

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №3

Выполнил студент:

Моторина Евгения Викторовна
группа: М32051

Проверил:

Бутенко Олег Романович

Санкт-Петербург,
2022 г.

1.1. Текст задания

3 лабораторная

К созданному в прошлой лабораторной сервису добавляется Spring.

Сервис должен предоставлять http интерфейс (REST API) для получения информации о конкретных котиках и владельцах и для получения фильтрованной информации (например, получить всех рыжих котиков)

Внимание: недопустимо отдавать через HTTP интерфейс сущности JPA. Рекомендуется создать отдельные оберточные классы.

Сервисы и dao должны превратиться в Spring Bean'ы с использованием Dependency Injection (Autowired). Dao при этом наследуют JpaRepository и имеет шаблонные Spring Data JPA методы: <https://www.baeldung.com/spring-data-repositoriesrepositories>

При сдаче лабораторной нужно будет показать работоспособность endpoint'ов через http запросы (рекомендуется Postman).

В рамках лабораторной к проекту должен быть добавлен CI/CD, запускающий тесты проекта kotiki-java.

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
6 @SpringBootApplication
7 public class KotikiApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(KotikiApplication.class, args);
11     }
12
13
14
15 }
```

Листинг 1.2: CatController.java

```

1 package ru.itmo.api;
2
3 import lombok.RequiredArgsConstructor;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.*;
7 import ru.itmo.models.Cat;
8 import ru.itmo.models.Owner;
9 import ru.itmo.serviceml.CatServiceImpl;
10 import ru.itmo.serviceml.OwnerServiceImpl;
11 import ru.itmo.wrapper.CatWrapper;
12
13 import java.util.List;
14 import java.util.stream.Collectors;
15
16 @RestController
17 @RequestMapping("/api/cat")
18 @RequiredArgsConstructor
19 public class CatController {
20     private final CatServiceImpl catService;
21     private final OwnerServiceImpl ownerService;
22
23     @PostMapping("/create")
24     public ResponseEntity<CatWrapper> createCat(@RequestParam String
25 name, @RequestParam String birthday,
26                                     @RequestParam String
27 breed, @RequestParam int colorId,
28                                     @RequestParam int
29 ownerId) {
30         Owner owner = ownerService.getOwnerById(ownerId);
31         return ResponseEntity.ok().body(catService.createCat(name,
32 birthday, breed, colorId, owner).getWrapper());
33     }
34
35     @GetMapping("/getById")
36     public ResponseEntity<CatWrapper> getCat(@RequestParam int catId)
37     {
38         return ResponseEntity.ok().body(catService.getCatById(catId).
39 getWrapper());
40     }
41
42     @GetMapping("/all")
43     public ResponseEntity<List<CatWrapper>> getCats() {
44         return new ResponseEntity<>(
45             catService.getCats().stream().map((Cat::getWrapper)).
46             collect(Collectors.toList()), HttpStatus.
47 ACCEPTED);
48     }
49 }

```

```
43     @DeleteMapping("/delete")
44     public ResponseEntity<CatWrapper> deleteCat(@RequestParam int
catId) {
45         Cat cat = catService.getCatById(catId);
46         return ResponseEntity.ok().body(catService.deleteCat(cat).
getWrapper());
47     }
48 }
```

Листинг 1.3: FriendshipController.java

```
1 package ru.itmo.api;
2
3 import lombok.RequiredArgsConstructor;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.*;
7 import ru.itmo.models.Friendship;
8 import ru.itmo.serviceml.FriendshipServiceImpl;
9 import ru.itmo.wrapper.FriendshipWrapper;
10
11 import java.util.List;
12 import java.util.stream.Collectors;
13
14 @RestController
15 @RequestMapping("/api/friendship")
16 @RequiredArgsConstructor
17 public class FriendshipController {
18     private final FriendshipServiceImpl friendshipService;
19
20     @PostMapping("/create")
21     public ResponseEntity<FriendshipWrapper> createFriendship(
22         @RequestParam int idFirstCat, @RequestParam int idSecondCat) {
23         return ResponseEntity.ok().body(friendshipService.
24             createFriendship(idFirstCat, idSecondCat).getWrapper());
25     }
26
27     @GetMapping("/getById")
28     public ResponseEntity<FriendshipWrapper> getFriendship(
29         @RequestParam int friendshipId) {
30         return ResponseEntity.ok().body(friendshipService.
31             getFriendshipById(friendshipId).getWrapper());
32     }
33
34     @GetMapping("/all")
35     public ResponseEntity<List<FriendshipWrapper>> getFriendships() {
36         return new ResponseEntity<>(
37             friendshipService.getFriendships().stream().map((
38                 Friendship::getWrapper)).
39             collect(Collectors.toList()), HttpStatus.
40             ACCEPTED);
41     }
42
43     @DeleteMapping("/delete")
44     public ResponseEntity<FriendshipWrapper> deleteFriendship(
45         @RequestParam int friendshipId) {
46         Friendship friendship = friendshipService.getFriendshipById(
47             friendshipId);
48         return ResponseEntity.ok().body(friendshipService.
49             deleteFriendship(friendship).getWrapper());
50     }
```

41

}

42

}

Листинг 1.4: OwnerController.java

```
1 package ru.itmo.api;
2
3 import lombok.RequiredArgsConstructor;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.*;
7 import ru.itmo.models.Owner;
8 import ru.itmo.serviceml.CatServiceImpl;
9 import ru.itmo.serviceml.OwnerServiceImpl;
10 import ru.itmo.wrapper.OwnerWrapper;
11
12 import java.util.List;
13 import java.util.stream.Collectors;
14
15 @RestController
16 @RequestMapping("/api/owner")
17 @RequiredArgsConstructor
18 public class OwnerController {
19     private final OwnerServiceImpl ownerService;
20     private final CatServiceImpl catService;
21
22     @PostMapping("/create")
23     public ResponseEntity<OwnerWrapper> createOwner(@RequestParam
24 String name, @RequestParam String birthday) {
25         return ResponseEntity.ok().body(ownerService.createOwner(name,
26 birthday).getWrapper());
27     }
28
29     @PostMapping("/addCatToOwner")
30     public ResponseEntity<?> addCatToOwner(@RequestParam int catId,
31 @RequestParam int ownerId) {
32         ownerService.addCatToOwner(catId, ownerId);
33         return ResponseEntity.ok().build();
34     }
35
36     @GetMapping("/getById")
37     public ResponseEntity<OwnerWrapper> getOwner(@RequestParam int
38 ownerId) {
39         return ResponseEntity.ok().body(ownerService.getOwnerById(
40 ownerId).getWrapper());
41     }
42
43     @GetMapping("/all")
44     public ResponseEntity<List<OwnerWrapper>> getOwners() {
45         return new ResponseEntity<>(
46             ownerService.getOwners().stream().map((Owner::
47 getWrapper)).
48             collect(Collectors.toList()), HttpStatus.
49 ACCEPTED);
50 }
```



```
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 1.5: 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 @Data @NoArgsConstructor @AllArgsConstructor
14 @Table(name = "cats")
15 public class Cat {
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private int id;
19
20     @Column(name = "name")
21     private String name;
22
23     @Column(name = "birthday")
24     private LocalDate birthday;
25
26     @Column(name = "breed")
27     private String breed;
28
29     @Column(name = "color_id")
30     private int colorId;
31
32     @ManyToOne(fetch = FetchType.LAZY)
33     @JoinColumn(name = "owner_id")
34     private Owner owner;
35
36     @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
37     @JoinTable(name = "friends", joinColumns = @JoinColumn(name = "
id_first_cat"), inverseJoinColumns = @JoinColumn(name = "
id_second_cat"))
38     private List<Cat> friends;
39
40     public Cat(String name, LocalDate birthday, String breed, int
colorId, Owner owner) {
41         this.name = name;
42         this.birthday = birthday;
43         this.breed = breed;
44         this.colorId = colorId;
45         this.owner = owner;
46         this.friends = new ArrayList<>();
```

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

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

Листинг 1.6: Color.java

```
1 package ru.itmo.models;
2
3 public enum Color {
4     BLACK,
5     WHITE,
6     GRAY,
7     CALICO;
8
9     public Color getColorById(int id) throws Exception {
10         for (Color color: Color.values()) {
11             if (color.ordinal() == id) {
12                 return color;
13             }
14         }
15         throw new Exception("No color with this id");
16     }
17 }
```

Листинг 1.7: Friendship.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.FriendshipWrapper;
8
9 import javax.persistence.*;
10 import java.time.LocalDate;
11 import java.util.List;
12
13 @Entity @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Table(name = "friends")
17 public class Friendship {
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private int id;
21
22     @Column(name = "id_first_cat")
23     private int idFirstCat;
24
25     @Column(name = "id_second_cat")
26     private int idSecondCat;
27
28     public Friendship(int idFirstCat, int idSecondCat) {
29         this.idFirstCat = idFirstCat;
30         this.idSecondCat = idSecondCat;
31     }
32     // @ManyToMany(mappedBy = "friends")
33     // private List<Cat> cats;
34
35     public int getId() {
36         return id;
37     }
38
39     public void setId(int id) {
40         this.id = id;
41     }
42
43     public FriendshipWrapper getWrapper() {
44         return new FriendshipWrapper(id, idFirstCat, idSecondCat);
45     }
46
47 }
```

Листинг 1.8: 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 @Data
17 @NoArgsConstructor
18 @AllArgsConstructor
19 @Table (name = "owners")
20 public class Owner {
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     private int id;
24
25     @Column
26     private String name;
27
28     @Column
29     private LocalDate birthday;
30
31     @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL,
32 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 }
```

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


Листинг 1.9: 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 1.10: 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  
9     , Integer> {  
10     Friendship save(Friendship friendship);  
11     Friendship findById(int id);  
12     List<Friendship> findAll();  
13     void delete(Friendship friendship);  
14 }
```

Листинг 1.11: 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     {  
10     Owner save(Owner owner);  
11     Owner findById(int id);  
12     List<Owner> findAll();  
13     void delete(Owner owner);  
14 }
```

Листинг 1.12: 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  
10    colorId, Owner owner);  
11    Cat saveCat(Cat cat);  
12    Cat getCatById(int id);  
13    List<Cat> getCats();  
14    Cat deleteCat(Cat cat);  
15 }
```

Листинг 1.13: 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 1.14: 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 1.15: CatServiceImpl.java

```
1 package ru.itmo.servicelimpl;
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,
25         int colorId, Owner owner) {
26         return catRepository.save(new Cat(name, LocalDate.parse(
27             birthday,
28             DateTimeFormatter.ofPattern("yyyy.MM.dd")), breed,
29             colorId, owner));
30     }
31
32     @Override
33     public Cat saveCat(Cat cat) {
34         return catRepository.save(cat);
35     }
36
37     @Override
38     public Cat getCatById(int id) {
39         return catRepository.findById(id);
40     }
41
42     @Override
43     public List<Cat> getCats() {
44         return catRepository.findAll();
45     }
46
47     @Override
48     public Cat deleteCat(Cat cat) {
49         catRepository.delete(cat);
50     }
51 }
```

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


Листинг 1.16: FriendshipServiceImpl.java

```
1 package ru.itmo.servicelmpl;  
2  
3 import lombok.RequiredArgsConstructor;  
4 import lombok.extern.slf4j.Slf4j;  
5 import org.springframework.stereotype.Service;  
6 import ru.itmo.models.Friendship;  
7 import ru.itmo.repository.FriendshipRepository;  
8 import ru.itmo.service.FriendshipService;  
9  
10 import javax.transaction.Transactional;  
11 import java.util.List;  
12  
13 @Service  
14 @RequiredArgsConstructor  
15 @Transactional  
16 @Slf4j  
17 public class FriendshipServiceImpl implements FriendshipService {  
18     private final FriendshipRepository friendshipRepository;  
19  
20     @Override  
21     public Friendship createFriendship(int idFirstCat, int idSecondCat  
22 ) {  
23         return friendshipRepository.save(new Friendship(idFirstCat,  
24 idSecondCat));  
25     }  
26  
27     @Override  
28     public Friendship saveFriendship(Friendship friendship) {  
29         return friendshipRepository.save(friendship);  
30     }  
31  
32     @Override  
33     public Friendship getFriendshipById(int id) {  
34         return friendshipRepository.findById(id);  
35     }  
36  
37     @Override  
38     public List<Friendship> getFriendships() {  
39         return friendshipRepository.findAll();  
40     }  
41  
42     @Override  
43     public Friendship deleteFriendship(Friendship friendship) {  
44         friendshipRepository.delete(friendship);  
45         return friendship;  
46     }  
47 }
```

Листинг 1.17: OwnerServiceImpl.java

```
1 package ru.itmo.servicelmpl;
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.repository.OwnerRepository;
10 import ru.itmo.service.OwnerService;
11
12 import javax.transaction.Transactional;
13 import java.time.LocalDate;
14 import java.time.format.DateTimeFormatter;
15 import java.util.List;
16
17 @Service
18 @RequiredArgsConstructor
19 @Transactional
20 @Slf4j
21 public class OwnerServiceImpl implements OwnerService {
22     private final OwnerRepository ownerRepository;
23     private final CatRepository catRepository;
24
25     @Override
26     public Owner createOwner(String name, String birthday) {
27         return ownerRepository.save(new Owner(name, LocalDate.parse(
28             birthday,
29             DateTimeFormatter.ofPattern("yyyy.MM.dd"))));
30     }
31
32     @Override
33     public Owner saveOwner(Owner owner) {
34         return ownerRepository.save(owner);
35     }
36
37     @Override
38     public void addCatToOwner(int catId, int ownerId) {
39         Owner owner = ownerRepository.findById(ownerId);
40         Cat cat = catRepository.findById(catId);
41         cat.setOwner(owner);
42         owner.getCats().add(cat);
43     }
44
45     @Override
46     public Owner getOwnerById(int ownerId) {
47         return ownerRepository.findById(ownerId);
48     }
49 }
```

```
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 1.18: CatWrapper.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 CatWrapper {  
12     private int id;  
13     private String name;  
14     private LocalDate birthday;  
15     private String breed;  
16     private int colorId;  
17     private int ownerId;  
18     private List<Integer> friendsId;  
19 }
```

Листинг 1.19: 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 1.20: 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 1.21: application.properties

```
1 spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
2 spring.datasource.username=postgres
3 spring.datasource.password=qwerty
4 spring.jpa.show-sql=true
5 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.
   PostgreSQL95Dialect
6 spring.jpa.properties.hibernate.format_sql=true
```

Листинг 1.22: KotikiSpringTest.java

```
1 import org.junit.Assert;
2 import org.junit.jupiter.api.Test;
3 import org.junit.runner.RunWith;
4 import org.mockito.Mockito;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.boot.test.mock.mockito.MockBean;
8 import org.springframework.test.context.junit4.SpringRunner;
9 import ru.itmo.KotikiApplication;
10 import ru.itmo.models.Cat;
11 import ru.itmo.models.Friendship;
12 import ru.itmo.models.Owner;
13 import ru.itmo.repository.CatRepository;
14 import ru.itmo.repository.FriendshipRepository;
15 import ru.itmo.repository.OwnerRepository;
16 import ru.itmo.serviceml.CatServiceImpl;
17 import ru.itmo.serviceml.FriendshipServiceImpl;
18 import ru.itmo.serviceml.OwnerServiceImpl;
19
20 import java.time.LocalDate;
21
22 @RunWith(SpringRunner.class)
23 @SpringBootTest(classes = KotikiApplication.class)
24 public class KotikiSpringTest {
25
26     @Autowired
27     private OwnerServiceImpl ownerService;
28
29     @MockBean
30     private OwnerRepository ownerRepository;
31
32     @Autowired
33     private CatServiceImpl catService;
34
35     @MockBean
36     private CatRepository catRepository;
37
38     @Autowired
39     private FriendshipServiceImpl friendshipService;
40
41     @MockBean
42     private FriendshipRepository friendshipRepository;
43
44     @Test
45     public void testOwnerService() {
46         Owner owner = new Owner("Klara", LocalDate.of(1999,9,1));
47         ownerService.saveOwner(owner);
48         Mockito.verify(ownerRepository, Mockito.times(1)).save(owner);
49         Assert.assertEquals("Klara", owner.getName());
50     }
51 }
```



```
50 Mockito.doReturn(owner).when(ownerRepository).findById(0);
51 Assert.assertEquals(owner, ownerService.getOwnerById(0));
52 }
53
54 @Test
55 public void testCatService() {
56     Owner owner = new Owner("Zina", LocalDate.of(2007,3,28));
57     Cat cat = new Cat("Sandra", LocalDate.of(2022, 4, 15), "
Bengali",
58         3, owner);
59     catService.saveCat(cat);
60     Mockito.verify(catRepository, Mockito.times(1)).save(cat);
61     Assert.assertEquals(3, cat.getColorId());
62     Mockito.doReturn(cat).when(catRepository).findById(0);
63     Assert.assertEquals(cat, catService.getCatById(0));
64 }
65
66 @Test
67 public void testFriendshipService() {
68     Owner owner = new Owner("Pavel", LocalDate.of(2000,1,1));
69     Cat firstCat = new Cat("Leo", LocalDate.of(2019, 6, 5), "
Burmese",
70         1, owner);
71     Cat secondCat = new Cat("Kate", LocalDate.of(2019, 6, 6), "
Burmese",
72         1, owner);
73     Friendship friendship = new Friendship(firstCat.getId(),
secondCat.getId());
74     friendshipService.saveFriendship(friendship);
75     Mockito.verify(friendshipRepository, Mockito.times(1)).save(
friendship);
76 }
77 }
```