**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКО-ГЕРМАНСКИЙ ИНСТИТУТ ПРИКЛАДНОЙ ИНФОРМАТИКИ**

**CРСП**

**по дисциплине «Разработка объектно-ориентированных систем»**

**Проверил: ст. преп. Сарыпбекова Ж.**

**Выполнил: ст.гр. AIN-1-20 Куштарбеков К.**

**БИШКЕК 2022**

Цель работы: научиться разрабатывать микросервисные приложения с использованием фреймворка Spring Boot, Spring Cloud Netflix Eureka, Spring Cloud API Gateway , Axon-фреймворк.

Назначение: Приложение Online Food для бронирования разных блюд, десертов и т.д.

Данное приложение разрабатывается следующими микросервисами:

* food-online-service – микросервис по работе с едой
* order-service – микросервис по работе с заказом еды
* email-sender-service – микросервис по работе с почтами
* gateway – микросервис по работе маршутизации и безопасности(авторизация и аутентификация)

Запросы микросервиса food-online-service:

* <http://localhost:8090/api/food/create> – создает еду
* <http://localhost:8090/api/food/update> – редактирует еду
* [http://localhost:8090/api/food/delete/{id}](http://localhost:8090/api/food/delete/%7bid%7d) – удаляет еду по индексу
* [http://localhost:8090/api/food/get/{id}](http://localhost:8090/api/food/get/%7bid%7d) – возвращает еду по индексу, свернутую в другой объект
* <http://localhost:8090/api/food/get-all> – возвращает все блюда, десерты и т.д, свернутые в другие объекты
* [http://localhost:8090/api/food/get-{id}](http://localhost:8090/api/food/get-%7bid%7d) – возвращает еду по индексу

Запросы микросервиса email-sender-service:

* <http://localhost:8090/api/email/send> – отправляет письмо по почте

Запросы микросервиса gateway:

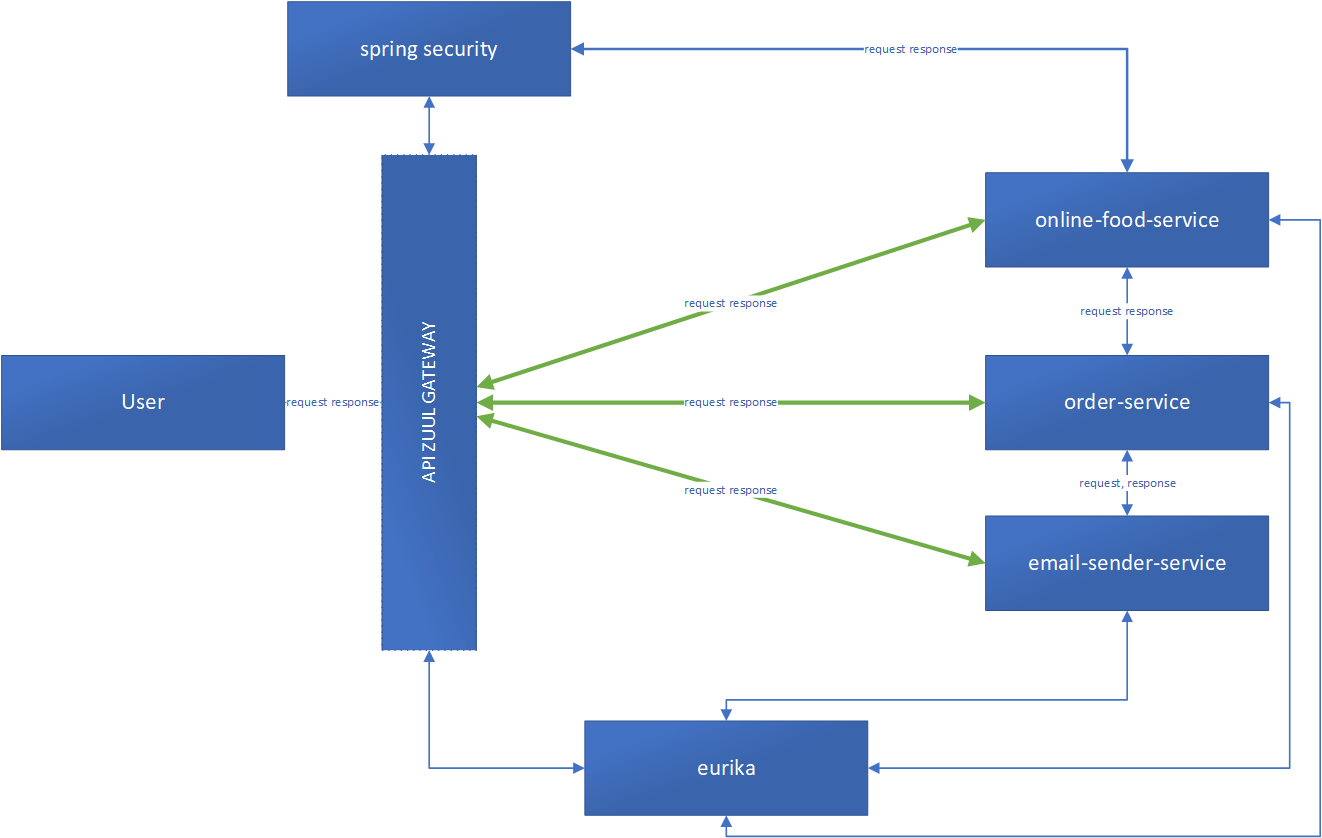
* <http://localhost:8090/auth/login> - аутентифицирует пользователя
* <http://localhost:8090/auth/register> – регистрирует пользователя
* <http://localhost:8090/auth/get-myself> – возвращает текущего аутентифицированного пользователя

Запросы микросервиса order-service:

* <http://localhost:8090/api/order/food> - заказывает еду
* [http://localhost:8090/api/order/complete/{id}](http://localhost:8090/api/order/complete/%7bid%7d) – принимает заказ
* [http://localhost:8090/api/order/cancel/{id}](http://localhost:8090/api/order/cancel/%7bid%7d) – отклоняет заказ
* [http://localhost:8090/api/order/get/{id}](http://localhost:8090/api/order/get/%7bid%7d) – возвращает заказ по индексу
* <http://localhost:8090/api/order/get-all> – возвращает все заказы
* <http://localhost:8090/api/order/get-all-waiting> - возвращает все заказы, которые в ожидании

Данный микросервис использует все другие микросервисы

Диаграмма микросервисов:



**Главный Pom.xml**

…

<modules>

<module>eureka-server</module>

<module>food-online-service</module>

<module>order-service</module>

<module>gateway</module>

<module>email-sender-service</module>

</modules>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>16</maven.compiler.source>

<maven.compiler.target>16</maven.compiler.target>

<spring.boot.maven.plugin.version>2.7.5</spring.boot.maven.plugin.version>

<spring.boot.dependencies.version>2.7.5</spring.boot.dependencies.version>

<spring.cloud-version>2021.0.2</spring.cloud-version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.dependencies.version}</version>

<scope>import</scope>

<type>pom</type>

</dependency>

<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>

<dependencies>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

</dependencies>

**Подключение Eureka Server.**

Для подключения Eureka Server создан Spring Boot проект на языке Java – eureka-server со следующими классами и конфигурационными файлами:

1. Pom.xml

…

<properties>

<maven.compiler.source>16</maven.compiler.source>

<maven.compiler.target>16</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

</dependencies>

2. application.properties, расположенный в src/main/resources, имеет следующие параметры.

spring:

application:

name: eureka-server

server:

port: 8761

eureka:

client:

fetch-registry: false

register-with-eureka: false

service-url:

defaultZone: <http://localhost:8761/eureka/>

3. В класс EurekaServerApplication добавлена аннотация @EnableEurekaServer

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

А в клиентских сервисах online-food-service, order-service, email-sender-service указываем аннотацию @EnableDiscoveryClient

**Netflix Zuul API Gateway**

Zuul — это основанный на JVM маршрутизатор и серверный балансировщик нагрузки от Netflix.

Используется с аннотацией @EnableZuulProxy.

Zuul автоматически выберет список серверов в Eureka.

Создан сервис gateway, который зарегистрирован в Eureka Server и имеет следующие свойства:

1. Pom.xml

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.3.1</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>com.auth0</groupId>

<artifactId>java-jwt</artifactId>

<version>4.0.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-zuul</artifactId>

<version>2.2.10.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

2. application.yml

server:

port: 8090

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka/

fetch-registry: true

register-with-eureka: true

logging:

level:

root: INFO

org.springframework.cloud.gateway.route.RouteDefinitionLocator: INFO

org.springframework.cloud.gateway: TRACE

spring:

application:

name: api-gateway

datasource:

url: jdbc:postgresql://localhost:5432/online\_food

username: postgres

password: postgres

driver-class-name: org.postgresql.Driver

jpa:

show-sql: true

hibernate:

ddl-auto: update

database: postgresql

database-platform: org.hibernate.dialect.PostgreSQL94Dialect

zuul:

routes:

food-service:

path: /api/food/\*\*

sensitiveHeaders: Cookie,Set-Cookie,Authorization

url: http://localhost:8085/api/food

order-service:

path: /api/order/\*\*

sensitiveHeaders: Cookie,Set-Cookie,Authorization

url: <http://localhost:8086/api/order>

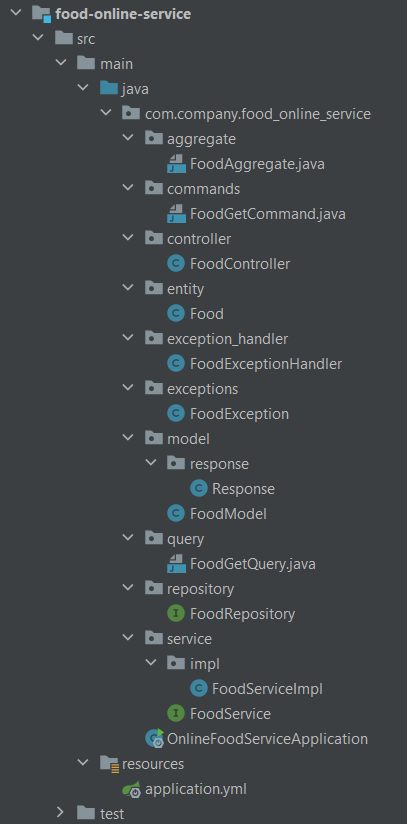
3. В данном сервисе также был добавлен Spring Security для безопасности приложения. Все конфигурационные классы находятся в пакете config

**Создание клиентских микросервисов**

Так как одной важных характеристик микросервисов является построение ее вокруг RESTful ресурсов и связь между сервисами может быть на основе событий или протокола HTTP, то для быстрого создания готовых RESTful приложений на языке Java был использован Spring Boot имеющий встроенные серверы (простота развертывания с помощью контейнеров) и внешнию конфигурацию

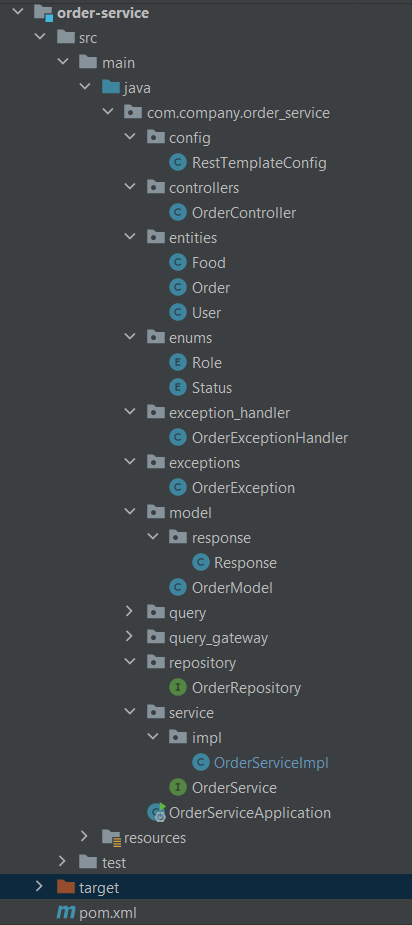
**online-food-service** – сервис еды, с помощью данного сервиса создается, редактируется, удаляется, возвращается еда.

структура проекта



**order-service** – сервис заказов

структура проекта:



email-sender-service – сервис почты

структура проекта:

