Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабороторная работа №2

Выполнил студент:

Моторина Евгения Викторовна
Группа: M32051

Проверил:

Бутенко Олег Романович

Санкт-Петербург,
2022 г.

1.1. Текст задания

2 лабораторная
Нужно написать сервис по учету котиков и их владельцев.
Существующая информация о котиках:
- Имя
- Дата рождения
- Порода
- Цвет (один из заранее заданных вариантов)
- Хозяин
- Список котиков, с которыми дружит этот котик (из представленных в базе)
Существующая информация о хозяевах:
- Имя
- Дата рождения
- Список котиков
Сервис должен реализовывать архитектуру controller-service-dao.
Вся информация хранится в БД PostgreSQL. Для связи с БД должен использоваться Hibernate.
Проект должен собираться с помощью Maven или Gradle (на выбор студента). Слой доступа к данным и сервисный слой должны являться двумя разными модулями Maven/Gradle. При этом проект должен полностью собираться одной командой.
При тестировании рекомендуется использовать Mockito, чтобы избежать подключения к реальным базам данных. Фреймворк для тестирования рекомендуется Junit 5.
В данной лабораторной нельзя использовать Spring или подобные ему фреймворки.

1.2. Решение

Листинг 1.1: Program.java

```
1  package ru.itmo;
2
3  import ru.itmo.models.Cat;
4  import ru.itmo.models.Color;
5  import ru.itmo.models.Owner;
6  import ru.itmo.service.CatService;
7  import ru.itmo.service.FriendshipService;
8  import ru.itmo.service.OwnerService;
9
10 import javax.transaction.HeuristicMixedException;
11 import javax.transaction.HeuristicRollbackException;
12 import javax.transaction.RollbackException;
13 import javax.transaction.SystemException;
14 import java.time.LocalDate;
15
16 public class Program {
17     public static void main(String[] args) throws
       HeuristicRollbackException, SystemException,
       HeuristicMixedException, RollbackException {
18         CatService catService = new CatService();
19         OwnerService ownerService = new OwnerService();
20         //Owner owner = ownerService.create("Tanya", LocalDate.of
       (2002, 5, 29));
21         //catService.create("Klaus", LocalDate.of(2015, 2, 5), Color.
       GRAY, "Cymric", owner);
22         FriendshipService friendshipService = new FriendshipService();
23         friendshipService.create(23, 24);
24     }
25
26 }
```

Листинг 1.2: AbstractDao.java

```
1  package ru.itmo.dao;
2
3  import org.hibernate.Session;
4  import org.hibernate.Transaction;
5  import ru.itmo.models.Owner;
6  import ru.itmo.tools.HibernateSessionFactoryUtil;
7
8  import javax.persistence.criteria.CriteriaBuilder;
9  import javax.persistence.criteria.CriteriaQuery;
10 import java.util.List;
11
12 public class AbstractDao {
13
14     private Object entity;
15
16     public AbstractDao(Object entity) {
17         this.entity = entity;
18     }
19
20     public Object findById(int id) {
21         Session session = HibernateSessionFactoryUtil.
    getSessionFactory().openSession();
22         Object object = session.get(entity.getClass(), id);
23         session.close();
24         return object;
25     }
26
27     public void save(Object object) {
28         Session session = HibernateSessionFactoryUtil.
    getSessionFactory().openSession();
29         Transaction tx = session.beginTransaction();
30         session.save(object);
31         tx.commit();
32         session.close();
33     }
34
35     public void update(Object object){
36         Session session = HibernateSessionFactoryUtil.
    getSessionFactory().openSession();
37         Transaction tx = session.beginTransaction();
38         session.update(object);
39         tx.commit();
40         session.close();
41     }
42
43     public void delete(Object object) {
44         Session session = HibernateSessionFactoryUtil.
    getSessionFactory().openSession();
45         Transaction tx = session.beginTransaction();
```

```
46          session.delete(object);
47          tx.commit();
48          session.close();
49      }
50
51      public List<Object> findAll() {
52          Session session = HibernateSessionFactoryUtil.
    getSessionFactory().openSession();
53          CriteriaBuilder builder = session.getCriteriaBuilder();
54          CriteriaQuery<Object> criteria = builder.createQuery((Class<
    Object>) entity.getClass());
55          criteria.from(entity.getClass());
56          List<Object> objects = session.createQuery(criteria).
    getResultList();
57          session.close();
58          return objects;
59      }
60  }
```

Листинг 1.3: Cat.java

```
1  package ru.itmo.models;
2
3  import javax.persistence.*;
4  import java.time.LocalDate;
5  import java.util.List;
6
7  @Entity
8  @Table(name = "cats")
9  public class Cat {
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private int id;
13
14     @Column(name = "name")
15     private String name;
16
17     @Column(name = "birthday")
18     private LocalDate birthday;
19
20     @Column(name = "breed")
21     private String breed;
22
23     @Column (name = "color_id")
24     private int colorId;
25
26     @ManyToOne(fetch = FetchType.LAZY)
27     @JoinColumn(name = "owner_id")
28     private Owner owner;
29
30     @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
31     @JoinTable(name = "friends",
32             joinColumns = @JoinColumn(name = "id_first_cat"),
33             inverseJoinColumns = @JoinColumn(name = "id_second_cat"))
34     private List<Cat> friends;
35
36     public Cat() {
37     }
38
39     public Cat(String name, LocalDate birthday, Color color, String
       breed) {
40         this.name = name;
41         this.birthday = birthday;
42         this.colorId = color.ordinal();
43         this.breed = breed;
44     }
45
46     public int getId() {
47         return id;
48     }
```

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

```
99      }*/
100 }
```

Листинг 1.4: 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 1.5: Friendship.java

```
1  package ru.itmo.models;
2
3  import javax.persistence.*;
4  import java.time.LocalDate;
5  import java.util.List;
6
7  @Entity
8  @Table(name = "friends")
9  public class Friendship {
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private int id;
13
14     @Column(name = "id_first_cat")
15     private int idFirstCat;
16
17     @Column(name = "id_second_cat")
18     private int idSecondCat;
19
20     //@ManyToMany(mappedBy = "friends")
21     //private List<Cat> cats;
22
23     public Friendship() {
24     }
25
26     public Friendship(int idFirstCat, int idSecondCat) {
27         this.idFirstCat = idFirstCat;
28         this.idSecondCat = idSecondCat;
29     }
30
31     public int getId() {
32         return id;
33     }
34
35     public void setId(int id) {
36         this.id = id;
37     }
38
39 }
```

Листинг 1.6: Owner.java

```java
package ru.itmo.models;

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

@Entity
@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 Owner() {
    }

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

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

   // public void removeAuto(Cat cat) {
//         cats.remove(cat);
//    }

    public int getId() {
        return id;
    }

```

```
49      public String getName() {
50          return name;
51      }
52
53      public void setName(String name) {
54          this.name = name;
55      }
56
57      public LocalDate getBirthday() {
58          return birthday;
59      }
60
61      public void setBirthday(LocalDate birthday) {
62          this.birthday = birthday;
63      }
64
65  //      public List<Cat> getCats() {
66  //          return cats;
67  //      }
68  //
69  //      public void setAutos(List<Cat> autos) {
70  //          this.cats = cats;
71  //      }
72
73      @Override
74      public String toString() {
75          return "models.User{" +
76                  "id=" + id +
77                  ", name='" + name + '\'' +
78                  ", birthday=" + birthday +
79                  '}';
80      }
81  }
```

Листинг 1.7: CatService.java

```
1  package ru.itmo.service;
2
3  import ru.itmo.dao.AbstractDao;
4  import ru.itmo.models.Cat;
5  import ru.itmo.models.Color;
6  import ru.itmo.models.Owner;
7
8  import java.time.LocalDate;
9  import java.util.ArrayList;
10 import java.util.List;
11 import java.util.Objects;
12
13 public class CatService {
14
15     private AbstractDao abstractDao = new AbstractDao(new Cat());
16
17     public CatService() {
18     }
19
20     public Cat create(String name, LocalDate birthday, Color color,
       String breed, Owner owner) {
21         Cat newCat = new Cat(name, birthday, color, breed);
22         owner.addCat(newCat);
23         abstractDao.save(newCat);
24         return newCat;
25     }
26
27     public Cat update(Cat cat) {
28         abstractDao.update(cat);
29         return cat;
30     }
31
32     public Cat delete(Cat cat) {
33         abstractDao.delete(cat);
34         return cat;
35     }
36
37     public Cat findById(int id) {
38         return (Cat) abstractDao.findById(id);
39     }
40
41     public List<Cat> findAll() {
42         List<Object> objects = abstractDao.findAll();
43         List<Cat> cats = new ArrayList<Cat>();
44         for (Object item : objects) {
45             cats.add((Cat) item);
46         }
47         return cats;
48     }
```

```
49 }
```

Листинг 1.8: FriendshipService.java

```java
package ru.itmo.service;

import ru.itmo.dao.AbstractDao;
import ru.itmo.models.Cat;
import ru.itmo.models.Friendship;
import ru.itmo.models.Owner;

import java.util.ArrayList;
import java.util.List;

public class FriendshipService {
    private AbstractDao abstractDao = new AbstractDao(new Friendship());

    public FriendshipService() {
    }

    public Friendship create(int idFirstCat, int idSecondCat) {
        Friendship newFriendship = new Friendship(idFirstCat, idSecondCat);
        abstractDao.save(newFriendship);
        return newFriendship;
    }

    public Friendship update(Friendship friendship) {
        abstractDao.update(friendship);
        return friendship;
    }

    public Friendship delete(Friendship friendship) {
        abstractDao.delete(friendship);
        return friendship;
    }

    public Friendship findById(int id) {
        return (Friendship) abstractDao.findById(id);
    }

    public List<Friendship> findAll() {
        List<Object> objects = abstractDao.findAll();
        List<Friendship> friendships = new ArrayList<Friendship>();
        for (Object item : objects) {
            friendships.add((Friendship) item);
        }
        return friendships;
    }
}
```

Листинг 1.9: OwnerService.java

```java
package ru.itmo.service;

import ru.itmo.dao.AbstractDao;
import ru.itmo.models.Owner;

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

public class OwnerService {
    private AbstractDao abstractDao = new AbstractDao(new Owner());

    public OwnerService() {
    }

    public Owner create(String name, LocalDate date){
        Owner newOwner = new Owner(name, date);
        abstractDao.save(newOwner);
        return newOwner;
    }

    public Owner update(Owner owner) {
        abstractDao.update(owner);
        return owner;
    }

    public Owner delete(Owner owner) {
        abstractDao.delete(owner);
        return owner;
    }

    public Owner findById(int id) {
        return (Owner) abstractDao.findById(id);
    }

    public List<Owner> findAll() {
        List<Object> objects = abstractDao.findAll();
        List<Owner> owners = new ArrayList<Owner>();
        for (Object item : objects) {
            owners.add((Owner) item);
        }
        return owners;
    }
}
```

Листинг 1.10: 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 1.11: hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd"
    >
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">org.
    postgresql.Driver</property>
        <property name="hibernate.connection.url">jdbc:postgresql://
    localhost:5432/postgres</property>
        <property name="hibernate.connection.username">postgres</
    property>
        <property name="hibernate.connection.password">qwerty</
    property>
        <property name="show_sql">true</property>
        <property name="dialect">org.hibernate.dialect.
    PostgreSQL95Dialect</property>
    </session-factory>
</hibernate-configuration>
```

Листинг 1.12: build.gradle

```
plugins {
    id 'java'
}

group 'org.example'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'junit:junit:4.13.1'
    implementation group: 'org.postgresql', name: 'postgresql',
   version: '42.3.3'
    implementation 'org.hibernate:hibernate-core:5.6.5.Final'
    testImplementation group: 'org.mockito', name: 'mockito-core',
   version: '4.3.1'
}

test {
    useJUnit()
}
```

Листинг 1.13: KotikiTest.java

```
1  import org.junit.Assert;
2  import org.junit.Before;
3  import org.junit.Test;
4  import ru.itmo.models.Cat;
5  import ru.itmo.models.Color;
6  import ru.itmo.models.Owner;
7  import ru.itmo.service.CatService;
8  import ru.itmo.service.OwnerService;
9
10 import java.time.LocalDate;
11
12 public class KotikiTest {
13     private CatService catService;
14     private OwnerService ownerService;
15
16     @Before
17     public void setUp() {
18         catService = new CatService();
19         ownerService = new OwnerService();
20     }
21
22     @Test
23     public void createOwner() {
24         int size = ownerService.findAll().size();
25         ownerService.create("Fedor", LocalDate.of(2001, 3, 31));
26         Assert.assertEquals(size + 1, ownerService.findAll().size());
27     }
28
29     @Test
30     public void createOwner_addCat() {
31         Owner owner = ownerService.create("Kristina", LocalDate.of
       (2007, 12, 7));
32         Cat cat = catService.create("Stefan", LocalDate.of(2022,4,1),
       Color.CALICO,
33                 "Balinese", owner);
34         Assert.assertEquals(catService.findById(cat.getId()).getId(),
       cat.getId());
35      }
36
37      @Test
38      public void deleteCat() {
39         Owner owner = ownerService.create("Lev", LocalDate.of(20011,
       8, 21));
40         Cat cat = catService.create("Bella", LocalDate.of(2018,9,3),
       Color.BLACK,
41                 "Siamese", owner);
42         int size = ownerService.findAll().size();
43         catService.delete(cat);
44         Assert.assertNotEquals(size, catService.findAll().size());;
```

```
45         }
46 }
```

Листинг 1.14: KotikiTestMockito.java

```
1  import org.junit.Assert;
2  import org.junit.Test;
3  import org.mockito.InjectMocks;
4  import org.mockito.Mock;
5  import ru.itmo.dao.AbstractDao;
6  import ru.itmo.models.Cat;
7  import ru.itmo.models.Color;
8  import ru.itmo.service.CatService;
9
10 import java.time.LocalDate;
11
12 import static org.mockito.Mockito.*;
13
14 public class KotikiTestMockito {
15
16     @Mock
17     private AbstractDao catDao;
18     @InjectMocks
19     private CatService catService;
20
21     @Test
22     public void createCat() {
23         Cat mockCat = new Cat("Stefan", LocalDate.of(2022, 04, 01),
       Color.CALICO, "Balinese ");
24         when(catDao.findById(1)).thenReturn(mockCat);
25
26         Cat cat = catService.findById(1);
27         verify(catDao).findById(1);
28
29         Assert.assertEquals(mockCat, cat);
30
31     }
32
33 }
```