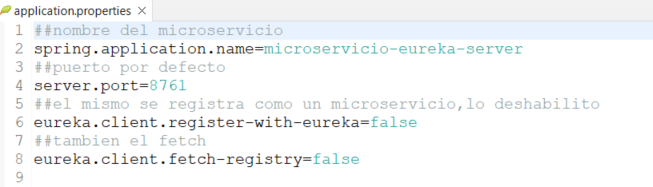
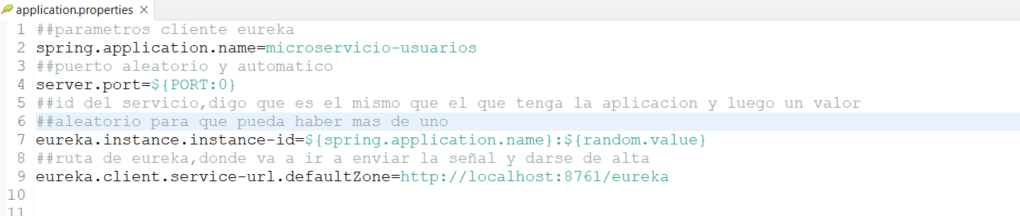
1→Creamos microservicio usuarios

2→Creamos el servidor eureka y los relacionamos en el properties de ambos

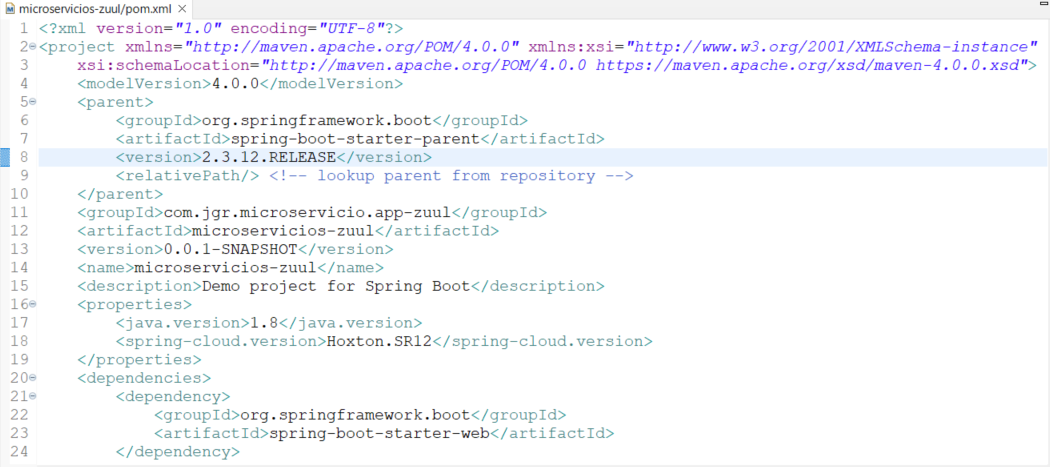
Properties de eureka:

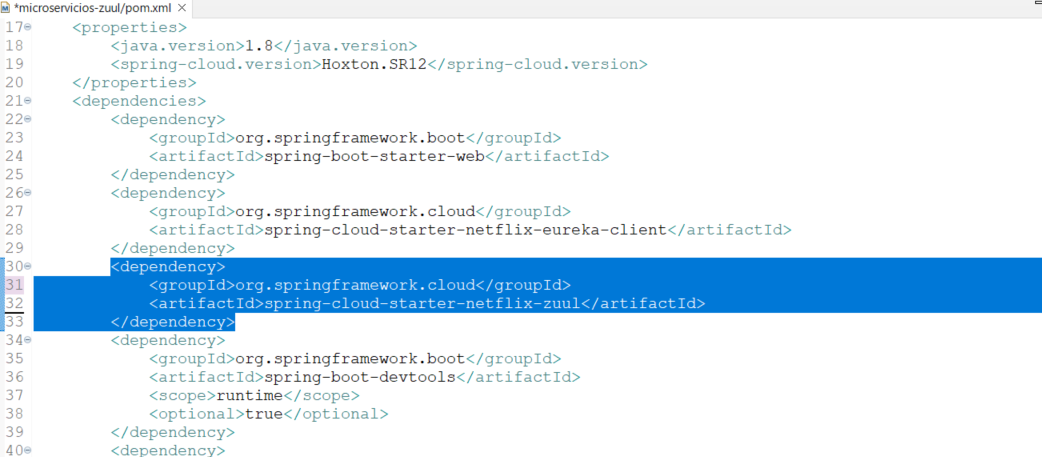


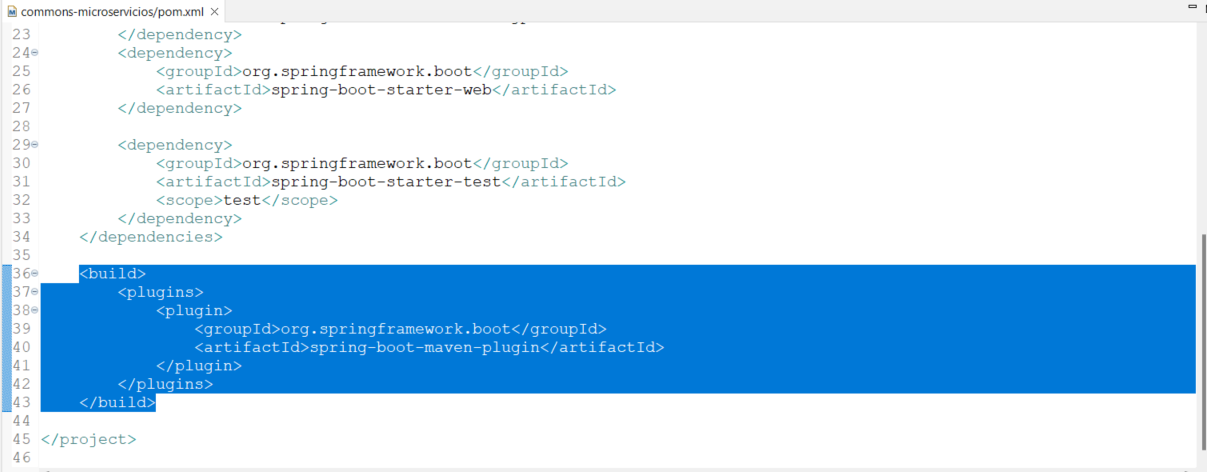
Properties de usuarios:



3→Creamos el gateway,todas las peticiones se dirigen a el que las redirige, con Zuul, aquí hay que tener cuidado con el pom,cambiamos la version de spring en el pom de zuul:

y luego añadimos en el mismo pom del zuul la dependencia:

4→Creamos un microservicio capa de SERVICIOcomun(el commons-microservicios) , le quitamos el starter en el pom del generico para que no se inicie:

esto lo hacemos para poder reutilizarlo en el resto de microservicios. Como no vamos a inyectarlo,sino a heredar de el le quitamos el @Service.

Lo hacemos lo mas generico posible,

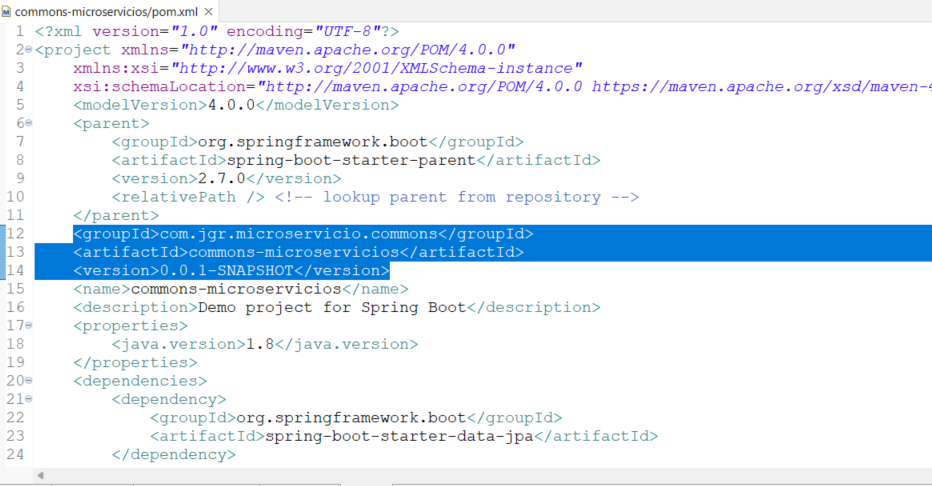
**public** **class** CommonServiceImpl<E,R **extends** CrudRepository<E,Long>> **implements** ICommonService<E>{

como parametros de entrada tiene E, que indica que es una entidad generica.

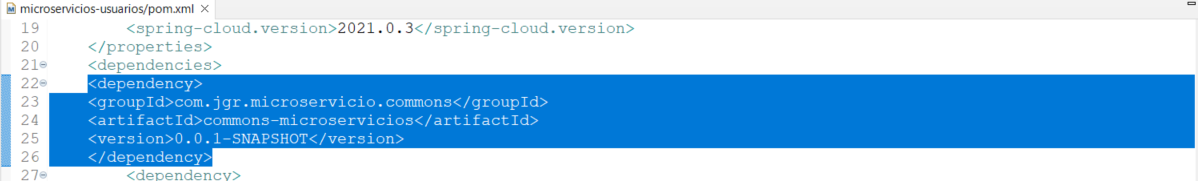
R es el nombre del repositorio que le vamos a pasar. Extiende de CrudRepository y como parametros le pasamos el nombre de la entidad “E” y el tipo de dato que tiene la clave, en este caso un Long. El ICommonService es el nombre de la interfaz que en este caso se llama CommonService,

5→modificamos el pom de microservicios-usuarios para que implemente el service de commons, en vez del suyo propio.

Esto es lo que copiamos del pom de commons:



Y lo incluimos como dependencia en el pom de usuarios:

Modificamos el IAlumnoService para que extienda de ICommonService

5-Creamos un CONTROLADOR generico en common-microservicios que va a tener los métodos comunes a todos los servicios como es listar,ver por id,alta y borrar. Actualizar no,porque pueden ser campos distintos dependiendo de la entidad

No ponemos RestController porque se va a heredar.

//le entra una entidad E y un servicio S, este ultimo hereda de ICommonService,que como es generico tambien recibe una entidad como parametro de entrada

**public** **class** CommonController<E,S **extends** ICommonService<E>> {

//IMPORTANTE QUE SEA PROTECTED,NO PRIVATE PARA QUE PODAMOS USARLO EN LOS QUE HEREDAN ESTA CLASE GENERICA

@Autowired

**protected** S service;

En el controlador de Alumno indicamos que hereda de CommonController y le pasamos la entidad,que es alumno y el servicio que es IAlumnoService, que a su vez hereda de CommonService.

Como repositorio vamos a utilizar el que tiene CommonController, que se llama **protected** S service

**public** **class** AlumnoController **extends** CommonController<Alumno,IalumnoService>{

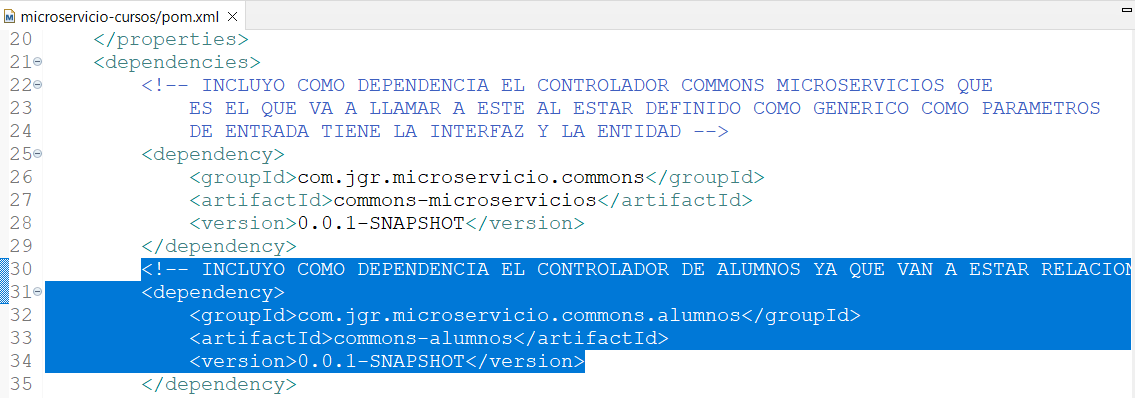
6-Creamos microservicio cursos con discovery y el resto de dependencias que tiene usuarios,copiamos la dependencia en el pom de commons-microservicios, configuramos el properties como el que tiene usuarios,cambiamos el nombre del microservicio a cursos,el resto es igual,misma bbdd,conexion a eureka…

El servicio y el controlador quedan igual que en alumnos.

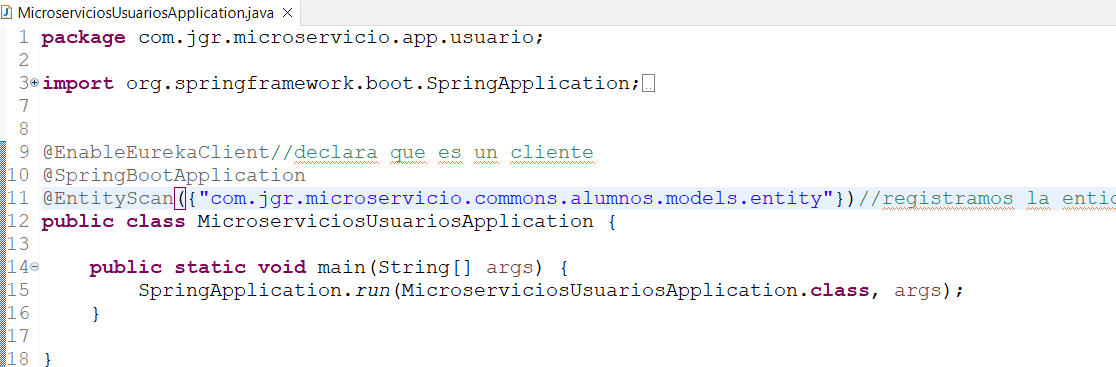
Definimos en zuul el microservicio igual que hemos hecho con alumnos

7-Para la relacion entre alumno y el resto de servicios creamos un proyecto commons-alumnos que contiene la definicion de la entidad, solo hace falta que tenga jpa en el pom, el resto incluido el spring-boot-maven no hace falta.

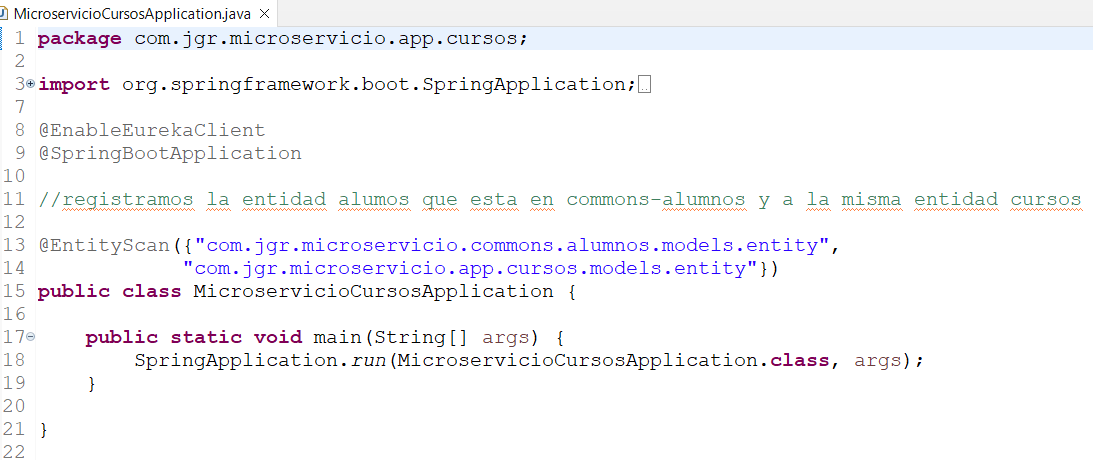
8-Añadimos como librería en el pom de usuarios y cursos el commons alumnos:



Como hemos sacado fuera del microservicio la entidad, la inportamos en el microservicio-usuarios:



Lo mismo hacemos en cursos,relacionamos las dos:

Incluimos cursos aunque sea una entidad de este microservicio porque al incluir alumnos lo sobreescribimos

9-Relacionamos las entidades alumno y curso. En curso creando un array de alumnos, no la hace bidireccional porque asi esta mas desacoplada¿?. Tenemos que tener cuidado con el metodo delete , sobreescribimos el metodo equals en alumno para que compare por id.

10-Implementa en curso controller el metodo añadir lista de alumnos y borrar alumno por id