

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 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гижевская В.Д.

Обучающийся группы 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алкеев М.Г.

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

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146727791)

[2 Результаты работы 4](#_Toc146727792)

[ПРИЛОЖЕНИЕ А Листинг программы](#_Toc146727796) 8

1. Постановка задачи

Получение навыков асинхронного взаимодействия микросервисов.

1. Добавить в docker-compose файл новый сервис для Kafka.
2. Обновить configuration server, добавив в него новые настройки для kafka.
3. Создать енд-поинт для удаления компании (компания помечается удаленной и перестает искаться в енд-поинте 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.
4. Продемонстрировать работу енд-поинта удаления компании.
5. Результаты работы

На рисунках 1-5 представлены результаты работы.

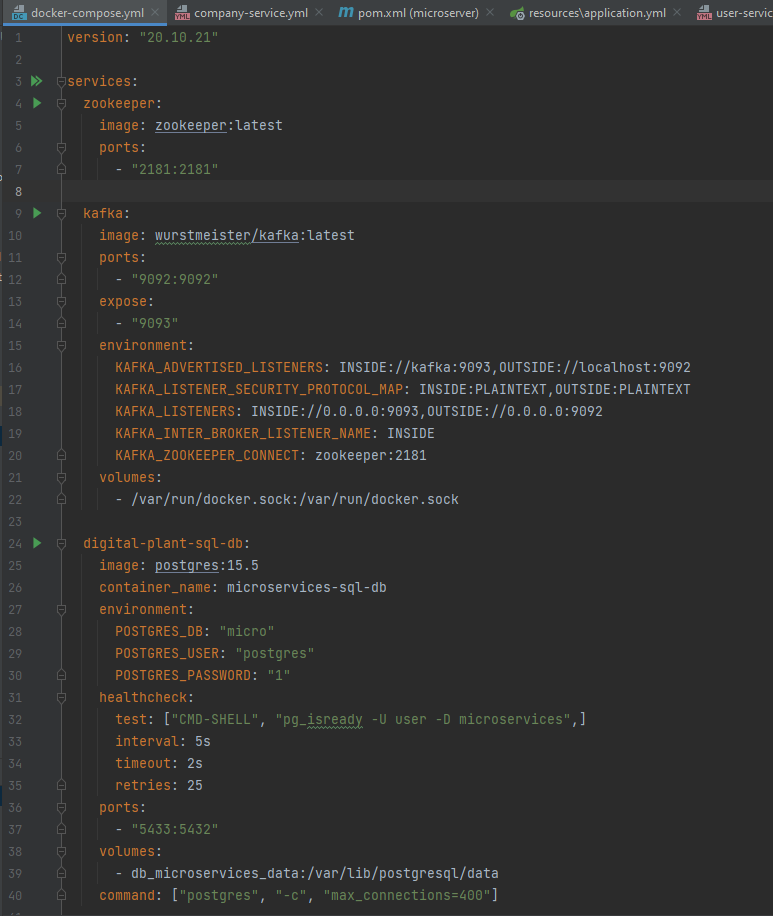
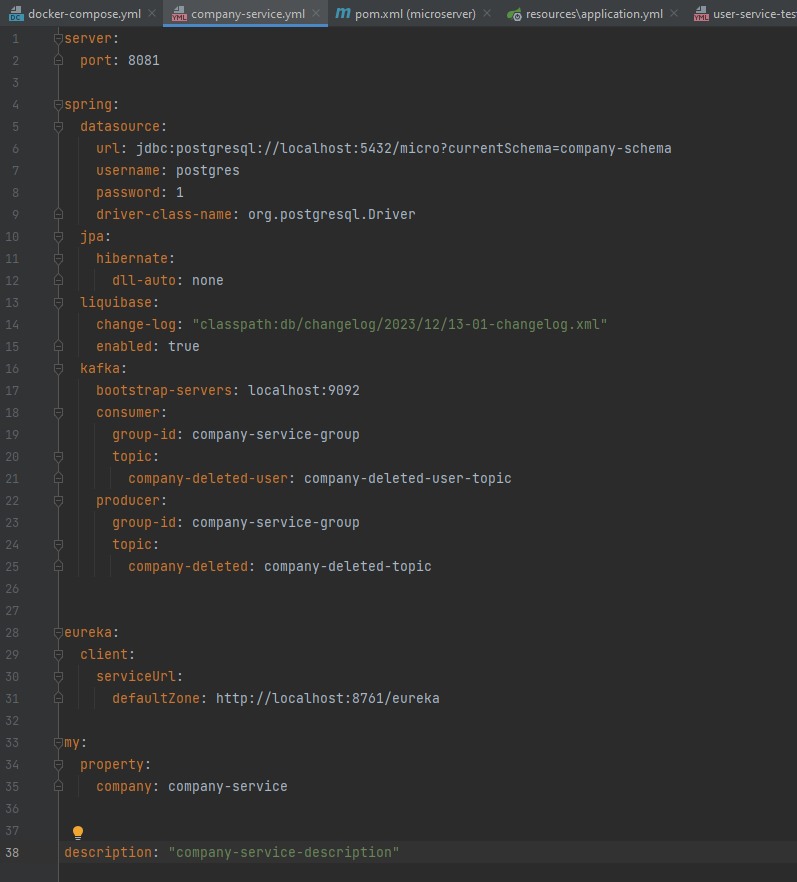


Рисунок 1 – конфиг сервиса kafka в docker-compose файл

Рисунок 2 – конфиг company-service

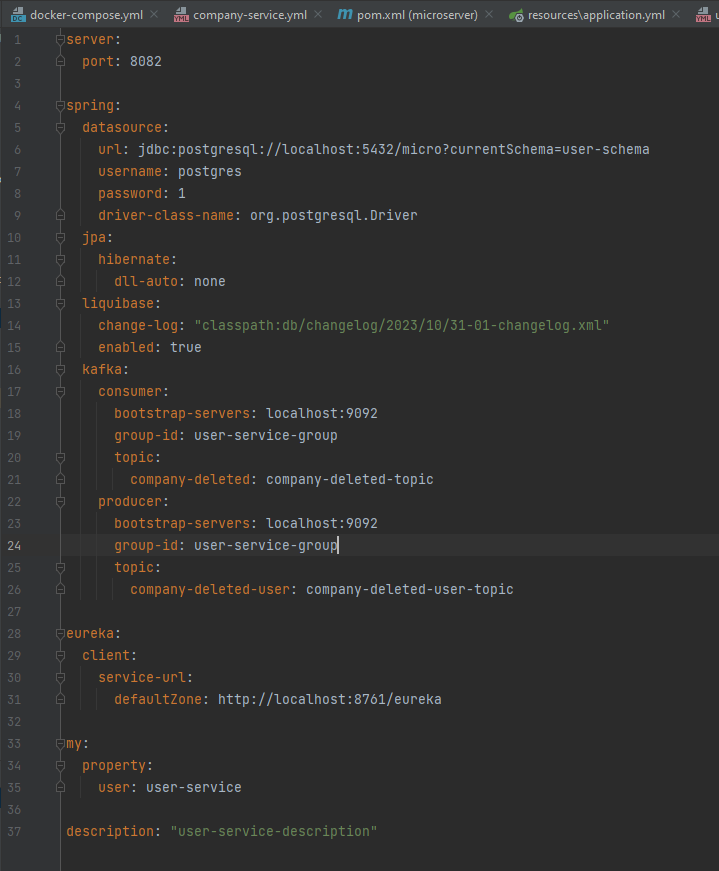


Рисунок 3 – конфиг user-service



Рисунок 3 –метод в company-service для мягкого удаления компании

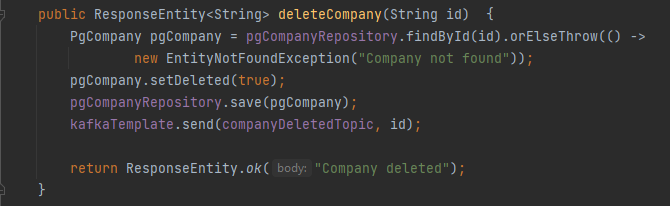


Рисунок 4 –метод в company-service для физического удаления компании

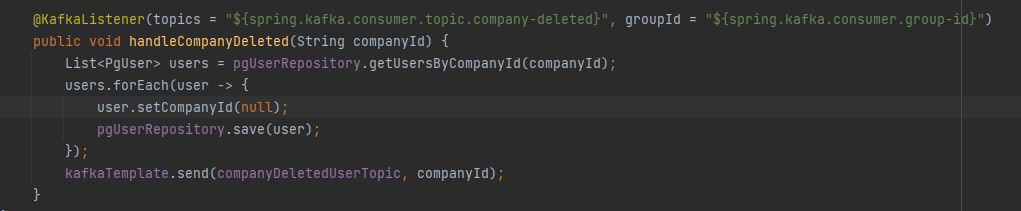


Рисунок 5 –метод в user-service для установки в null id компании пользователя и отправки сообщений в company-service

ПРИЛОЖЕНИЕ А  
Листинг программы

server:

port: 8040

spring:

application:

name: "apiGateway"

cloud:

gateway:

routes:

- id: company-service

uri: lb://company-service

predicates:

- Path=/company/

filters:

- StripPrefix=1

- id: user-service

uri: lb://user-service

predicates:

- Path=/user/

filters:

- StripPrefix=1

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

package com.labs.companyserv.service;

import com.labs.companyserv.entity.PgCompany;

import com.labs.companyserv.model.CompanyDto;

import com.labs.companyserv.model.converter.CompanyDtoConverter;

import com.labs.companyserv.repository.PgCompanyRepository;

import jakarta.persistence.EntityNotFoundException;

import lombok.RequiredArgsConstructor;

import lombok.extern.slf4j.Slf4j;

import org.apache.kafka.clients.admin.NewTopic;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.http.ResponseEntity;

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.kafka.config.TopicBuilder;

import org.springframework.kafka.core.KafkaTemplate;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

@Slf4j

@Service

@RequiredArgsConstructor

public class CompanyService {

@Autowired

private PgCompanyRepository pgCompanyRepository;

@Autowired

private CompanyServiceFeignClients companyServiceFeignClients;

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

@Value("${spring.kafka.producer.topic.company-deleted}")

private String companyDeletedTopic;

@Transactional

public String createCompany(CompanyDto companyDto) {

Boolean exist = companyServiceFeignClients.existsById(companyDto.getDirectorId());

if(!exist) {

throw new EntityNotFoundException("Director with id = %s does not exist".formatted(companyDto.getDirectorId()));

}

return pgCompanyRepository.save(CompanyDtoConverter.toEntity(companyDto)).getId();

}

public Boolean existsById(String id) {

Optional<PgCompany> pgCompany = pgCompanyRepository.findById(id);

EntityNotFoundException exception = new EntityNotFoundException("Company not found");;

if(pgCompany.isPresent()) {

if(!pgCompany.get().isDeleted())

return pgCompanyRepository.existsById(id);

else

throw exception;

}

else

throw exception;

}

public CompanyDto getById(String id) {

EntityNotFoundException exception = new EntityNotFoundException(

"Company with id: " + id + " - does not exist");

CompanyDto companyDto = CompanyDtoConverter

.toDto(pgCompanyRepository.findById(id)

.orElseThrow(() -> exception));

if(!companyDto.isDeleted())

return companyDto;

else

throw exception;

}

public List<CompanyDto> getAllCompanies() {

List<PgCompany> pgCompanies = pgCompanyRepository.findAll();

List<CompanyDto> companyDtos = new ArrayList<>();

for (PgCompany entity : pgCompanies) {

CompanyDto companyDto = CompanyDtoConverter.toDto(entity);

companyDto.setDirectorName(companyServiceFeignClients.getDirName(entity.getDirectorId()));

companyDtos.add(companyDto);

}

return companyDtos;

}

public ResponseEntity<String> deleteCompany(String id) {

PgCompany pgCompany = pgCompanyRepository.findById(id).orElseThrow(() ->

new EntityNotFoundException("Company not found"));

pgCompany.setDeleted(true);

pgCompanyRepository.save(pgCompany);

kafkaTemplate.send(companyDeletedTopic, id);

return ResponseEntity.ok("Company deleted");

}

@KafkaListener(topics = "${spring.kafka.consumer.topic.company-deleted-user}",

groupId = "${spring.kafka.consumer.group-id}")

public void handleCompanyDeleted(String id) {

Optional<PgCompany> pgCompany = pgCompanyRepository.findById(id);

if(pgCompany.isPresent())

pgCompanyRepository.delete(pgCompany.get());

else

throw new EntityNotFoundException("Company not found");

}

}

spring:

application:

name: "discoveryServer"

server:

port: 8761

eureka:

client:

registerWithEureka: false

fetch-registry: false

server:

port: 8082

spring:

datasource:

url: jdbc:postgresql://localhost:5432/micro?currentSchema=user-schema

username: postgres

password: 1

driver-class-name: org.postgresql.Driver

jpa:

hibernate:

dll-auto: none

liquibase:

change-log: "classpath:db/changelog/2023/10/31-01-changelog.xml"

enabled: true

kafka:

consumer:

bootstrap-servers: localhost:9092

group-id: user-service-group

topic:

company-deleted: company-deleted-topic

producer:

bootstrap-servers: localhost:9092

group-id: user-service-group

topic:

company-deleted-user: company-deleted-user-topic

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

my:

property:

user: user-service

description: "user-service-description"

server:

port: 8081

spring:

datasource:

url: jdbc:postgresql://localhost:5432/micro?currentSchema=company-schema

username: postgres

password: 1

driver-class-name: org.postgresql.Driver

jpa:

hibernate:

dll-auto: none

liquibase:

change-log: "classpath:db/changelog/2023/12/13-01-changelog.xml"

enabled: true

kafka:

bootstrap-servers: localhost:9092

consumer:

group-id: company-service-group

topic:

company-deleted-user: company-deleted-user-topic

producer:

group-id: company-service-group

topic:

company-deleted: company-deleted-topic

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka

my:

property:

company: company-service

description: "company-service-description"

package com.labs.userservice.service;

import com.labs.userservice.entity.PgUser;

import com.labs.userservice.model.ChangeUserDto;

import com.labs.userservice.model.CompanyInfoDto;

import com.labs.userservice.model.UserDto;

import com.labs.userservice.model.converter.UserDtoConverter;

import com.labs.userservice.repository.PgUserRepository;

import jakarta.persistence.EntityNotFoundException;

import lombok.RequiredArgsConstructor;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.kafka.core.KafkaTemplate;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.ArrayList;

import java.util.List;

@Slf4j

@Service

public class UserService {

@Autowired

private PgUserRepository pgUserRepository;

@Autowired

private UserServiceFeignClients userServiceFeignClients;

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

@Value("${spring.kafka.producer.topic.company-deleted-user}")

private String companyDeletedUserTopic;

@Transactional

public String createUser(UserDto userDto) {

return pgUserRepository.save(UserDtoConverter.toEntity(userDto)).getId();

}

public Boolean existsById(String id) {

PgUser pgUser = pgUserRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("User with id: " + id + " does not exist"));

if (!pgUser.getEnabled())

throw new EntityNotFoundException("User with id: " + id + "does not active");

return true;

}

public String getName(String id) {

PgUser pgUser = pgUserRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("User with id: " + id + " does not exist"));

return UserDtoConverter.toDto(pgUser).getName();

}

public List<UserDto> getAllUsers() {

List<PgUser> pgUsers = pgUserRepository.findAll();

List<CompanyInfoDto> companyInfoDtos = userServiceFeignClients.getAllCompanies();

List<UserDto> userDtos = new ArrayList<>();

for (int i = 0; i < pgUsers.size(); i++) {

UserDto userDto = UserDtoConverter.toDto(pgUsers.get(i));

userDto.setCompanyInfoDto(userServiceFeignClients.getById(userDto.getCompanyId()));

userDtos.add(userDto);

}

return userDtos;

}

@Transactional

public String changeEnabled(String id) {

PgUser pgUser = pgUserRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("User with id: " + id + "does not exist"));

UserDto userDto = UserDtoConverter.toDto(pgUser);

if (userDto.getEnabled()) {

userDto.setEnabled(false);

pgUserRepository.save(UserDtoConverter.toEntity(userDto));

return "disabled";

} else {

userDto.setEnabled(true);

pgUserRepository.save(UserDtoConverter.toEntity(userDto));

return "enabled";

}

}

@Transactional

public String changeUser(String id, ChangeUserDto changeUserDto) {

PgUser pgUser = pgUserRepository.findById(id)

.orElseThrow(() -> new EntityNotFoundException("User with id: " + id + "does not exist"));

UserDto userDto = UserDtoConverter.toDto(pgUser);

if (!changeUserDto.getName().equals("")) {

userDto.setName(changeUserDto.getName());

}

if (!changeUserDto.getEmail().equals("")) {

userDto.setEmail(changeUserDto.getEmail());

}

if (!changeUserDto.getCompanyId().equals("")) {

if(!userServiceFeignClients.existsById(changeUserDto.getCompanyId()))

throw new EntityNotFoundException("Company with id = %s does not exist".formatted(userDto.getCompanyId()));

userDto.setCompanyId(changeUserDto.getCompanyId());

}

pgUserRepository.save(UserDtoConverter.toEntity(userDto));

return "updated";

}

@KafkaListener(topics = "${spring.kafka.consumer.topic.company-deleted}", groupId = "${spring.kafka.consumer.group-id}")

public void handleCompanyDeleted(String companyId) {

List<PgUser> users = pgUserRepository.getUsersByCompanyId(companyId);

users.forEach(user -> {

user.setCompanyId(null);

pgUserRepository.save(user);

});

kafkaTemplate.send(companyDeletedUserTopic, companyId);

}

}