Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Лабороторная работа №5

Выполнил студент:

Моторина Евгения Викторовна
группа: M32051

Проверил:

Бутенко Олег Романович

Санкт-Петербург,
2022 г.

1.1. Текст задания

5 лабораторная
Бизнес прочитал статью о том, что микросервисы это круто и попросил нас разбить программу на микросервисы.
Из созданного приложения выделяются три микросервиса:
- Микросервис доступа к котикам
- Микросервис доступа к владельцам
- Микросервис с внешними интерфейсами.

Все они являются разными приложениями.
Все созданные ранее эндпоинты и авторизация переезжает на третий микросервис.
Общение между микросервисами происходит посредством RabbitMQ/Kafka (на выбор студента).
Сервисы доступа к котикам и доступа к владельцам могут либо быть подключены к одной БД, либо иметь разные БД. Во втором случае недопустимо делать один запрос на получение данных из двух БД, запроса должно быть два (по одному в каждую).
Внимание: недопустимо передавать через RabbitMQ/Kafka JPA сущности. Рекомендуется создать отдельные оберточные классы.
Рекомендуется выделить модуль с JPA сущностями в отдельный подключаемый модуль.

1.2. Решение

Листинг 1.1: KafkaConsumerConfig.java

```
1  package ru.itmo.kotikiservices.kafka;
2
3  import org.apache.kafka.clients.consumer.ConsumerConfig;
4  import org.apache.kafka.common.serialization.LongDeserializer;
5  import org.apache.kafka.common.serialization.StringDeserializer;
6  import org.springframework.beans.factory.annotation.Value;
7  import org.springframework.context.annotation.Bean;
8  import org.springframework.context.annotation.Configuration;
9  import org.springframework.kafka.config.
       ConcurrentKafkaListenerContainerFactory;
10 import org.springframework.kafka.config.KafkaListenerContainerFactory;
11 import org.springframework.kafka.core.ConsumerFactory;
12 import org.springframework.kafka.core.DefaultKafkaConsumerFactory;
13 import org.springframework.kafka.support.converter.
       BatchMessagingMessageConverter;
14 import org.springframework.kafka.support.converter.
       StringJsonMessageConverter;
15
16 import java.util.HashMap;
17 import java.util.Map;
18
19 @Configuration
20 public class KafkaConsumerConfig {
21     @Value("${kafka.server}")
22     private String kafkaServer;
23
24     @Value("${kafka.group.id}")
25     private String kafkaGroupId;
26
27     @Bean
28     public KafkaListenerContainerFactory<?> batchFactory() {
29         ConcurrentKafkaListenerContainerFactory<Long, String> factory
       =
30                 new ConcurrentKafkaListenerContainerFactory<>();
31         factory.setConsumerFactory(consumerFactory());
32         factory.setBatchListener(true);
33         factory.setMessageConverter(new BatchMessagingMessageConverter
       (converter()));
34         return factory;
35     }
36
37     @Bean
38     public KafkaListenerContainerFactory<?> singleFactory() {
39         ConcurrentKafkaListenerContainerFactory<Long, String> factory
       =
40                 new ConcurrentKafkaListenerContainerFactory<>();
41         factory.setConsumerFactory(consumerFactory());
42         factory.setBatchListener(false);
43         factory.setMessageConverter(new StringJsonMessageConverter());
```

```
44          return factory;
45      }
46
47      @Bean
48      public ConsumerFactory<? super Long, ? super String>
    consumerFactory() {
49          return new DefaultKafkaConsumerFactory<>(consumerConfigs());
50      }
51
52      @Bean
53      public KafkaListenerContainerFactory<?>
    kafkaListenerContainerFactory() {
54          return new ConcurrentKafkaListenerContainerFactory<>();
55      }
56
57      @Bean
58      public Map<String, Object> consumerConfigs() {
59          Map<String, Object> props = new HashMap<>();
60          props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaServer
    );
61          props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
    LongDeserializer.class);
62          props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
    StringDeserializer.class);
63          props.put(ConsumerConfig.GROUP_ID_CONFIG, kafkaGroupId);
64          props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, true);
65          return props;
66      }
67
68      @Bean
69      public StringJsonMessageConverter converter() {
70          return new StringJsonMessageConverter();
71      }
72  }
```

Листинг 1.2: Cat.java

```java
package ru.itmo.kotikiservices.models;


import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import ru.itmo.kotikiservices.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;
```

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

```
97      public void setOwner(Owner owner) {
98          this.owner = owner;
99      }
100
101     public int getOwnerId() {
102         return owner.getId();
103     }
104
105     public List<Integer> getFriendsId() {
106         List<Integer> friendsId = new ArrayList<>();
107         for (Cat friend: friends) {
108             friendsId.add(friend.id);
109         }
110         return friendsId;
111     }
112
113     public CatWrapper getWrapper() {
114         return new CatWrapper(id, name, birthday, breed, colorId, this
    .getOwnerId(), this.getFriendsId());
115     }
116 }
```

Листинг 1.3: Color.java

```
package ru.itmo.kotikiservices.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 1.4: Friendship.java

```
package ru.itmo.kotikiservices.models;

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

import javax.persistence.*;

@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;
    }

    public int getId() {
        return id;
    }

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

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

}
```

Листинг 1.5: Owner.java

```java
package ru.itmo.kotikiservices.models;

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

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

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

    @Column
    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() {
```

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

Листинг 1.6: CatRepository.java

```
1  package ru.itmo.kotikiservices.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.kotikiservices.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 1.7: FriendshipRepository.java

```
1  package ru.itmo.kotikiservices.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.kotikiservices.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 deleteById(Friendship friendship);
13 }
```

Листинг 1.8: OwnerRepository.java

```
1  package ru.itmo.kotikiservices.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import ru.itmo.kotikiservices.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 1.9: CatService.java

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

Листинг 1.10: FriendshipService.java

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

Листинг 1.11: OwnerService.java

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

Листинг 1.12: CatServiceImpl.java

```
1  package ru.itmo.kotikiservices.serviceImpl;
2
3  import com.google.gson.Gson;
4  import com.google.gson.GsonBuilder;
5  import lombok.RequiredArgsConstructor;
6  import lombok.extern.slf4j.Slf4j;
7  import org.springframework.kafka.annotation.KafkaListener;
8  import org.springframework.stereotype.Service;
9  import ru.itmo.kotikiservices.models.Cat;
10 import ru.itmo.kotikiservices.models.Owner;
11 import ru.itmo.kotikiservices.repository.CatRepository;
12 import ru.itmo.kotikiservices.service.CatService;
13 import ru.itmo.kotikiservices.service.OwnerService;
14
15 import javax.transaction.Transactional;
16 import java.time.LocalDate;
17 import java.time.format.DateTimeFormatter;
18 import java.util.HashMap;
19 import java.util.List;
20 import java.util.Map;
21
22 @Service
23 @RequiredArgsConstructor
24 @Transactional
25 @Slf4j
26 public class CatServiceImpl implements CatService {
27     private final CatRepository catRepository;
28     private final OwnerService ownerService;
29
30     @Override
31     @KafkaListener(id = "createCat", topics = {"cat.create"},
   containerFactory = "singleFactory")
32     public Cat createCat(String message) {
33         var data = new GsonBuilder().create().fromJson(message,
   HashMap.class);
34         String name = data.get("name").toString();
35         String birthday = data.get("birthday").toString();
36         String breed = data.get("breed").toString();
37         Integer colorId = Integer.valueOf((String) data.get("colorId")
   );
38         Integer ownerId = Integer.valueOf((String) data.get("ownerId")
   );
39         Owner owner = ownerService.getOwnerById(ownerId);
40         return catRepository.save(new Cat(name, LocalDate.parse(
   birthday,
41                 DateTimeFormatter.ofPattern("yyyy.MM.dd")), breed,
   colorId, owner));
42     }
43
```

```
44      @Override
45      public Cat saveCat(Cat cat) {
46          return catRepository.save(cat);
47      }
48
49      @Override
50      public Cat getCatById(int id) {
51          return catRepository.findById(id);
52      }
53
54      @Override
55      public List<Cat> getCats() {
56          return catRepository.findAll();
57      }
58
59      @KafkaListener(id = "deleteCat", topics = {"cat.delete"},
    containerFactory = "singleFactory")
60      public void deleteCat(String message) {
61          var data = new GsonBuilder().create().fromJson(message,
    HashMap.class);
62          Integer catId = Integer.valueOf((String) data.get("catId"));
63          catRepository.deleteById(catId);
64      }
65  }
```

Листинг 1.13: FriendshipServiceImpl.java

```
1  package ru.itmo.kotikiservices.serviceImpl;
2
3  import com.google.gson.GsonBuilder;
4  import lombok.RequiredArgsConstructor;
5  import lombok.extern.slf4j.Slf4j;
6  import org.springframework.kafka.annotation.KafkaListener;
7  import org.springframework.stereotype.Service;
8  import ru.itmo.kotikiservices.models.Friendship;
9  import ru.itmo.kotikiservices.models.Owner;
10 import ru.itmo.kotikiservices.repository.FriendshipRepository;
11 import ru.itmo.kotikiservices.service.FriendshipService;
12
13 import javax.transaction.Transactional;
14 import java.util.HashMap;
15 import java.util.List;
16
17 @Service
18 @RequiredArgsConstructor
19 @Transactional
20 @Slf4j
21 public class FriendshipServiceImpl implements FriendshipService {
22     private final FriendshipRepository friendshipRepository;
23
24     @KafkaListener(id = "friendship", topics = {"friendship.create"},
    containerFactory = "singleFactory")
25     public Friendship createFriendship(String message) {
26         var data = new GsonBuilder().create().fromJson(message,
    HashMap.class);
27         Integer idFirstCat = Integer.valueOf((String) data.get("
    idFirstCat"));
28         Integer idSecondCat = Integer.valueOf((String) data.get("
    idSecondCat"));
29         return friendshipRepository.save(new Friendship(idFirstCat,
    idSecondCat));
30     }
31
32     @Override
33     public Friendship saveFriendship(Friendship friendship) {
34         return friendshipRepository.save(friendship);
35     }
36
37     @Override
38     public Friendship getFriendshipById(int id) {
39         return friendshipRepository.findById(id);
40     }
41
42     @Override
43     public List<Friendship> getFriendships() {
44         return friendshipRepository.findAll();
```

```
45      }
46
47      @KafkaListener(id = "deleteFriendship", topics = {"friendship.
    delete"}, containerFactory = "singleFactory")
48      public void deleteFriendship(String message) {
49          var data = new GsonBuilder().create().fromJson(message,
    HashMap.class);
50          Integer friendshipId = Integer.valueOf((String) data.get("
    friendshipId"));
51          friendshipRepository.deleteById(friendshipId);
52      }
53  }
```

Листинг 1.14: OwnerServiceImpl.java

```
1  package ru.itmo.kotikiservices.serviceImpl;
2
3  import com.google.gson.GsonBuilder;
4  import lombok.RequiredArgsConstructor;
5  import lombok.extern.slf4j.Slf4j;
6  import org.springframework.kafka.annotation.KafkaListener;
7  import org.springframework.stereotype.Service;
8  import ru.itmo.kotikiservices.models.Cat;
9  import ru.itmo.kotikiservices.models.Owner;
10 import ru.itmo.kotikiservices.repository.CatRepository;
11 import ru.itmo.kotikiservices.repository.OwnerRepository;
12 import ru.itmo.kotikiservices.service.OwnerService;
13
14 import javax.transaction.Transactional;
15 import java.time.LocalDate;
16 import java.time.format.DateTimeFormatter;
17 import java.util.HashMap;
18 import java.util.List;
19 import java.util.Optional;
20
21 @Service
22 @RequiredArgsConstructor
23 @Transactional
24 @Slf4j
25 public class OwnerServiceImpl implements OwnerService {
26     private final OwnerRepository ownerRepository;
27     private final CatRepository catRepository;
28
29     @KafkaListener(id = "createOwner", topics = {"owner.create"},
    containerFactory = "singleFactory")
30     public Owner createOwner(String message) {
31         var data = new GsonBuilder().create().fromJson(message,
    HashMap.class);
32         String name = data.get("name").toString();
33         String birthday = data.get("birthday").toString();
34         return ownerRepository.save(new Owner(name, LocalDate.parse(
    birthday,
35                 DateTimeFormatter.ofPattern("yyyy.MM.dd"))));
36     }
37
38     @Override
39     public Owner saveOwner(Owner owner) {
40         return ownerRepository.save(owner);
41     }
42
43     @Override
44     public void addCatToOwner(int catId, int ownerId) {
45         Owner owner = ownerRepository.findById(ownerId);
46         Cat cat = catRepository.findById(catId);
```

```
47          cat.setOwner(owner);
48          owner.getCats().add(cat);
49      }
50
51      @Override
52      public Owner getOwnerById(int ownerId) {
53          return ownerRepository.findById(ownerId);
54      }
55
56      @Override
57      public List<Owner> getOwners() {
58          return ownerRepository.findAll();
59      }
60
61      @KafkaListener(id = "deleteOwner", topics = {"owner.delete"},
    containerFactory = "singleFactory")
62      public void deleteOwner(String message) {
63          var data = new GsonBuilder().create().fromJson(message,
    HashMap.class);
64          Integer ownerId = Integer.valueOf((String) data.get("ownerId")
    );
65          ownerRepository.deleteById(ownerId);
66      }
67  }
```

Листинг 1.15: CatWrapper.java

```java
package ru.itmo.kotikiservices.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 1.16: FriendshipWrapper.java

```
1  package ru.itmo.kotikiservices.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 1.17: OwnerWrapper.java

```java
package ru.itmo.kotikiservices.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 1.18: UserWrapper.java

```
1  package ru.itmo.kotikiservices.wrapper;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5
6  @Data
7  @AllArgsConstructor
8  public class UserWrapper {
9      private int id;
10     private String username;
11     private String password;
12     private int roleId;
13     private int ownerId;
14 }
```

Листинг 1.19: KotikiServicesApplication.java

```
1  package ru.itmo.kotikiservices;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class KotikiServicesApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(KotikiServicesApplication.class, args);
11     }
12
13 }
```

Листинг 1.20: build.gradle

```
plugins {
    id 'org.springframework.boot' version '2.7.0'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'ru.itmo'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.kafka:spring-kafka'

    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.security:spring-security-test'
    implementation 'com.google.code.gson:gson:2.9.0'

    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'org.postgresql:postgresql'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

Листинг 1.21: CatController.java

```
package ru.itmo.kotikicontrollers.api;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/cat")
@RequiredArgsConstructor
public class CatController {
    @Autowired
    private KafkaTemplate<Long, String> kafkaTemplate;

    @PostMapping("/create")
    public ResponseEntity<String> createCat(@RequestParam String name,
    @RequestParam String birthday,
                                            @RequestParam String breed,
   @RequestParam int colorId,
                                            @RequestParam int ownerId) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("name", name);
            put("birthday", birthday);
            put("breed", breed);
            put("colorId", String.valueOf(colorId));
            put("ownerId", String.valueOf(ownerId));
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.
   class);
        kafkaTemplate.send("cat.create", message);
        return ResponseEntity.ok().body("Cat is created");
    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> deleteCat(@RequestParam int catId) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("catId", String.valueOf(catId));
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.
   class);
        kafkaTemplate.send("cat.delete", message);
        return ResponseEntity.ok().body("Cat is deleted");
```

```
46          }
47  }
```

Листинг 1.22: FriendshipController.java

```java
package ru.itmo.kotikicontrollers.api;

import com.google.gson.GsonBuilder;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/friendship")
@RequiredArgsConstructor
public class FriendshipController {
    @Autowired
    private KafkaTemplate<Long, String> kafkaTemplate;

    @PostMapping("/create")
    public ResponseEntity<String> createFriendship(@RequestParam int idFirstCat, @RequestParam int idSecondCat) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("idFirstCat", String.valueOf(idFirstCat));
            put("idSecondCat", String.valueOf(idSecondCat));
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.class);
        kafkaTemplate.send("friendship.create", message);
        return ResponseEntity.ok().body("Friendship is created");
    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> deleteFriendship(@RequestParam int friendshipId) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("friendshipId", String.valueOf(friendshipId));
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.class);
        kafkaTemplate.send("friendship.delete", message);
        return ResponseEntity.ok().body("Friendship is deleted");
    }
}
```

Листинг 1.23: OwnerController.java

```
package ru.itmo.kotikicontrollers.api;

import com.google.gson.GsonBuilder;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping(path = "/owner")
@RequiredArgsConstructor
public class OwnerController {
    @Autowired
    private KafkaTemplate<Long, String> kafkaTemplate;

    @PostMapping("/create")
    public ResponseEntity<String> createOwner(@RequestParam String
    name, @RequestParam String birthday) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("name", name);
            put("birthday", birthday);
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.
    class);
        kafkaTemplate.send("owner.create", message);
        return ResponseEntity.ok().body("Owner is created");
    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> deleteOwner(@RequestParam int
    ownerId) {
        Map<String, String> data = new HashMap<String, String>() {{
            put("ownerId", String.valueOf(ownerId));
        }};
        var message = new GsonBuilder().create().toJson(data, HashMap.
    class);
        kafkaTemplate.send("owner.delete", message);
        return ResponseEntity.ok().body("Owner is deleted");
    }
}
```

Листинг 1.24: KafkaProducerConfig.java

```
1  package ru.itmo.kotikicontrollers.kafka;
2
3  import org.apache.kafka.clients.producer.ProducerConfig;
4  import org.apache.kafka.common.serialization.LongSerializer;
5  import org.springframework.beans.factory.annotation.Value;
6  import org.springframework.context.annotation.Bean;
7  import org.springframework.context.annotation.Configuration;
8  import org.springframework.kafka.core.DefaultKafkaProducerFactory;
9  import org.springframework.kafka.core.KafkaTemplate;
10 import org.springframework.kafka.core.ProducerFactory;
11 import org.springframework.kafka.support.converter.
       StringJsonMessageConverter;
12 import org.springframework.kafka.support.serializer.JsonSerializer;
13
14 import java.util.HashMap;
15 import java.util.Map;
16
17 @Configuration
18 public class KafkaProducerConfig {
19
20     @Value("${kafka.server}")
21     private String kafkaServer;
22
23
24     @Bean
25     public Map<String, Object> producerConfigs() {
26         Map<String, Object> props = new HashMap<>();
27         props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaServer
       );
28         props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
       LongSerializer.class);
29         props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
       JsonSerializer.class);
30         return props;
31     }
32
33     @Bean
34     public ProducerFactory<Long, String> producerStarshipFactory() {
35         return new DefaultKafkaProducerFactory<>(producerConfigs());
36     }
37
38     @Bean
39     public KafkaTemplate<Long, String> kafkaTemplate() {
40         KafkaTemplate<Long, String> template = new KafkaTemplate<>(
       producerStarshipFactory());
41         template.setMessageConverter(new StringJsonMessageConverter())
       ;
42         return template;
43     }
```

```
44 }
```

Листинг 1.25: KotikiControllersApplication.java

```
1  package ru.itmo.kotikicontrollers;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class KotikiControllersApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(KotikiControllersApplication.class, args
       );
11     }
12
13 }
```

Листинг 1.26: build.gradle

```
plugins {
    id 'org.springframework.boot' version '2.7.0'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'ru.itmo'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.kafka:spring-kafka'
    implementation 'com.google.code.gson:gson:2.9.0'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-
   test'
}

tasks.named('test') {
    useJUnitPlatform()
}
```