SECUENCIA DE LLAMADAS CUANDO ESTA TODO CREADO

1->Zuul que es el gateway recibe las peticiones en el puerto 8090 que tiene definido en el properties se conecta con eureka en el 8761,tambien definidos en el properties los microservicios que va a atender

2→Eureka,invocado por Zuul lo que hace es dirigir la llamada a los microservicios, que pueden tener mas de una instancia cada uno

3→Microservicios ,cada uno de ellos tiene definido su nombre y se relaciona con eureka en su propio properties

ORDEN PARA LEVANTAR SERVICIOS

1-Eureka, para que puedan registrarse los servicios

2-microservicios

3-Zuul ,es el punto de entrada

RESUMEN

Crea microservicios de manera individual con las entidades que están mas relacionadas. Luego se crea paquetes commons-\* a las que saca las entidades, que luego relaciona incluyéndolo en el pom del microservicio ,estas no se van a ejecutar son solo librerías,

Se crea un commons-microservicios que lo que tiene por una parte es lo común a todos los controladores :buscar por id,listar todo,crear y eliminar. El update no,porque cada entidad tiene sus propiedades que son distintas de las del resto. En este mismo microservicio se crea la capa de servicio, tanto interfaz como imp que hereda del crud que es tambien comun a todos :listar,buscar por id,guardar y borrar por id que también es igual para todas las entidades.

Luego relaciona los microservicios con el commons que tiene el controlador con la parte común heredando del commons y tambien la parte del servicio. En caso de que tenga entidades relacionadas hereda tambien del commons que corresponda.

En el properties de cada microservicio se crea una relacion con eureka indicando el nombre que tiene y el puerto(se puede poner 0 para que sea automatico) y la ip del servidor eureka.

Eureka lo que hace es publicar los distintos microservicios relacionados y puede contener mas de una instancia del mismo microservicio en ejecución.

Para acceder a la aplicación lo que se hace es utilizar Zuul,que a su vez esta relacionado con eureka,

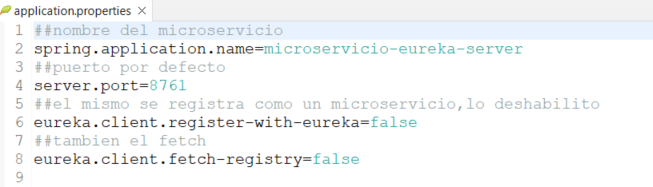
y con los microservicios, se pone su nombre y su ruta de acceso.

Cuando se entra a la aplicación se hace a traves de zuul y es eureka el que gestiona las instancias de cada uno de los servicios y lo redirecciona ahi.

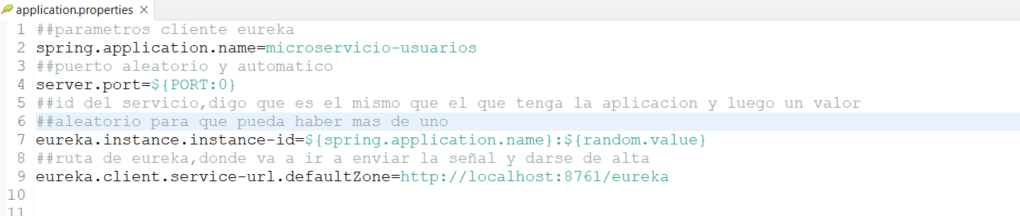
1→Creamos microservicio usuarios

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

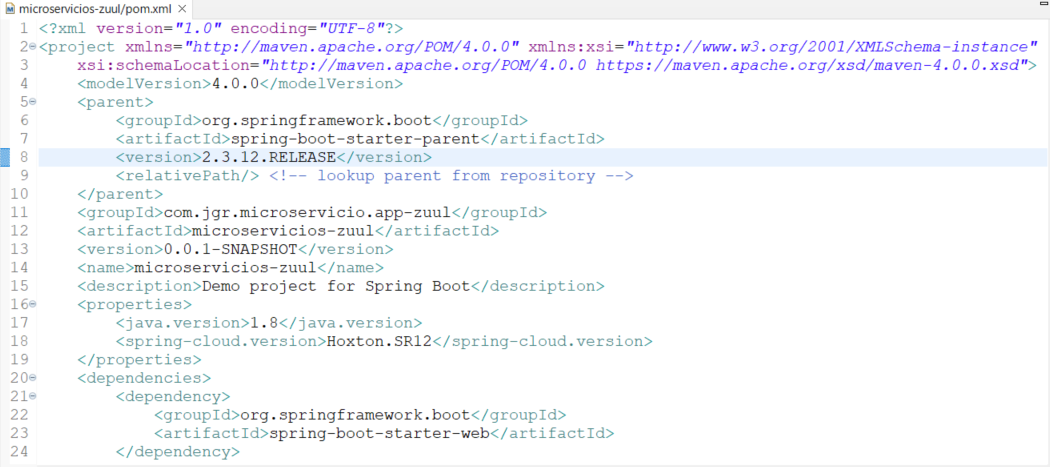
Properties de eureka:

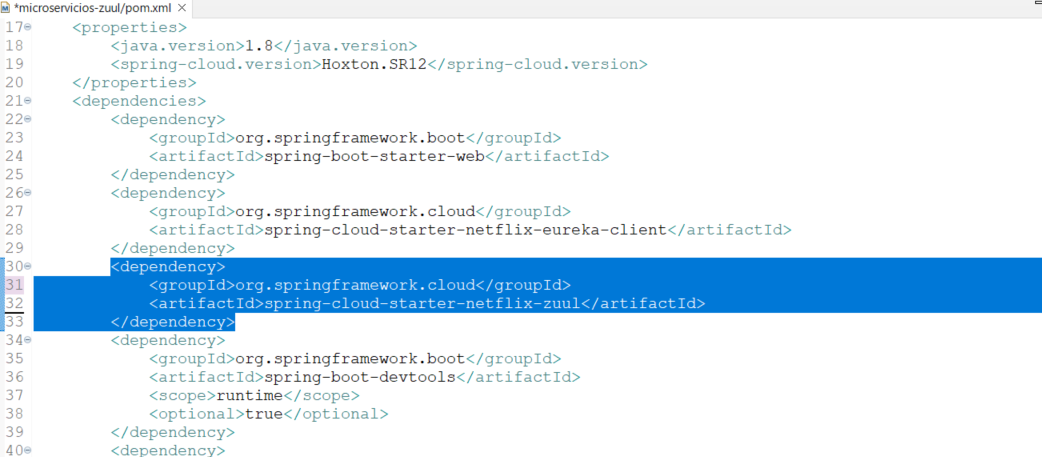


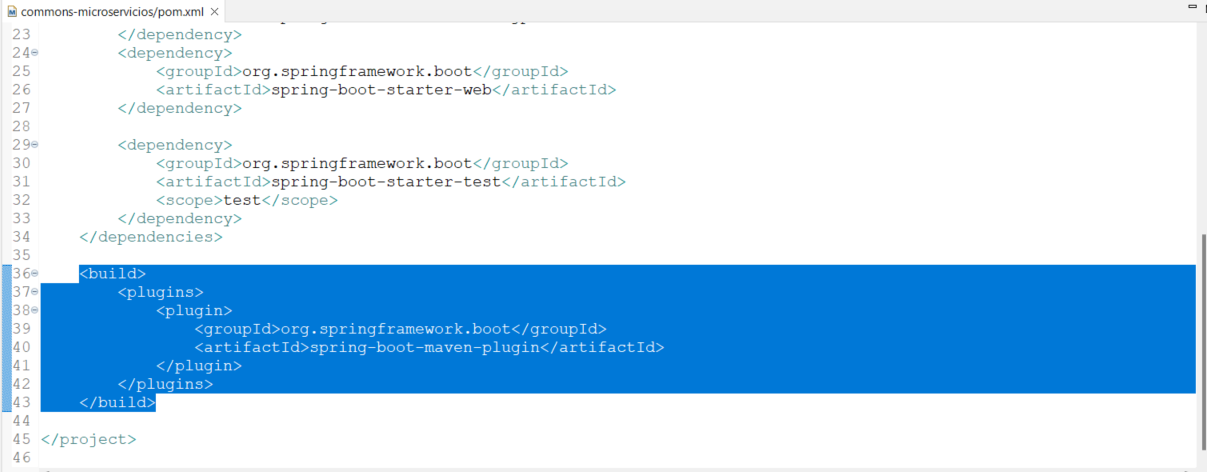
Properties de usuarios:



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,

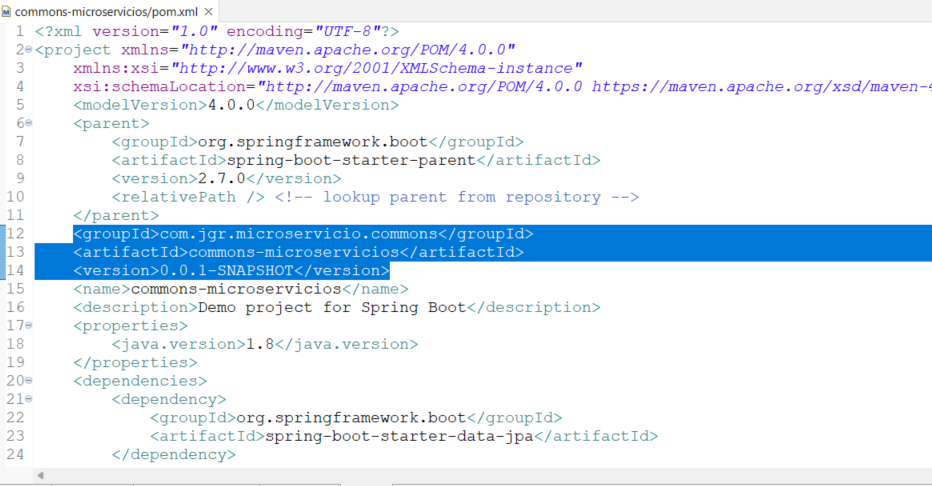
**public** **class** CommonServiceImpl<E,R **extends** CrudRepository<E,Long>> **implements** ICommonService<E>{

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

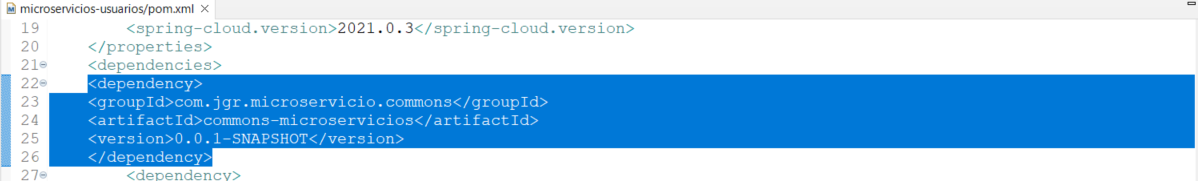
R es el nombre del repositorio que le vamos a pasar. Extiende de CrudRepository y como parametros le pasamos el nombre de la entidad “E” y el tipo de dato que tiene la clave, en este caso un Long. El ICommonService es el nombre de la interfaz que en este caso se llama CommonService,

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>{

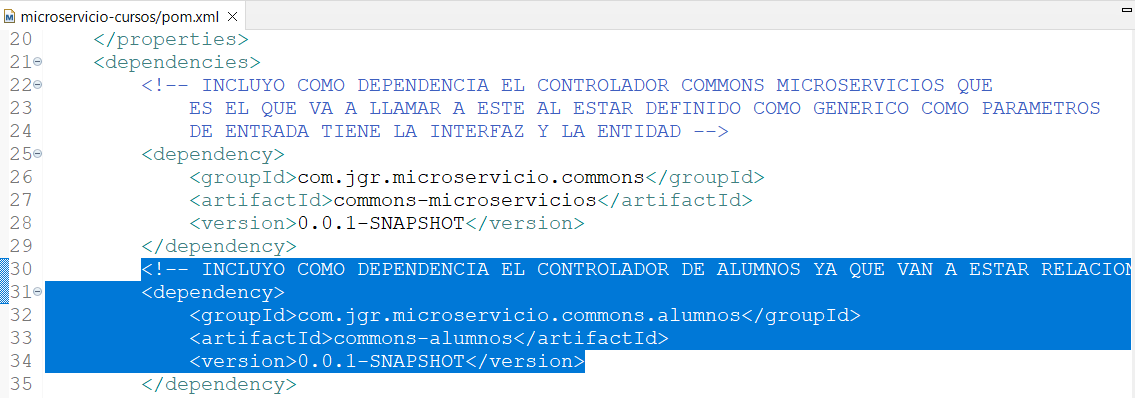
6-Creamos microservicio cursos con discovery y el resto de dependencias que tiene usuarios,copiamos la dependencia en el pom de commons-microservicios, configuramos el properties como el que tiene usuarios,cambiamos el nombre del microservicio a cursos,el resto es igual,misma bbdd,conexion a eureka…

El servicio y el controlador quedan igual que en alumnos.

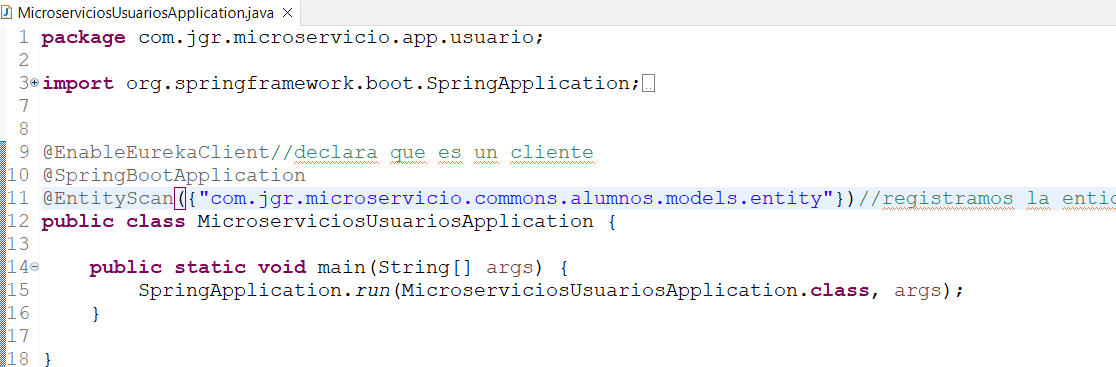
Definimos en zuul el microservicio igual que hemos hecho con alumnos

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.

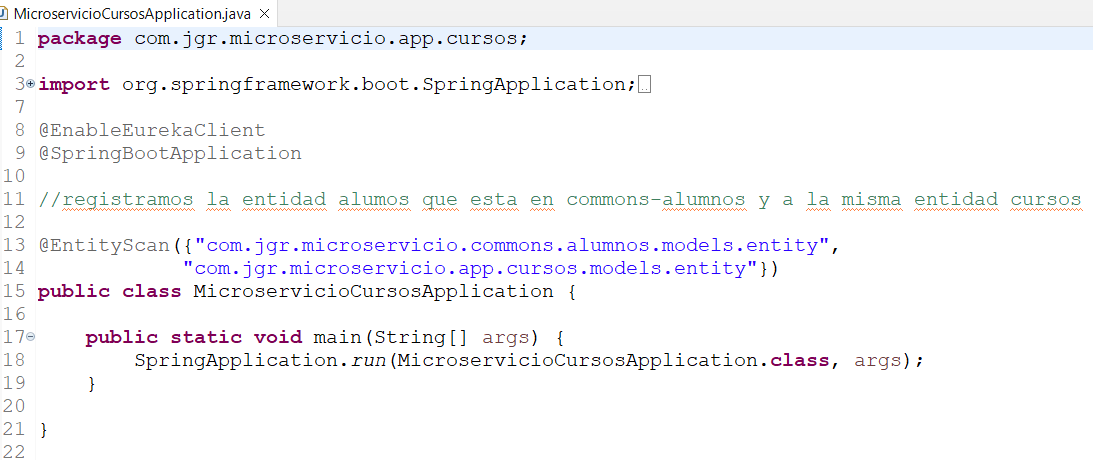
8-Añadimos como librería en el pom de usuarios y cursos el commons alumnos:



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



Lo mismo hacemos en cursos,relacionamos las dos:

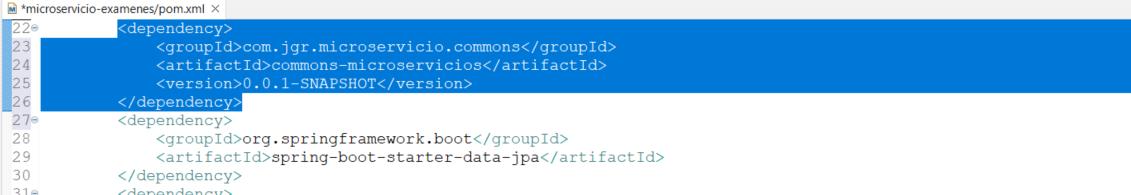
Incluimos cursos aunque sea una entidad de este microservicio porque al incluir alumnos lo sobreescribimos

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

11-Crea microservicio examenes que tiene examen y pregunta relacionada 1-n .

Incluimos la dependencia del commonservice en el pom,como hemos hecho en usuarios y en cursos

Al igual que hemos hecho con el resto, creamos el repositorio que extiende del Crud,la capa de servicios, la interfaz extiende de la de commonservice public interface IExamenService extends IcommonService<Examen> y la implementacion public class ExamenServiceImpl extends CommonServiceImpl<Examen,IExamenRepository> implements IexamenService

Luego creamos el controlador que extiende de commoncontroller y como parametros tendra la entidad, que es examen y la interfaz de la capa de servicio

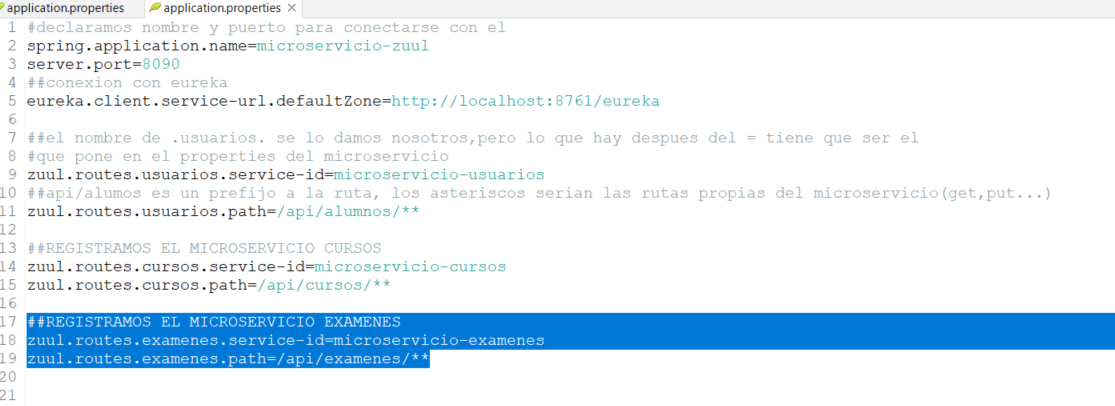
public class ExamenController extends CommonController<Examen,IexamenService>

Al hacerlo asi tendremos acceso al repositorio generico que hemos creado en el commoncontroller, la variable se llama service: protected S service;

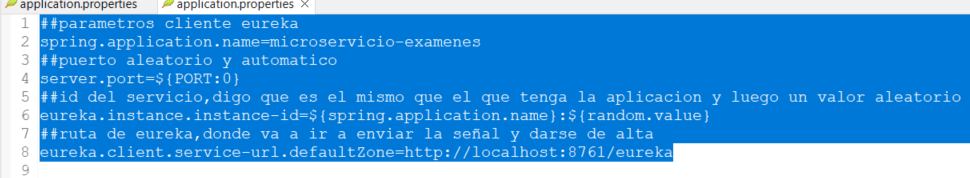
Tambien tendremos acceso a los metodos buscarporid,crear,listar y eliminar que tenemos definido dentro de el. Como en los otros microservicios declaramos el actualizar porque es el unico que tiene campos especificos, el resto de metodos se gestionan como objetos y da igual las propiedades que tengan.

En el controlador del microservicio implementamos el actualizar, como tenemos un array de preguntas dentro del examen, tendremos que actualizarlo con las nuevas y borrar las viejas.

Ademas de esto tenemos que darlo de alta en Zuul,como hemos hecho con los otros



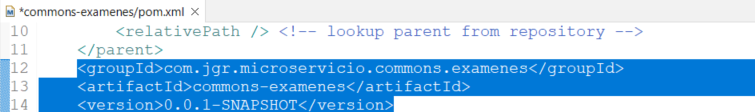
En eureka no hace falta darlo de alta,porque ya hemos creado la relación en el properties de exámenes:

12-Creamos el microservicios commons-examenes que solo tenga jpa,va a ser el servicio comun. Quitamos del pom la dependencia org.springframework.boot, tambien quitamos la clase principal porque va a ser una librería,no se va a ejecutar.

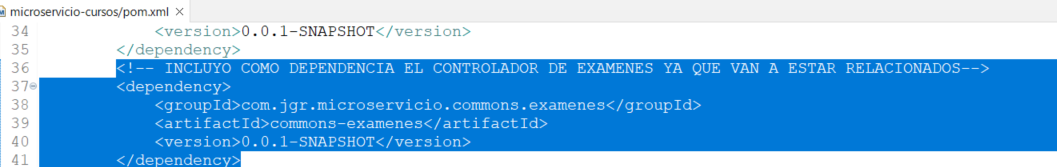
Quitamos las entidades del microservicio examenes y el package entities que va a quedar vacia.

En el pom de commons-examenes incluimos la dependencia spring-boot-starter-json para que podamos incluir las referencias que tenemos en las entidades

Relacionamos ambos microservicios tomando el groupid,artifact y la version de commons-examenes incluyendoolo en examenes:

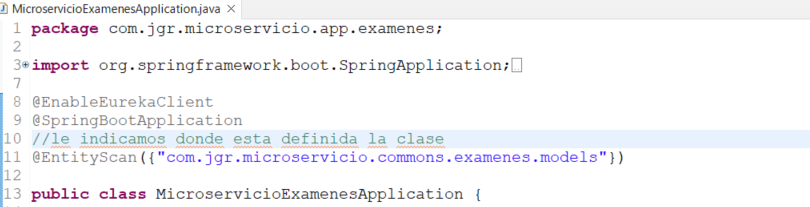
De esa manera en exámenes podemos usar los repositorios,

13-Ahora relacionamos con cursos copiando los datos commons-examenes en microservicio-cursos:

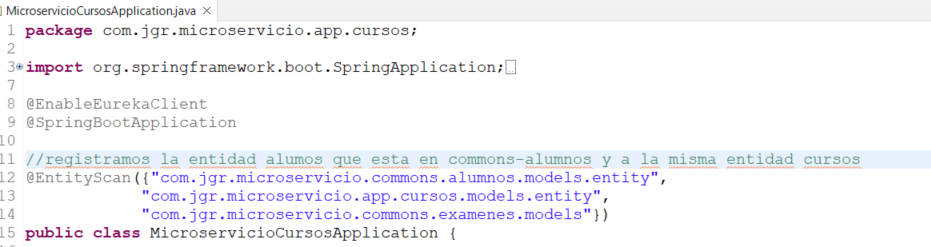
 Esto lo hacemos porque dentro de cursos vamos a crear una relacion manytomany de examenes, de esa manera se reconocen entre si.

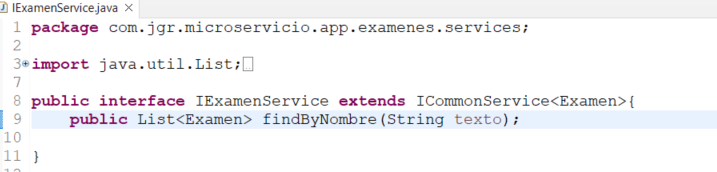
Añadimos los metodos get/set tanto del array como del objeto individual y en el controlador principal CursoController definimos los endpoint correspondientes a examen.

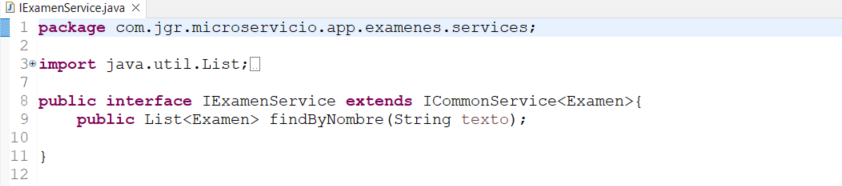
Tambien tenemos que incluir el @EntityScan para que reconozca la clase porque examen y pregunta estan en un proyecto distinto

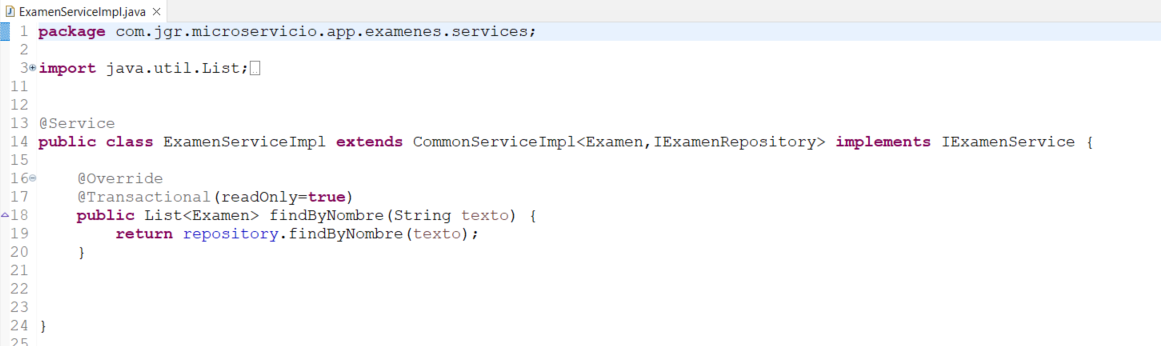


Y en la clase principal de la aplicación

Definimos busquedas propias en la interfaz IexamenRepository donde pintamos la query que luego copiamos tambien en la interfaz del servicio e implementamos







Como ya lo tenemos declarado, en su controlador propio lo implementamos:

