Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабороторная работа №3

Выполнил студент:

Моторина Евгения Викторовна
группа: M32051

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-repositoriesrepos
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

```java
package ru.itmo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class KotikiApplication {

    public static void main(String[] args) {
        SpringApplication.run(KotikiApplication.class, args);
    }



}
```

Листинг 1.2: CatController.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.Cat;
import ru.itmo.models.Owner;
import ru.itmo.serviceImpl.CatServiceImpl;
import ru.itmo.serviceImpl.OwnerServiceImpl;
import ru.itmo.wrapper.CatWrapper;

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

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

    @PostMapping("/create")
    public ResponseEntity<CatWrapper> createCat(@RequestParam String name, @RequestParam String birthday,
                                                @RequestParam String breed, @RequestParam int colorId,
                                                @RequestParam int ownerId) {
        Owner owner = ownerService.getOwnerById(ownerId);
        return ResponseEntity.ok().body(catService.createCat(name, birthday, breed, colorId, owner).getWrapper());
    }

    @GetMapping("/getById")
    public ResponseEntity<CatWrapper> getCat(@RequestParam int catId) {
        return ResponseEntity.ok().body(catService.getCatById(catId).getWrapper());
    }

    @GetMapping("/all")
    public ResponseEntity<List<CatWrapper>> getCats() {
        return new ResponseEntity<>(
                catService.getCats().stream().map((Cat::getWrapper)).
                        collect(Collectors.toList()), HttpStatus.ACCEPTED);
    }

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

Листинг 1.3: 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("/api/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());
        }
}
```

Листинг 1.4: OwnerController.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("/api/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);
```

```
    }

    @DeleteMapping("/delete")
    public ResponseEntity<OwnerWrapper> deleteOwner(@RequestParam int
  ownerId) {
        Owner owner = ownerService.getOwnerById(ownerId);
        return ResponseEntity.ok().body(ownerService.deleteOwner(owner
  ).getWrapper());
    }

}
```

Листинг 1.5: Cat.java

```java
package ru.itmo.models;

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

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

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

    @Column(name = "name")
    private String name;

    @Column(name = "birthday")
    private LocalDate birthday;

    @Column(name = "breed")
    private String breed;

    @Column(name = "color_id")
    private int colorId;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "owner_id")
    private Owner owner;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(name = "friends", joinColumns = @JoinColumn(name = "
  id_first_cat"), inverseJoinColumns = @JoinColumn(name = "
  id_second_cat"))
    private List<Cat> friends;

    public Cat(String name, LocalDate birthday, String breed, int
  colorId, Owner owner) {
        this.name = name;
        this.birthday = birthday;
        this.breed = breed;
        this.colorId = colorId;
        this.owner = owner;
        this.friends = new ArrayList<>();
    }

    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 String getBreed() {
        return breed;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    public int getColorId() {
        return colorId;
    }

    public void setColorId(int colorId) {
        this.colorId = colorId;
    }

    public Owner getOwner() {
        return owner;
    }

    public void setOwner(Owner owner) {
        this.owner = owner;
    }

    public int getOwnerId() {
        return owner.getId();
    }

    public List<Integer> getFriendsId() {
        List<Integer> friendsId = new ArrayList<>();
        for (Cat friend: friends) {
            friendsId.add(friend.id);
        }
        return friendsId;
    }

    public CatWrapper getWrapper() {
        return new CatWrapper(id, name, birthday, breed, colorId, this
   .getOwnerId(), this.getFriendsId());
    }
}
```

Листинг 1.6: Color.java

```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 1.7: Friendship.java

```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 1.8: Owner.java

```java
package ru.itmo.models;

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

import javax.persistence.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

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

    @Column
    private String name;

    @Column
    private LocalDate birthday;

    @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL,
   orphanRemoval = true)
    private List<Cat> cats;

    public void addCat(Cat cat) {
        cat.setOwner(this);
        cats.add(cat);
    }

    public Owner(String name, LocalDate birthday) {
        this.name = name;
        this.birthday = birthday;
        this.cats = new ArrayList<>() {
        };
    }

    public int getId() {
        return 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<Cat> getCats() {
        return cats;
    }

    public List<Integer> getCatsId() {
        List<Integer> catsId = new ArrayList<>();
        for (Cat cat: cats) {
            catsId.add(cat.getId());
        }
        return catsId;
    }

    public OwnerWrapper getWrapper() {
        return new OwnerWrapper(id, name, birthday, this.getCatsId());
    }
}
```

Листинг 1.9: CatRepository.java

```java
package ru.itmo.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import ru.itmo.models.Cat;

public interface CatRepository extends JpaRepository<Cat, Integer> {
    Cat save(Cat cat);
    Cat findById(int id);
    void delete(Cat cat);
}
```

Листинг 1.10: FriendshipRepository.java

```
package ru.itmo.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import ru.itmo.models.Friendship;

import java.util.List;

public interface FriendshipRepository extends JpaRepository<Friendship
    , Integer> {
    Friendship save(Friendship friendship);
    Friendship findById(int id);
    List<Friendship> findAll();
    void delete(Friendship friendship);
}
```

Листинг 1.11: OwnerRepository.java

```java
package ru.itmo.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import ru.itmo.models.Owner;

import java.util.List;

public interface OwnerRepository extends JpaRepository<Owner, Integer>
    {
    Owner save(Owner owner);
    Owner findById(int id);
    List<Owner> findAll();
    void delete(Owner owner);
}
```

Листинг 1.12: CatService.java

```java
package ru.itmo.service;

import ru.itmo.models.Cat;
import ru.itmo.models.Owner;

import java.util.List;

public interface CatService {
    Cat createCat(String name, String birthday, String breed, int
   colorId, Owner owner);
    Cat saveCat(Cat cat);
    Cat getCatById(int id);
    List<Cat> getCats();
    Cat deleteCat(Cat cat);
}
```

Листинг 1.13: FriendshipService.java

```java
package ru.itmo.service;

import ru.itmo.models.Friendship;

import java.util.List;

public interface FriendshipService {
    Friendship createFriendship(int idFirstCat, int idSecondCat);
    Friendship saveFriendship(Friendship friendship);
    Friendship getFriendshipById(int id);
    List<Friendship> getFriendships();
    Friendship deleteFriendship(Friendship friendship);
}
```

Листинг 1.14: OwnerService.java

```java
package ru.itmo.service;

import ru.itmo.models.Owner;

import java.util.List;

public interface OwnerService {
    Owner createOwner(String name, String birthday);
    Owner saveOwner(Owner owner);
    void addCatToOwner(int catId, int ownerId);
    Owner getOwnerById(int ownerId);
    List<Owner> getOwners();
    Owner deleteOwner(Owner owner);
}
```

Листинг 1.15: CatServiceImpl.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.service.CatService;

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

@Service
@RequiredArgsConstructor
@Transactional
@Slf4j
public class CatServiceImpl implements CatService {
    private final CatRepository catRepository;

    @Override
    public Cat createCat(String name, String birthday, String breed,
  int colorId, Owner owner) {
        return catRepository.save(new Cat(name, LocalDate.parse(
  birthday,
                DateTimeFormatter.ofPattern("yyyy.MM.dd")), breed,
  colorId, owner));
    }

    @Override
    public Cat saveCat(Cat cat) {
        return catRepository.save(cat);
    }

    @Override
    public Cat getCatById(int id) {
        return catRepository.findById(id);
    }

    @Override
    public List<Cat> getCats() {
        return catRepository.findAll();
    }

    @Override
    public Cat deleteCat(Cat cat) {
        catRepository.delete(cat);
```

```
            return cat;
    }
}
```

Листинг 1.16: 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 1.17: 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);
    }

    @Override
    public List<Owner> getOwners() {
        return ownerRepository.findAll();
    }

    @Override
    public Owner deleteOwner(Owner owner) {
        ownerRepository.delete(owner);
        return owner;
    }
}
```

Листинг 1.18: CatWrapper.java

```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 1.19: FriendshipWrapper.java

```java
package ru.itmo.wrapper;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class FriendshipWrapper {
    private int id;
    private int idFirstCat;
    private int idSecondCat;
}
```

Листинг 1.20: OwnerWrapper.java

```java
package ru.itmo.wrapper;

import lombok.AllArgsConstructor;
import lombok.Data;

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

@Data
@AllArgsConstructor
public class OwnerWrapper {
    private int id;
    private String name;
    private LocalDate birthday;
    private List<Integer> cats;
}
```

Листинг 1.21: application.properties

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

Листинг 1.22: KotikiSpringTest.java

```java
import org.junit.Assert;
import org.junit.jupiter.api.Test;
import org.junit.runner.RunWith;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.test.context.junit4.SpringRunner;
import ru.itmo.KotikiApplication;
import ru.itmo.models.Cat;
import ru.itmo.models.Friendship;
import ru.itmo.models.Owner;
import ru.itmo.repository.CatRepository;
import ru.itmo.repository.FriendshipRepository;
import ru.itmo.repository.OwnerRepository;
import ru.itmo.serviceImpl.CatServiceImpl;
import ru.itmo.serviceImpl.FriendshipServiceImpl;
import ru.itmo.serviceImpl.OwnerServiceImpl;

import java.time.LocalDate;

@RunWith(SpringRunner.class)
@SpringBootTest(classes = KotikiApplication.class)
public class KotikiSpringTest {

    @Autowired
    private OwnerServiceImpl ownerService;

    @MockBean
    private OwnerRepository ownerRepository;

    @Autowired
    private CatServiceImpl catService;

    @MockBean
    private CatRepository catRepository;

    @Autowired
    private FriendshipServiceImpl friendshipService;

    @MockBean
    private FriendshipRepository friendshipRepository;

    @Test
    public void testOwnerService() {
        Owner owner = new Owner("Klara", LocalDate.of(1999,9,1));
        ownerService.saveOwner(owner);
        Mockito.verify(ownerRepository, Mockito.times(1)).save(owner);
        Assert.assertEquals("Klara", owner.getName());
        Mockito.doReturn(owner).when(ownerRepository).findById(0);
        Assert.assertEquals(owner, ownerService.getOwnerById(0));
    }

    @Test
    public void testCatService() {
        Owner owner = new Owner("Zina", LocalDate.of(2007,3,28));
        Cat cat = new Cat("Sandra", LocalDate.of(2022, 4, 15), "Bengali",
                3, owner);
        catService.saveCat(cat);
        Mockito.verify(catRepository, Mockito.times(1)).save(cat);
        Assert.assertEquals(3, cat.getColorId());
        Mockito.doReturn(cat).when(catRepository).findById(0);
        Assert.assertEquals(cat, catService.getCatById(0));
    }

    @Test
    public void testFriendshipService() {
        Owner owner = new Owner("Pavel", LocalDate.of(2000,1,1));
        Cat firstCat = new Cat("Leo", LocalDate.of(2019, 6, 5), "Burmese",
                1, owner);
        Cat secondCat = new Cat("Kate", LocalDate.of(2019, 6, 6), "Burmese",
                1, owner);
        Friendship friendship = new Friendship(firstCat.getId(), secondCat.getId());
        friendshipService.saveFriendship(friendship);
        Mockito.verify(friendshipRepository, Mockito.times(1)).save(friendship);
    }
}
```