

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 №3 по дисциплине   
«Разработка микросервисных приложений»

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

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

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147329731)

[2 Результаты работы 5](#_Toc147329732)

[2.1 Результаты задания 1 5](#_Toc147329733)

[2.2 Результаты задания 2 9](#_Toc147329734)

[2.3 Результаты задания 3 11](#_Toc147329735)

[2.4 Результат задания 4 17](#_Toc147329736)

[3 Вывод 18](#_Toc147329737)

[ПРИЛОЖЕНИЕ А Листинг программы 19](#_Toc147329738)

1. Постановка задачи

Цель и постановка задания

В процессе выполнения заданий подключить службу обнаружения микросервисов и настроить api шлюз

**Задание 1**

Создать новое Spring Boot приложение – Discovery server. В качестве службы обнаружения микросервисов можно использовать eureka.

**Задание 2**

Зарегистрируйте ваши микросервисы (user и company) в созданном Discovery server.

**Задание 3**

Создайте новое Spring boot приложение – Api Gateway. Зарегистрируйте его в Discovery server

**Задание 4**

Задайте маршруты в Api gateway таким образом, чтобы все запросы, которые идут по адресу …/user/… были перенаправлены в user-service, а все запросы по адресу …/company/… в company-service. Проверьте работоспособность приложения.

**Задание 5**

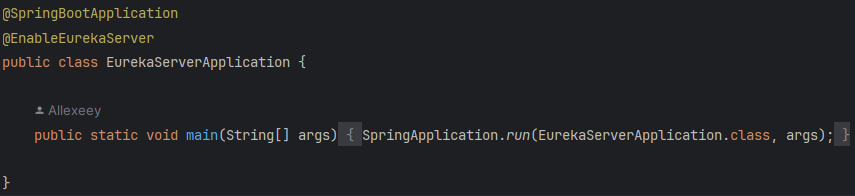
Поправьте url в своих feign clients, таким образом, чтобы вызов сервисов осуществлялся через api gateway. Проверьте работоспособность приложения.

1. Результаты работы
   1. Результаты задания 1

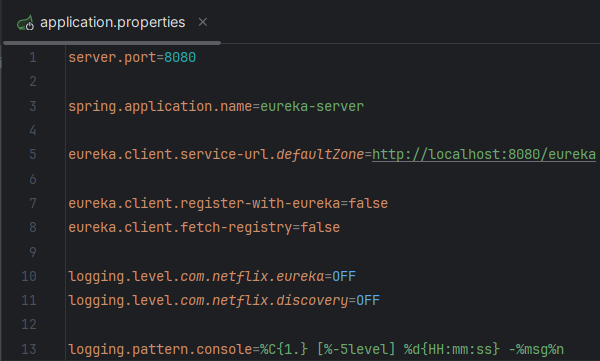
Подключим службу обнаружения микросервисов. Для этого создадим новое Spring Boot приложение – Discovery server Добавим необходимые библиотеки в pom.xml:



Добавим в наше приложение EurekaServerApplication необходимые анотации:



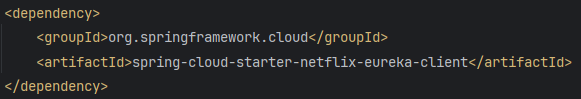
Настройки сервера:



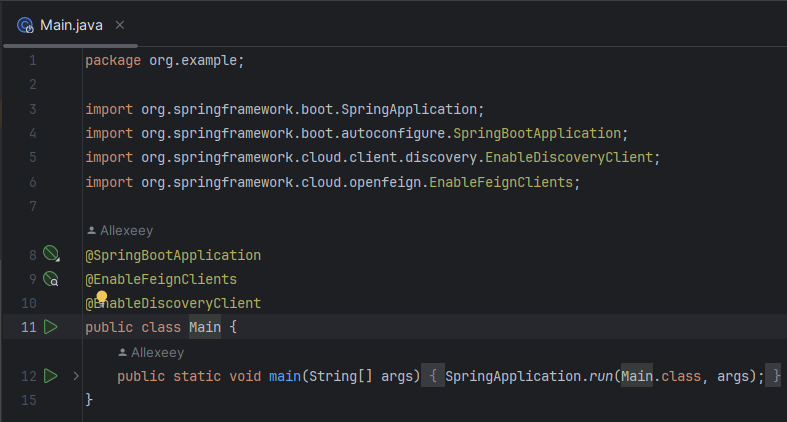
* 1. Результаты задания 2

Зарегистрируем наши микросервисы (user и company) в созданном Discovery server.

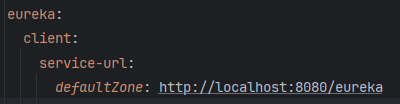
Добавим необходимые зависимости в pom.xml:



Добавим в наши приложения (user и company) необходимые анотации:



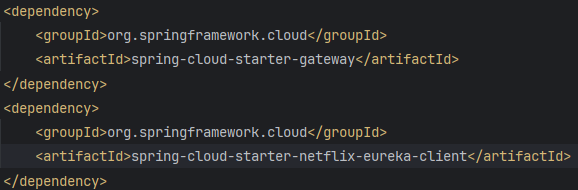
Добавим необходимые настройки для наших сервисов:



* 1. Результаты задания 3

Создим новое Spring boot приложение – Api Gateway. Зарегистрируем его в Discovery server

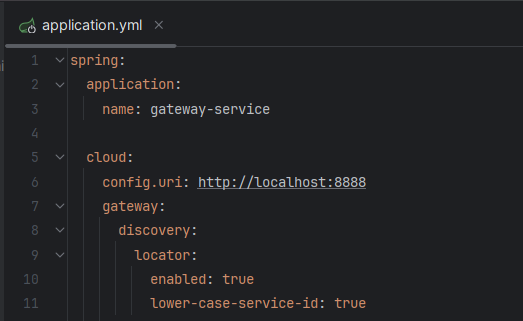
Добавим необходимые зависимости в pom.xml:

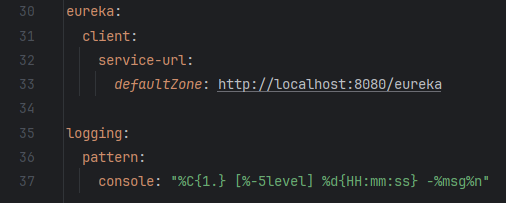


Добавим в наше приложение необходимые анотации:



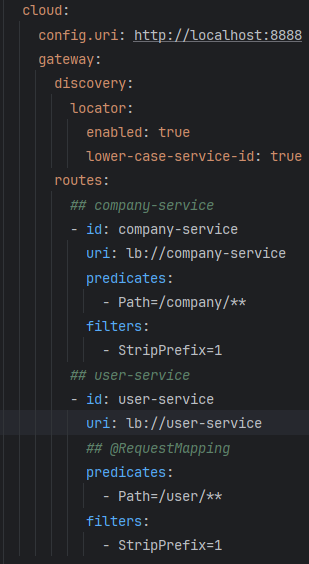
Добавим необходимые настройки для нашего сервиса:



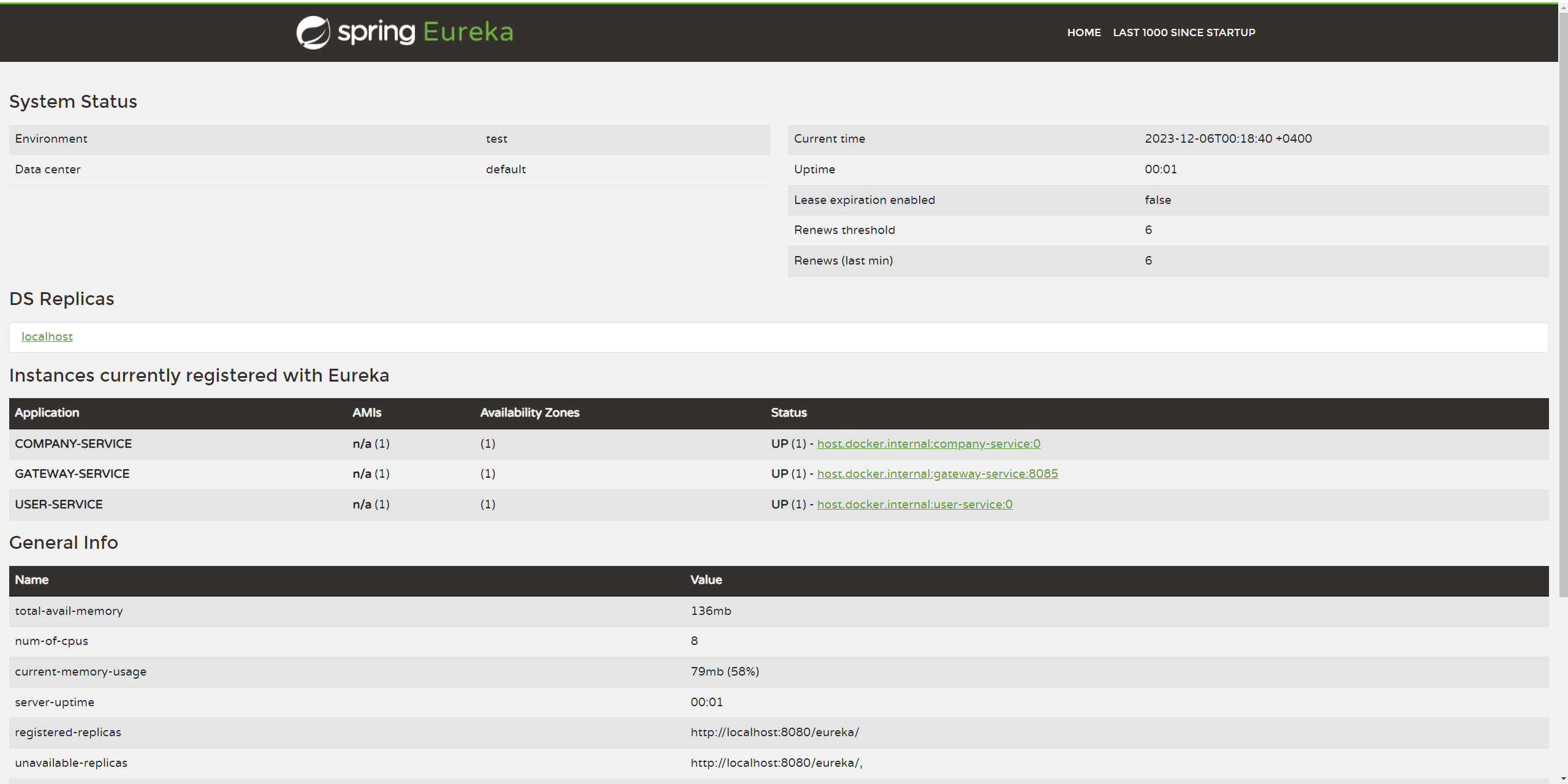


* 1. Результат задания 4 и 5

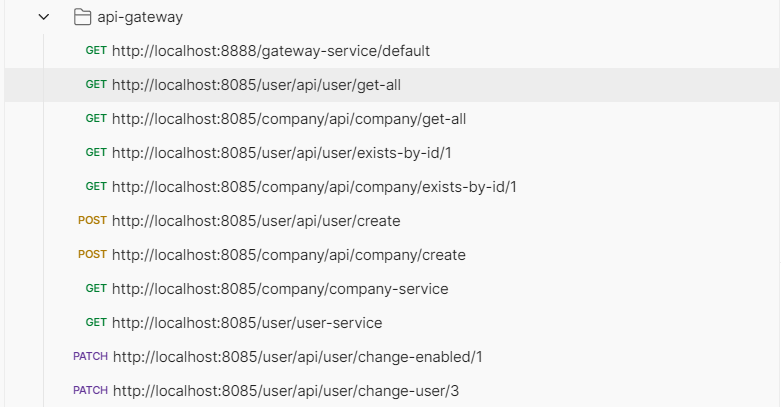
Зададим маршруты в Api gateway таким образом, чтобы все запросы, которые идут по адресу …/user/… были перенаправлены в user-service, а все запросы по адресу …/company/… в company-service. А также проверим работоспособность нашего приложения.



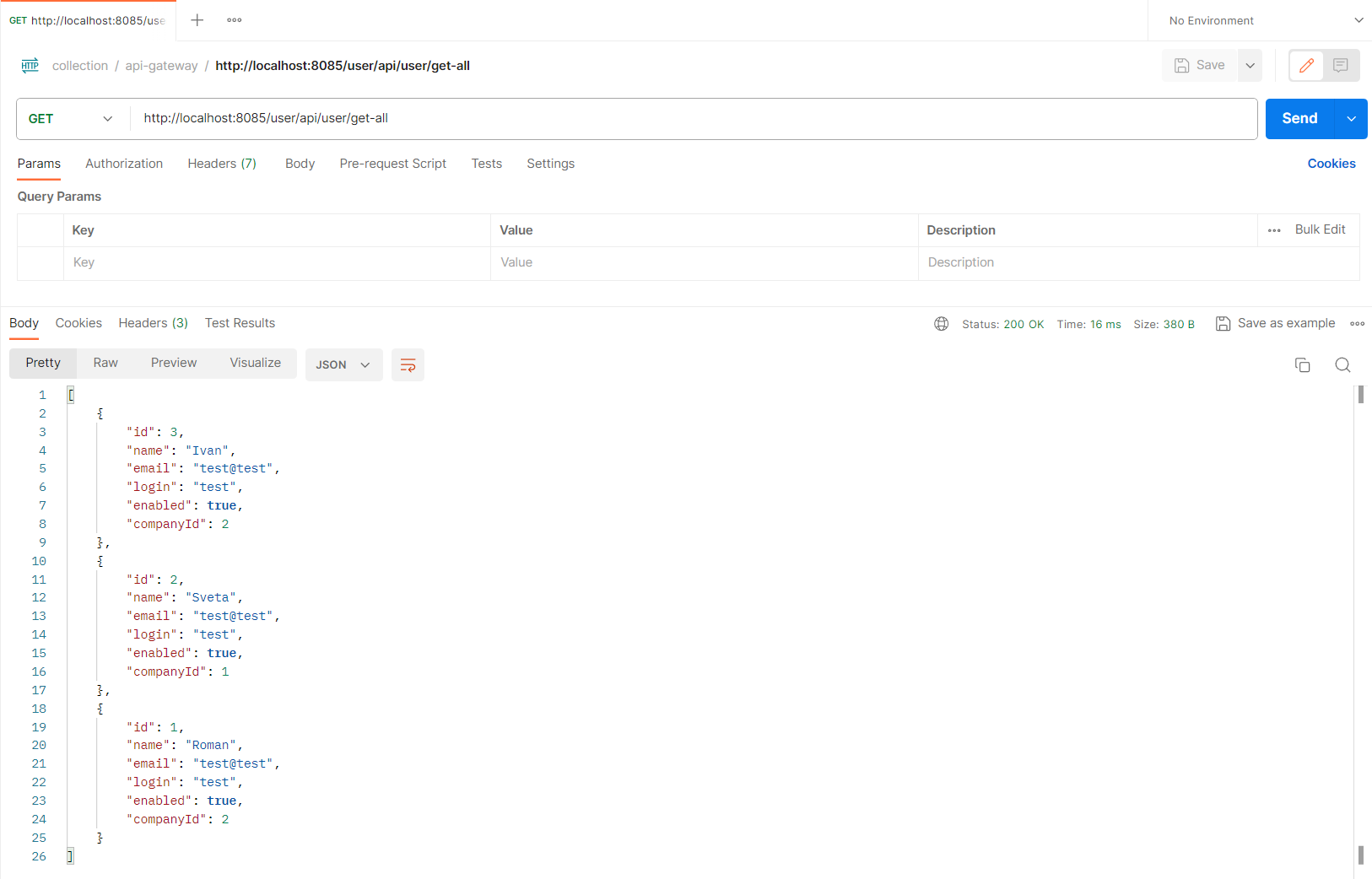
Результат работы Eureka server – вместе с нашими микросервисами:



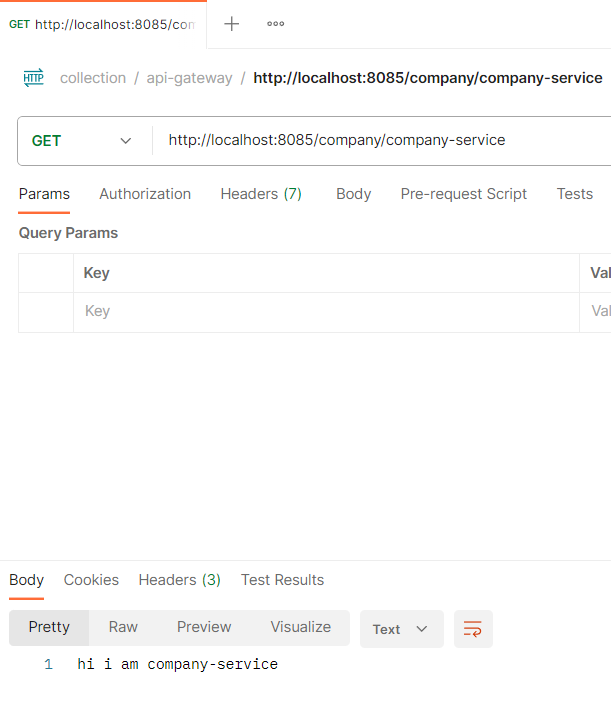
Конечные точки доступны через Postman:



Для проверки работоспособности user-service получим список всех пользователей (осуществим запрос через api gateway):



Для проверки работоспособности company-service вызовем нашу getTestProperty из api gateway:



Вызов сервисов осуществляется через api gateway. Запросы, которые идут по адресу …/user/… были перенаправлены в user-service, а все запросы по адресу …/company/… в company-service.

1. Вывод

В результате выполнения лабораторной работы была подключена служба обнаружения микросервисов и настроен api шлюз с необходимыми по заданию параметрами.

Написан отчёт.

[Исходный код доступен по ссылке](https://github.com/Allexeey/microservices/tree/lab-3)

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">

<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>company-service</description>

<artifactId>company-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>

</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>

package org.example.db.dto;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

public class CompanyDto {

private Long id;

private String name;

private String ogrn;

private String description;

private Long userId;

}

package org.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.openfeign.EnableFeignClients;

@EnableFeignClients

@SpringBootApplication

public class Main {

public static void main(String[] args) {

SpringApplication.run(Main.class, args);

}

}

package org.example.db.model;

import jakarta.persistence.\*;

import lombok.Getter;

import lombok.Setter;

@Getter

@Setter

@Entity

@Table(name = "COMPANY\_DS")

public class Company {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

private String ogrn;

private String description;

private Long userId;

}

package org.example.db.repo;

import org.example.db.model.Company;

import org.springframework.data.jpa.repository.JpaRepository;

public interface CompanyRepo extends JpaRepository<Company, Long> {

}

package org.example.config;

import org.modelmapper.ModelMapper;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@SuppressWarnings("Convert2Diamond")

@Configuration

public class MapperConfiguration {

@SuppressWarnings("Convert2Lambda")

@Bean

public ModelMapper modelMapper() {

ModelMapper modelMapper = new ModelMapper();

return modelMapper;

}

}

package org.example.api;

import org.example.api.service.CompanyService;

import org.example.db.dto.CompanyDto;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/company")

public class CompanyController {

@Autowired

private CompanyService companyService;

@PostMapping("/create")

public Long createCompany(@RequestBody CompanyDto companyDto) {

return companyService.createCompany(companyDto);

}

@GetMapping("/exists-by-id/{companyId}")

Boolean existsById (@PathVariable Long companyId) {

return companyService.existsById(companyId);

}

@GetMapping("/get-all")

List<CompanyDto> getAllCompanies () {

return companyService.getAllCompanies();

}

}

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("/ping")

public class ConfigController {

@Value("${description}")

private String description;

@GetMapping

public String getTestProperty() {

return description;

}

}

package org.example.api.service;

import jakarta.persistence.EntityNotFoundException;

import jakarta.transaction.Transactional;

import org.example.db.dto.CompanyDto;

import org.example.db.model.Company;

import org.example.db.repo.CompanyRepo;

import org.modelmapper.ModelMapper;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CompanyService {

@Autowired

private CompanyRepo repo;

@Autowired

private UserFeignClient client;

@Autowired

private ModelMapper mapper;

@Transactional

public Long createCompany(CompanyDto companyDto) {

Boolean exist = client.existsById(companyDto.getUserId());

if(!exist) {

repo.save(mapper.map(companyDto, Company.class));

throw new EntityNotFoundException("Компания создана. Директор с id = %s не существует".formatted(companyDto.getUserId()));

}

return repo.save(mapper.map(companyDto, Company.class)).getId();

}

@Transactional

public Boolean existsById(Long id) {

return repo.existsById(id);

}

@Transactional

public List<CompanyDto> getAllCompanies() {

return repo.findAll().stream().map(company -> mapper.map(company, CompanyDto.class)).toList();

}

}

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 = "user-service",

path = "/api/user",

url = "http://localhost:8089"

)

public interface UserFeignClient {

@GetMapping("/exists-by-id/{id}")

Boolean existsById (@PathVariable Long id);

}

url = jdbc:postgresql://localhost:5433/microservices?currentSchema=company-schema

username = user

password = password

referenceUrl=hibernate:spring:org.example.db.model\

?dialect=org.hibernate.dialect.PostgreSQLDialect\

&hibernate.physical\_naming\_strategy=org.hibernate.boot.model.naming.CamelCaseToUnderscoresNamingStrategy

changeLogFile=db/migration.sql

spring:

# profiles:

# active:

# test

application.name: company-service

cloud.config.uri: <http://localhost:8888>

-- liquibase formatted sql

-- changeset alexey:1698775667387-1

CREATE SEQUENCE IF NOT EXISTS company\_ds\_seq START WITH 1 INCREMENT BY 50;

-- changeset alexey:1698775667387-2

CREATE TABLE company\_ds (id BIGINT GENERATED BY DEFAULT AS IDENTITY NOT NULL,

description VARCHAR(255),

name VARCHAR(255),

ogrn VARCHAR(255),

user\_id BIGINT,

CONSTRAINT "company\_dsPK" PRIMARY KEY (id));