

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №4 по дисциплине   
«Разработка микросервисных приложений»

Обучающийся группы 6232-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Александров А.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc153840658)

[2 Результаты работы 4](#_Toc153840659)

[2.1 Результаты задания 1 4](#_Toc153840660)

[2.2 Результаты задания 2 5](#_Toc153840661)

[2.3 Результаты задания 3 7](#_Toc153840662)

[2.4 Результат задания 4 14](#_Toc153840663)

[3 Вывод 17](#_Toc153840664)

[ПРИЛОЖЕНИЕ А Листинг программы 18](#_Toc153840665)

1. Постановка задачи

**Цель и постановка задания**

Получение навыков асинхронного взаимодействия микросервисов.

**Задание 1**

Добавить в docker-compose файл новый сервис для Kafka.

**Задание 2**

Обновить configuration server, добавив в него новые настройки для Kafka.

**Задание 3**

Создать ендпоинт для удаления компании (компания помечается удаленной и перестает искаться в ендпоинте для поиска компаний, физически из БД компания на данном этапе не удаляется). Если компании по id не существует – кидаем 404 ошибку. При удалении компании должно отправляться сообщение в user сервис через Kafka. При получении сообщения в юзер сервисе, ищутся все пользователи, работающие в данной компании, и поле company\_id для них сбрасывается в null. После этого отправляется сообщение в company service, по которому компания физически удалятся из БД.

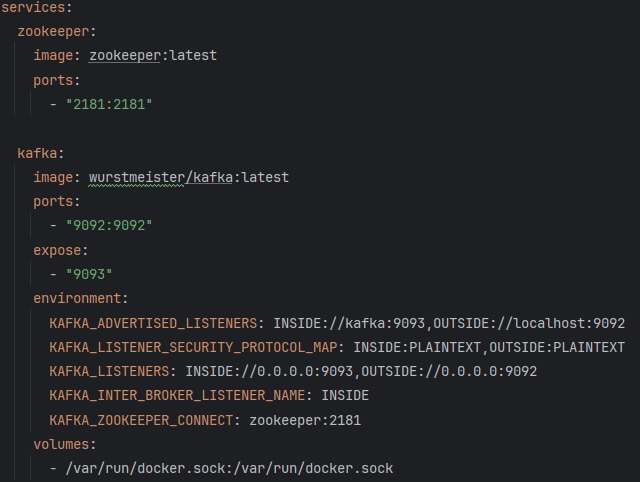
**Задание 4**

Продемонстрировать работу ендпоинта удаления компании.

1. Результаты работы
   1. Результаты задания 1

Добавим в docker-compose файл новый сервис для Kafka.

Настройки для сервиса выглядят следующим образом:

  
Рисунок 1 – Настройки для сервиса Kafka в docker-compose файле

* 1. Результаты задания 2

Обновим configuration server, добавив в него новые настройки для Kafka.

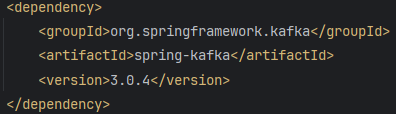
Добавим необходимые настройки в конфигурацию для company-service.yml:

  
Рисунок 2 – Конфигурация для company-service.yml

Добавим необходимые настройки в конфигурацию для user-service.yml:

  
Рисунок 3 – Конфигурация для user-service.yml

Для того чтобы Kafka работала в наших сервисах нужно добавить необходимые зависимости в pom.xml для наших сервисов (рисунок 4).

  
Рисунок 4 – Добавлены необходимые зависимости в pom.xml для company-service и user-service

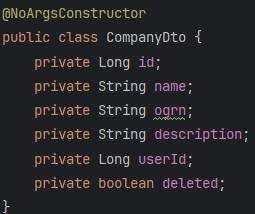
* 1. Результаты задания 3

Необходимо создать ендпоинт для удаления компании (компания помечается удаленной и перестает искаться в ендпоинте для поиска компаний, физически из БД компания на данном этапе не удаляется). Если компании по id не существует – кидаем 404 ошибку. При удалении компании должно отправляться сообщение в user сервис через Kafka. При получении сообщения в юзер сервисе, ищутся все пользователи, работающие в данной компании, и поле company\_id для них сбрасывается в null. После этого отправляется сообщение в company service, по которому компания физически удалятся из БД.

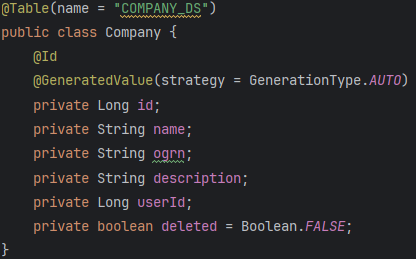
На рисунке 5 показан ендпоинт для удаления компании.

  
Рисунок 5 – Ендпоинт для удаления компании в CompanyController

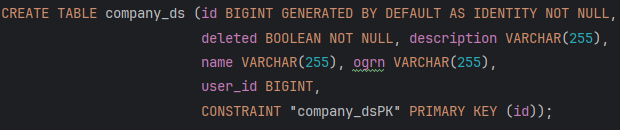
Для того чтобы пометить компанию удаленной введем новое поле в БД (private boolean deleted). Теперь CompanyDto выглядит следующим образом:

  
Рисунок 6 – CompanyDto

На рисунке 7 отображена таблица для компаний (Entity для company-service).

  
Рисунок 7 – Company

Файл с миграциями выглядит следующим образом:

  
Рисунок 8 – migration.sql для company-service

На рисунках 9-10 представлена реализация Service для company-service.

На рисунках 11-13 представлена реализация Service для user-service.

  
Рисунок 9 – Service для company-service

  
Рисунок 10 – Service для company-service

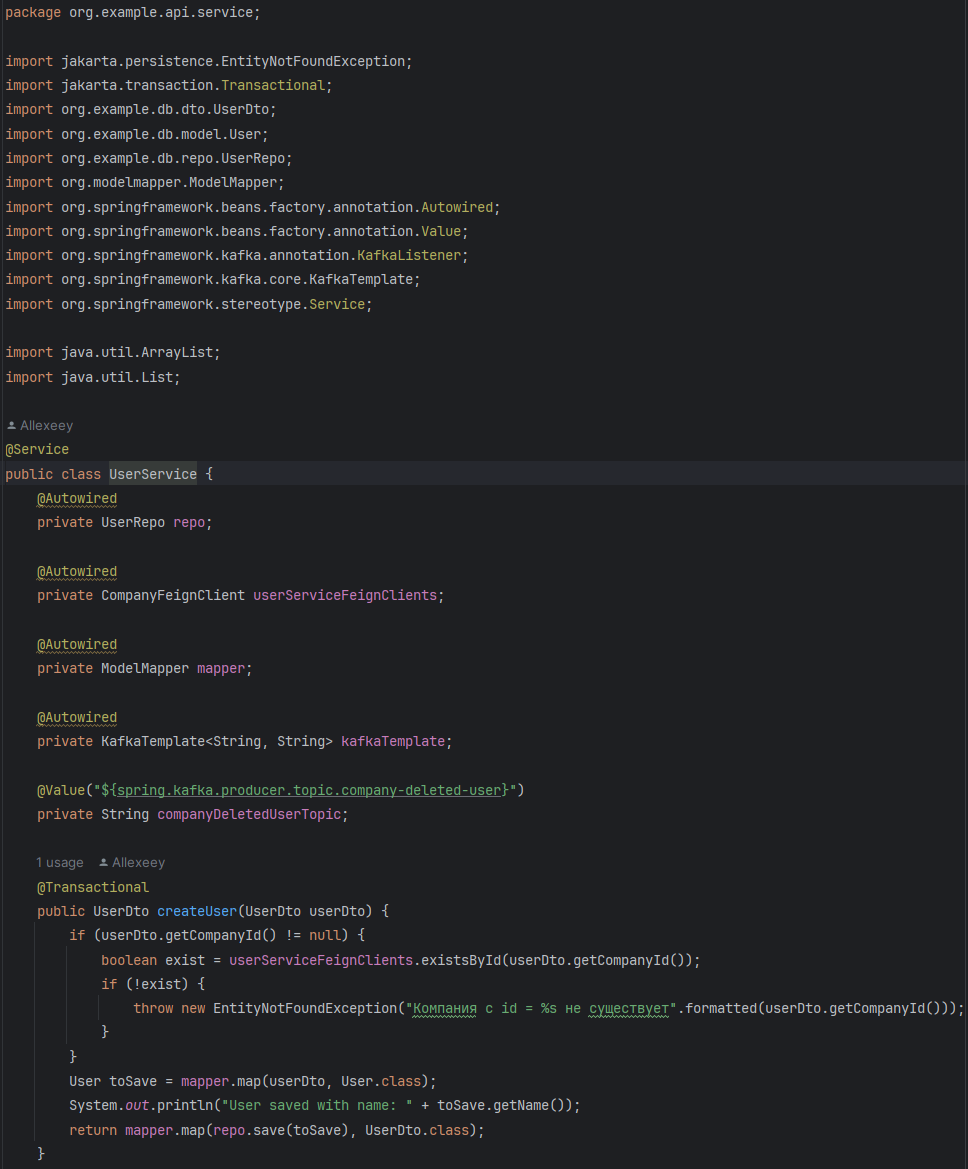
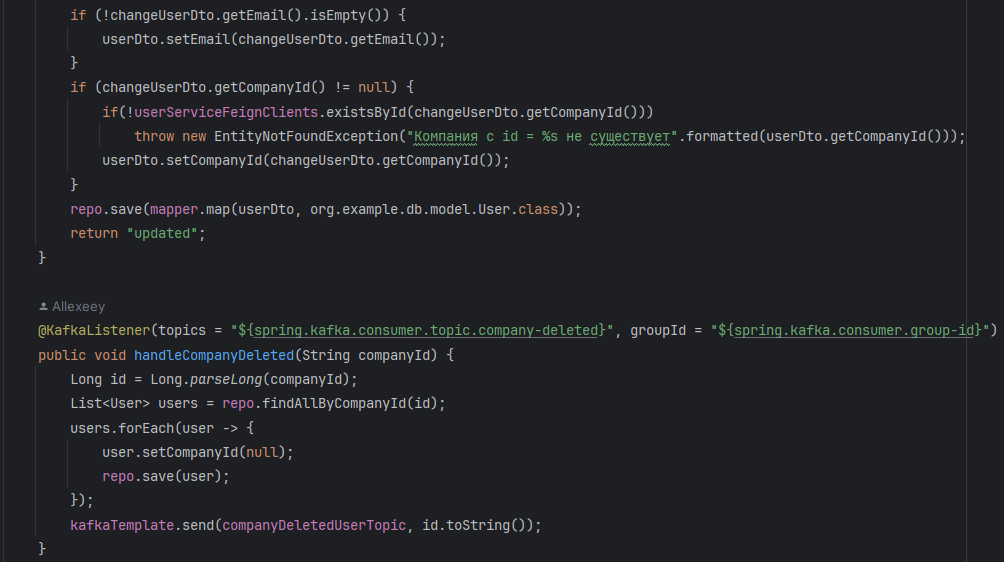
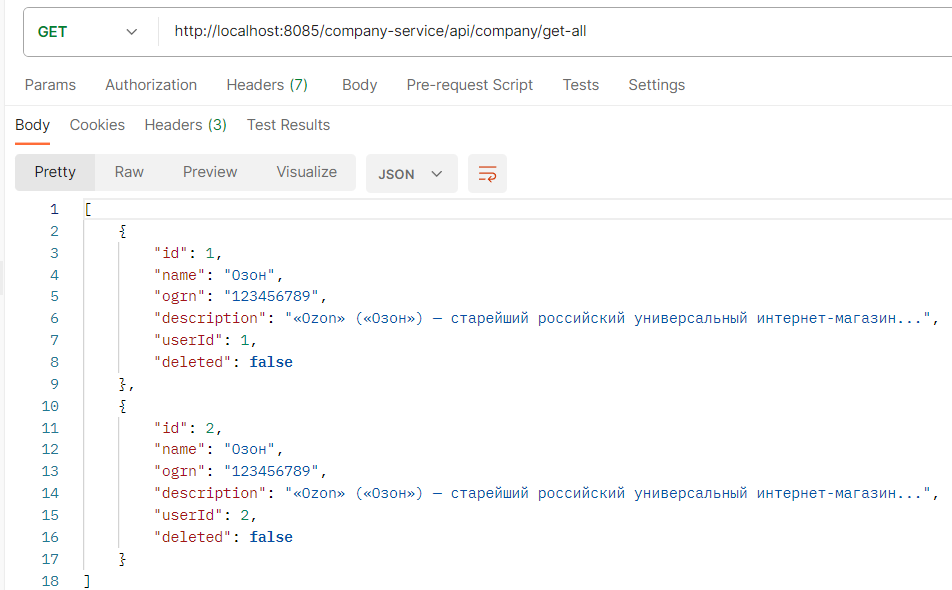
  
Рисунок 11 – Service для user-service

  
Рисунок 12 – Service для user-service

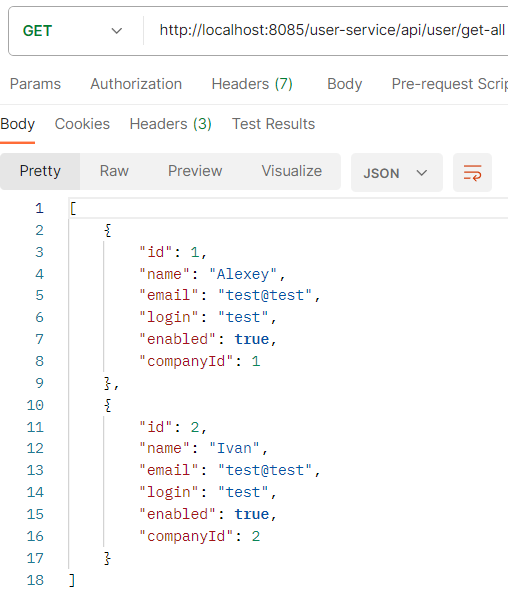
  
Рисунок 13 – Service для user-service

* 1. Результат задания 4

Для проверки работоспособности удаления компании получим список всех компаний (осуществим запрос через api gateway):

  
Рисунок 14 – Список всех существующих компаний в нашей БД.

Получим список всех сотрудников:

  
Рисунок 15 – Список всех существующих сотрудников в нашей БД.

Продемонстрируем результаты проделанной работы для ендпоинта удаления компании.

Конечная точка доступна через Postman (рисунок 16).



Рисунок 16 – Конечная точка для удаления компании

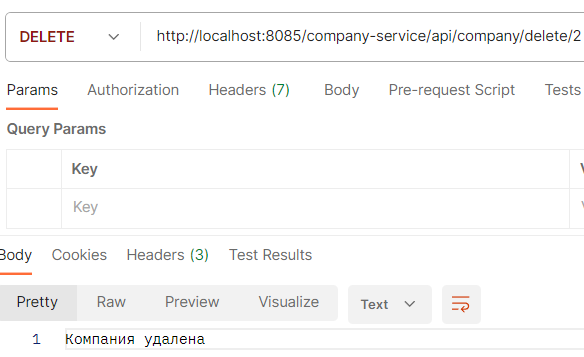
Попробуем удалить несуществующую компанию (id=10):

  
Рисунок 17 – Удаление несуществующей компании

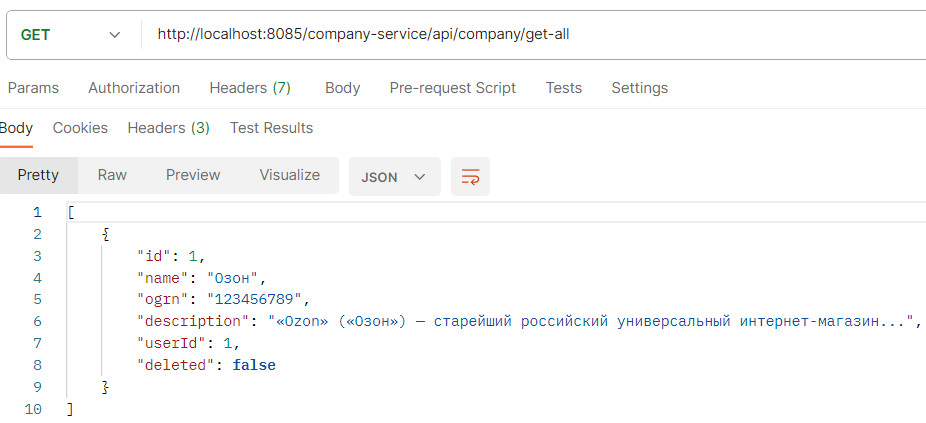
Получаем сообщение об ошибке:

  
Рисунок 17 – Сообщение об ошибке при удалении несуществующей компании

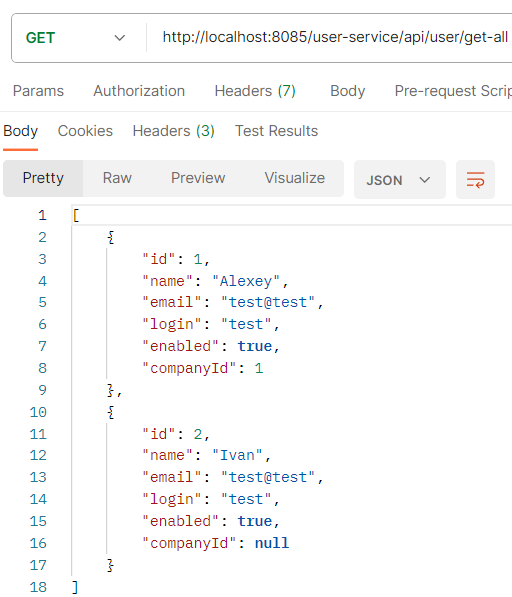
Попробуем удалить существующую компанию (id=2):

  
Рисунок 18 – Удаление существующей компании

Получим список всех компаний:

  
Рисунок 19 – Список всех существующих компаний в нашей БД.

Получим список всех сотрудников:

  
Рисунок 20 – Список всех существующих сотрудников в нашей БД.

Видим, что компания удалена из БД и для сотрудника №2 поле companyId сбросилось в null.

1. Вывод

В результате выполнения лабораторной работы добавили в docker-compose файл новый сервис для Kafka, а также обновили configuration server, добавив в него новые настройки для Kafka.

Создан ендпоинт для удаления компании с использованием Kafka. Продемонстрирована работа ендпоинта удаления компании.

Написан отчёт.

[Исходный код доступен по ссылке](https://github.com/Allexeey/microservices/tree/lab-4)

ПРИЛОЖЕНИЕ А  
Листинг программы

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.1.4</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <groupId>org.example</groupId>  
 <artifactId>microservices</artifactId>  
 <packaging>pom</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <modules>  
 <module>spring-cloud-server</module>  
 <module>user-service</module>  
 <module>company-service</module>  
 <module>discovery-server</module>  
 <module>gateway-service</module>  
 </modules>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
  
  
</project>

version: "3.9"  
  
services:  
 zookeeper:  
 image: zookeeper:latest  
 ports:  
 - "2181:2181"  
  
 kafka:  
 image: wurstmeister/kafka:latest  
 ports:  
 - "9092:9092"  
 expose:  
 - "9093"  
 environment:  
 KAFKA\_ADVERTISED\_LISTENERS: INSIDE://kafka:9093,OUTSIDE://localhost:9092  
 KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT  
 KAFKA\_LISTENERS: INSIDE://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092  
 KAFKA\_INTER\_BROKER\_LISTENER\_NAME: INSIDE  
 KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181  
 volumes:  
 - /var/run/docker.sock:/var/run/docker.sock  
  
 digital-plant-sql-db:  
 image: postgres:15.3-alpine  
 container\_name: microservices-sql-db  
 environment:  
 POSTGRES\_DB: "microservices"  
 POSTGRES\_USER: "user"  
 POSTGRES\_PASSWORD: "password"  
 healthcheck:  
 test: [ "CMD-SHELL", "pg\_isready -U user -D microservices" ]  
 interval: 5s  
 timeout: 2s  
 retries: 25  
 ports:  
 - "5433:5432"  
 volumes:  
 - db\_microservices\_data:/var/lib/postgresql/data  
 command: [ "postgres", "-c", "max-connections=400" ]  
  
volumes:  
 db\_microservices\_data: { }

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.1.2</version>  
 <relativePath/>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
 <description>user-service</description>  
 <artifactId>user-service</artifactId>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <spring-cloud.version>2022.0.4</spring-cloud.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 <exclusions>  
 <exclusion>  
 <groupId>org.junit.vintage</groupId>  
 <artifactId>junit-vintage-engine</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.postgresql</groupId>  
 <artifactId>postgresql</artifactId>  
 <version>42.2.27</version>  
 </dependency>  
 <dependency>  
 <groupId>org.liquibase</groupId>  
 <artifactId>liquibase-core</artifactId>  
 <version>4.24.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.liquibase.ext</groupId>  
 <artifactId>liquibase-hibernate6</artifactId>  
 <version>4.24.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-bootstrap</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.modelmapper</groupId>  
 <artifactId>modelmapper</artifactId>  
 <version>3.2.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.kafka</groupId>  
 <artifactId>spring-kafka</artifactId>  
 <version>3.0.4</version>  
 </dependency>  
 </dependencies>  
  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>  
  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 <plugin>  
 <groupId>org.liquibase</groupId>  
 <artifactId>liquibase-maven-plugin</artifactId>  
 <version>4.24.0</version>  
 <configuration>  
 <propertyFile>src/main/resources/liquibase.properties</propertyFile>  
 <dataDir>data</dataDir>  
 <driver>org.postgresql.Driver</driver>  
 <outputChangeLogFile>src/main/resources/db/migration.sql</outputChangeLogFile>  
 <diffChangeLogFile>src/main/resources/db/migration.sql</diffChangeLogFile>  
 <!-- log -->  
 <verbose>true</verbose>  
 <logging>debug</logging>  
 <promptOnNonLocalDatabase>false</promptOnNonLocalDatabase>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

@Service  
public class UserService {  
 @Autowired  
 private UserRepo repo;  
  
 @Autowired  
 private CompanyFeignClient userServiceFeignClients;  
  
 @Autowired  
 private ModelMapper mapper;  
  
 @Autowired  
 private KafkaTemplate<String, String> kafkaTemplate;  
  
 @Value("${spring.kafka.producer.topic.company-deleted-user}")  
 private String companyDeletedUserTopic;  
  
 @Transactional  
 public UserDto createUser(UserDto userDto) {  
 if (userDto.getCompanyId() != null) {  
 boolean exist = userServiceFeignClients.existsById(userDto.getCompanyId());  
 if (!exist) {  
 throw new EntityNotFoundException("Компания с id = %s не существует".formatted(userDto.getCompanyId()));  
 }  
 }  
 User toSave = mapper.map(userDto, User.class);  
 System.*out*.println("User saved with name: " + toSave.getName());  
 return mapper.map(repo.save(toSave), UserDto.class);  
 }  
  
 public Boolean existsById(Long id) {  
 User User = repo.findById(id)  
 .orElseThrow(() -> new EntityNotFoundException("Пользователь с id: " + id + " - не существует"));  
  
 if (!User.isEnabled())  
 throw new EntityNotFoundException("Пользователь с id: " + id + " - не активен");  
  
 return true;  
 }  
  
 public List<UserDto> getAllUsers() {  
 List<User> Users = repo.findAll();  
 List<UserDto> userDtos = new ArrayList<>();  
 for (User entity : Users) {  
 userDtos.add(mapper.map(entity, UserDto.class));  
 }  
 return userDtos;  
 }  
  
 @Transactional  
 public String changeEnabled(Long id) {  
 User User = repo.findById(id)  
 .orElseThrow(() -> new EntityNotFoundException("Пользователь с id: " + id + " - не существует"));  
  
 UserDto userDto = mapper.map(User, UserDto.class);  
 if (userDto.isEnabled()) {  
 userDto.setEnabled(false);  
 repo.save(mapper.map(userDto, org.example.db.model.User.class));  
 return "disabled";  
 } else {  
 userDto.setEnabled(true);  
 repo.save(mapper.map(userDto, org.example.db.model.User.class));  
 return "enabled";  
 }  
 }  
  
 @Transactional  
 public String changeUser(Long id, UserDto changeUserDto) {  
 User User = repo.findById(id)  
 .orElseThrow(() -> new EntityNotFoundException("Пользователь с id: " + id + " - не существует"));  
  
 UserDto userDto = mapper.map(User, UserDto.class);  
 if (!changeUserDto.getName().isEmpty()) {  
 userDto.setName(changeUserDto.getName());  
 }  
 if (!changeUserDto.getEmail().isEmpty()) {  
 userDto.setEmail(changeUserDto.getEmail());  
 }  
 if (changeUserDto.getCompanyId() != null) {  
 if(!userServiceFeignClients.existsById(changeUserDto.getCompanyId()))  
 throw new EntityNotFoundException("Компания с id = %s не существует".formatted(userDto.getCompanyId()));  
 userDto.setCompanyId(changeUserDto.getCompanyId());  
 }  
 repo.save(mapper.map(userDto, org.example.db.model.User.class));  
 return "updated";  
 }  
  
 @KafkaListener(topics = "${spring.kafka.consumer.topic.company-deleted}", groupId = "${spring.kafka.consumer.group-id}")  
 public void handleCompanyDeleted(String companyId) {  
 Long id = Long.*parseLong*(companyId);  
 List<User> users = repo.findAllByCompanyId(id);  
 users.forEach(user -> {  
 user.setCompanyId(null);  
 repo.save(user);  
 });  
 kafkaTemplate.send(companyDeletedUserTopic, id.toString());  
 }  
}

package org.example.api.service;  
  
import org.springframework.cloud.openfeign.FeignClient;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
  
@FeignClient(  
 name = "company-service",  
 path = "/company-service/api/company",  
 url = "localhost:8085"  
)  
public interface CompanyFeignClient {  
 @GetMapping("/exists-by-id/{companyId}")  
 Boolean existsById (@PathVariable Long companyId);  
}

package org.example.api;  
  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping("/user-service")  
public class ConfigController {  
  
 @Value("${description}")  
 private String description;  
  
 @GetMapping  
 public String getTestProperty() {  
 return description;  
 }  
}

package org.example.api;  
  
import org.example.api.service.UserService;  
import org.example.db.dto.UserDto;  
import org.example.db.model.User;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/user")  
public class UserController {  
  
 @Autowired  
 private UserService userService;  
  
 @PostMapping("/create")  
 public ResponseEntity<UserDto> createUser(@RequestBody UserDto userDto) {  
 return ResponseEntity.*ok*().body(userService.createUser(userDto));  
 }  
  
 @GetMapping("/exists-by-id/{id}")  
 public Boolean existsById(@PathVariable Long id) {  
 return userService.existsById(id);  
 }  
  
 @GetMapping("/get-all")  
 public List<UserDto> getAllUsers() {  
 return userService.getAllUsers();  
 }  
  
 @PatchMapping("/change-enabled/{id}")  
 public String changeEnabled(@PathVariable Long id) {  
 return userService.changeEnabled(id);  
 }  
  
 @PatchMapping("/change-user/{id}")  
 public String changeUser(@PathVariable Long id, @RequestBody UserDto userDto) {  
 return userService.changeUser(id, userDto);  
 }  
  
}

package org.example;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
import org.springframework.cloud.openfeign.EnableFeignClients;  
  
@SpringBootApplication  
@EnableFeignClients  
@EnableDiscoveryClient  
public class Main {  
 public static void main(String[] args) {  
 SpringApplication.*run*(Main.class, args);  
 }  
}

package org.example.db.dto;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@NoArgsConstructor  
public class UserDto {  
 private Long id;  
 private String name;  
 private String email;  
 private String login;  
 @JsonIgnore  
 private String password;  
 private boolean enabled;  
 private Long companyId;  
}