5 public interface RoleService {
6     Role getRoleById(int roleId);
7 }
```

Листинг 1.23: UserService.java

```
1  package ru.itmo.service;
2
3  import org.springframework.security.core.userdetails.UserDetails;
4  import org.springframework.security.core.userdetails.
       UserDetailsService;
5  import org.springframework.security.core.userdetails.
       UsernameNotFoundException;
6  import org.springframework.stereotype.Service;
7  import ru.itmo.models.CustomUser;
8  import ru.itmo.models.Owner;
9  import ru.itmo.models.Role;
10
11 @Service
12 public interface UserService extends UserDetailsService {
13     CustomUser createUser(String username, String password, Role role,
       Owner owner) throws Exception;
14     UserDetails loadUserByUsername(String username) throws
       UsernameNotFoundException;
15     CustomUser saveUser(CustomUser user);
16     CustomUser getUserById(int userId);
17     CustomUser getUserByUsername(String username);
18 }
```

Листинг 1.24: CatServiceImpl.java

```
1  package ru.itmo.serviceImpl;
2
3  import lombok.RequiredArgsConstructor;
4  import lombok.extern.slf4j.Slf4j;
5  import org.springframework.stereotype.Service;
6  import ru.itmo.models.Cat;
7  import ru.itmo.models.Owner;
8  import ru.itmo.repository.CatRepository;
9  import ru.itmo.service.CatService;
10
11 import javax.transaction.Transactional;
12 import java.time.LocalDate;
13 import java.time.format.DateTimeFormatter;
14 import java.util.List;
15
16 @Service
17 @RequiredArgsConstructor
18 @Transactional
19 @Slf4j
20 public class CatServiceImpl implements CatService {
21     private final CatRepository catRepository;
22
23     @Override
24     public Cat createCat(String name, String birthday, String breed,
    int colorId, Owner owner) {
25         return catRepository.save(new Cat(name, LocalDate.parse(
    birthday,
26                 DateTimeFormatter.ofPattern("yyyy.MM.dd")), breed,
    colorId, owner));
27     }
28
29     @Override
30     public Cat saveCat(Cat cat) {
31         return catRepository.save(cat);
32     }
33
34     @Override
35     public Cat getCatById(int id) {
36         return catRepository.findById(id);
37     }
38
39     @Override
40     public List<Cat> getCats() {
41         return catRepository.findAll();
42     }
43
44     @Override
45     public Cat deleteCat(Cat cat) {
46         catRepository.delete(cat);
```

```
47            return cat;
48        }
49    }
```

Листинг 1.25: FriendshipServiceImpl.java

```java
package ru.itmo.serviceImpl;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import ru.itmo.models.Friendship;
import ru.itmo.repository.FriendshipRepository;
import ru.itmo.service.FriendshipService;

import javax.transaction.Transactional;
import java.util.List;

@Service
@RequiredArgsConstructor
@Transactional
@Slf4j
public class FriendshipServiceImpl implements FriendshipService {
    private final FriendshipRepository friendshipRepository;

    @Override
    public Friendship createFriendship(int idFirstCat, int idSecondCat) {
        return friendshipRepository.save(new Friendship(idFirstCat, idSecondCat));
    }

    @Override
    public Friendship saveFriendship(Friendship friendship) {
        return friendshipRepository.save(friendship);
    }

    @Override
    public Friendship getFriendshipById(int id) {
        return friendshipRepository.findById(id);
    }

    @Override
    public List<Friendship> getFriendships() {
        return friendshipRepository.findAll();
    }

    @Override
    public Friendship deleteFriendship(Friendship friendship) {
        friendshipRepository.delete(friendship);
        return friendship;
    }
}
```

Листинг 1.26: OwnerServiceImpl.java

```java
package ru.itmo.serviceImpl;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import ru.itmo.models.Cat;
import ru.itmo.models.Owner;
import ru.itmo.repository.CatRepository;
import ru.itmo.repository.OwnerRepository;
import ru.itmo.service.OwnerService;

import javax.transaction.Transactional;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;

@Service
@RequiredArgsConstructor
@Transactional
@Slf4j
public class OwnerServiceImpl implements OwnerService {
    private final OwnerRepository ownerRepository;
    private final CatRepository catRepository;

    @Override
    public Owner createOwner(String name, String birthday) {
        return ownerRepository.save(new Owner(name, LocalDate.parse(
    birthday,
                DateTimeFormatter.ofPattern("yyyy.MM.dd"))));
    }

    @Override
    public Owner saveOwner(Owner owner) {
        return ownerRepository.save(owner);
    }

    @Override
    public void addCatToOwner(int catId, int ownerId) {
        Owner owner = ownerRepository.findById(ownerId);
        Cat cat = catRepository.findById(catId);
        cat.setOwner(owner);
        owner.getCats().add(cat);
    }

    @Override
    public Owner getOwnerById(int ownerId) {
        return ownerRepository.findById(ownerId);
    }

```

```
49      @Override
50      public List<Owner> getOwners() {
51          return ownerRepository.findAll();
52      }
53
54      @Override
55      public Owner deleteOwner(Owner owner) {
56          ownerRepository.delete(owner);
57          return owner;
58      }
59  }
```

Листинг 1.27: RoleServiceImpl.java

```
1  package ru.itmo.serviceImpl;
2
3  import lombok.AllArgsConstructor;
4  import lombok.RequiredArgsConstructor;
5  import lombok.extern.slf4j.Slf4j;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.stereotype.Service;
8  import ru.itmo.models.Role;
9  import ru.itmo.repository.OwnerRepository;
10 import ru.itmo.repository.RoleRepository;
11 import ru.itmo.service.RoleService;
12
13 import javax.transaction.Transactional;
14
15 @Service
16 @Transactional
17 @AllArgsConstructor
18 @Slf4j
19 public class RoleServiceImpl implements RoleService {
20     private final RoleRepository roleRepository;
21
22     @Override
23     public Role getRoleById(int roleId) {
24         return roleRepository.findById(roleId);
25     }
26 }
```

Листинг 1.28: UserServiceImpl.java

```
package ru.itmo.serviceImpl;

import lombok.AllArgsConstructor;
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.
    UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.
    BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import ru.itmo.models.Owner;
import ru.itmo.models.Role;
import ru.itmo.models.CustomUser;
import ru.itmo.repository.UserRepository;
import ru.itmo.service.UserService;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Collections;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

@AllArgsConstructor
@Service
public class UserServiceImpl implements UserService {
    @PersistenceContext
    private final UserRepository userRepository;
    private final BCryptPasswordEncoder bCryptPasswordEncoder = new
    BCryptPasswordEncoder();

    @Override
    public CustomUser createUser(String username, String password,
    Role role, Owner owner) throws Exception {
        List<String> users = userRepository.findAll().stream()
                .map(CustomUser::getUsername).collect(Collectors.
    toList());
        if (!users.contains(username)) {
            return userRepository.save(new CustomUser(username,
    bCryptPasswordEncoder.encode(password), role, owner));
        }
        throw new Exception("User with the same name already exists.")
    ;
    }

    @Override
```

```
43      public UserDetails loadUserByUsername(String username) throws
    UsernameNotFoundException {
44          CustomUser user = userRepository.findByUsername(username);
45          if (user == null) {
46              throw new UsernameNotFoundException("Unknown user: " +
    username);
47          }
48          return User.builder()
49                  .username(user.getUsername())
50                  .password(user.getPassword())
51                  .roles(user.getRole().getName())
52                  .build();
53      }
54
55      public CustomUser getUserById(int id) {
56          return userRepository.findById(id);
57      }
58
59      @Override
60      public CustomUser getUserByUsername(String username) {
61          return userRepository.findByUsername(username);
62      }
63
64      public CustomUser saveUser(CustomUser user) {
65          CustomUser userFromDB = userRepository.findByUsername(user.
    getUsername());
66
67          if (userFromDB != null) {
68              return null;
69          }
70
71          user.setRole(user.getRole());
72          user.setPassword(bCryptPasswordEncoder.encode(user.getPassword
    ()));
73          userRepository.save(user);
74          return user;
75      }
76  }
```

Листинг 1.29: HibernateSessionFactoryUtil.java

```
1  package ru.itmo.tools;
2
3  import org.hibernate.SessionFactory;
4  import ru.itmo.models.Cat;
5  import ru.itmo.models.Friendship;
6  import ru.itmo.models.Owner;
7  import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
8  import org.hibernate.cfg.Configuration;
9
10
11 public class HibernateSessionFactoryUtil {
12     private static SessionFactory sessionFactory;
13
14     private HibernateSessionFactoryUtil() {}
15
16     public static SessionFactory getSessionFactory() {
17         if (sessionFactory == null) {
18             try {
19                 Configuration configuration = new Configuration().
    configure();
20                 configuration.addAnnotatedClass(Owner.class);
21                 configuration.addAnnotatedClass(Cat.class);
22                 configuration.addAnnotatedClass(Friendship.class);
23                 StandardServiceRegistryBuilder builder = new
    StandardServiceRegistryBuilder().
24                         applySettings(configuration.getProperties());
25                 sessionFactory = configuration.buildSessionFactory(
    builder.build());
26
27             } catch (Exception e) {
28                 System.out.println("Exception!" + e);
29             }
30         }
31         return sessionFactory;
32     }
33 }
```

Листинг 1.30: CatWrapper.java

```
package ru.itmo.wrapper;

import lombok.AllArgsConstructor;
import lombok.Data;

import java.time.LocalDate;
import java.util.List;

@Data
@AllArgsConstructor
public class CatWrapper {
    private int id;
    private String name;
    private LocalDate birthday;
    private String breed;
    private int colorId;
    private int ownerId;
    private List<Integer> friendsId;
}
```

Листинг 1.31: FriendshipWrapper.java

```
1  package ru.itmo.wrapper;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5
6  @Data
7  @AllArgsConstructor
8  public class FriendshipWrapper {
9      private int id;
10     private int idFirstCat;
11     private int idSecondCat;
12 }
```

Листинг 1.32: OwnerWrapper.java

```
1  package ru.itmo.wrapper;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5
6  import java.time.LocalDate;
7  import java.util.List;
8
9  @Data
10 @AllArgsConstructor
11 public class OwnerWrapper {
12     private int id;
13     private String name;
14     private LocalDate birthday;
15     private List<Integer> cats;
16 }
```

Листинг 1.33: UserWrapper.java

```
1  package ru.itmo.wrapper;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import ru.itmo.models.Owner;
6  import ru.itmo.models.Role;
7
8  import java.util.Set;
9
10 @Data
11 @AllArgsConstructor
12 public class UserWrapper {
13     private int id;
14     private String username;
15     private String password;
16     private int roleId;
17     private int ownerId;
18 }
```