UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Objetivos de la sesión:

- *Recordar como **crear una base de datos MySql desde phpMyAdmin de Xampp** y ejecutar un script de creación de tablas e inserción de los datos iniciales.*
- *Aprender a **crear, configurar, ejecutar y depurar proyectos Spring Boot sobre la base de datos MySql**.*
- *Crear **entidades JPA** asociadas a una tabla que tengan **relaciones con otras tablas** de la Base de Datos con la notación **@ManyToOne***

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

De manera resumida, esta práctica vamos a dividirla en 3 grandes apartados:

1º Crear la BD futbol con todas sus tablas en MySQL utilizando Xampp:

- **1.1º Acceder a phpMyAdmin**
- **1.2º Crear un usuario nuevo y la Base de Datos de Futbol**
- **1.3º Ejecutar Script de creación de tablas e inserción de datos**
- **1.4º Comprobaciones**
- **1.5º Entender la Base de Datos de Futbol y sus relaciones**

2º Crear API RESTful con Spring Boot:

- **2.1º Crear el proyecto Spring Boot**
- **2.2º Configurar la Base de Datos**
- **2.3º Crear las entidades JPA**
- **2.4º Crear el repositorio para cada entidad JPA**
- **2.5º Crear la capa de servicios**
- **2.6º Crear la capa de control (controller)**
- **2.7º Pruebas**

3º Tablas relacionadas: @ManyToOne

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

1º Acceder a phpMyAdmin:

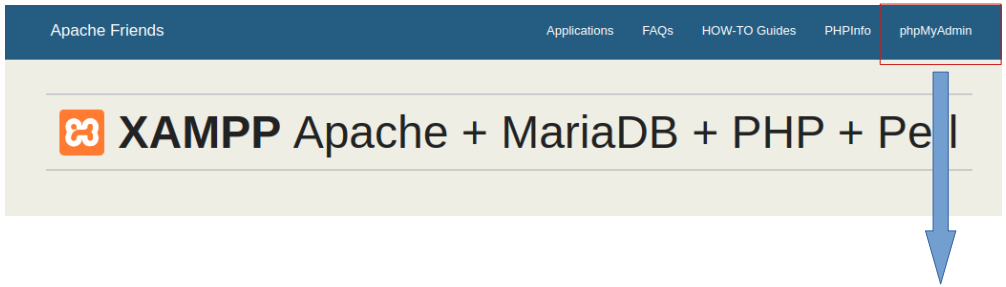
Presuponiendo que tenemos ***XAMPP instalado*** comprobamos que la *BD esta arrancada* .

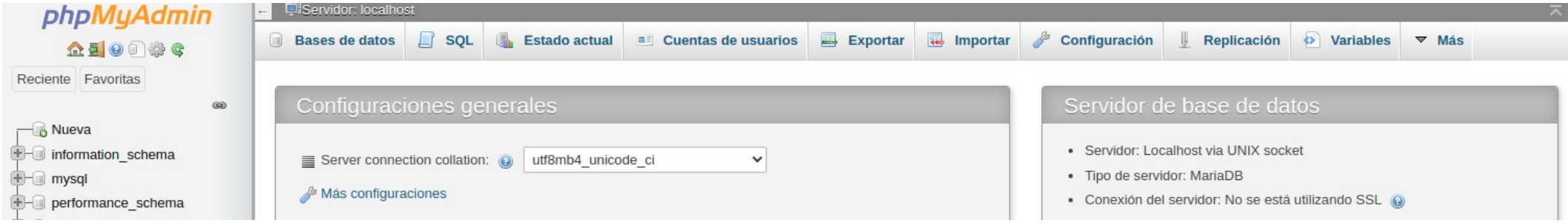
Lo podemos hacer por comandos: /opt/lampp/bin/mysql

O también accediendo al entorno gráfico de phpMyAdmin en : http://localhost

Si no lo esta la arrancamos manualmente: sudo /opt/lampp/lampp start

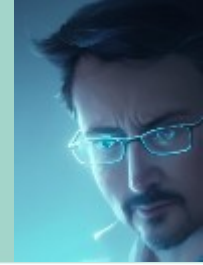
Ya podemos acceder a ***phpMyAdmin***:





UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

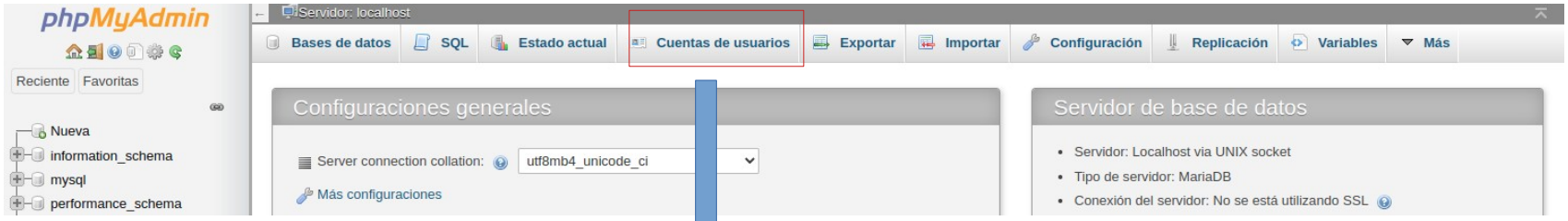
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

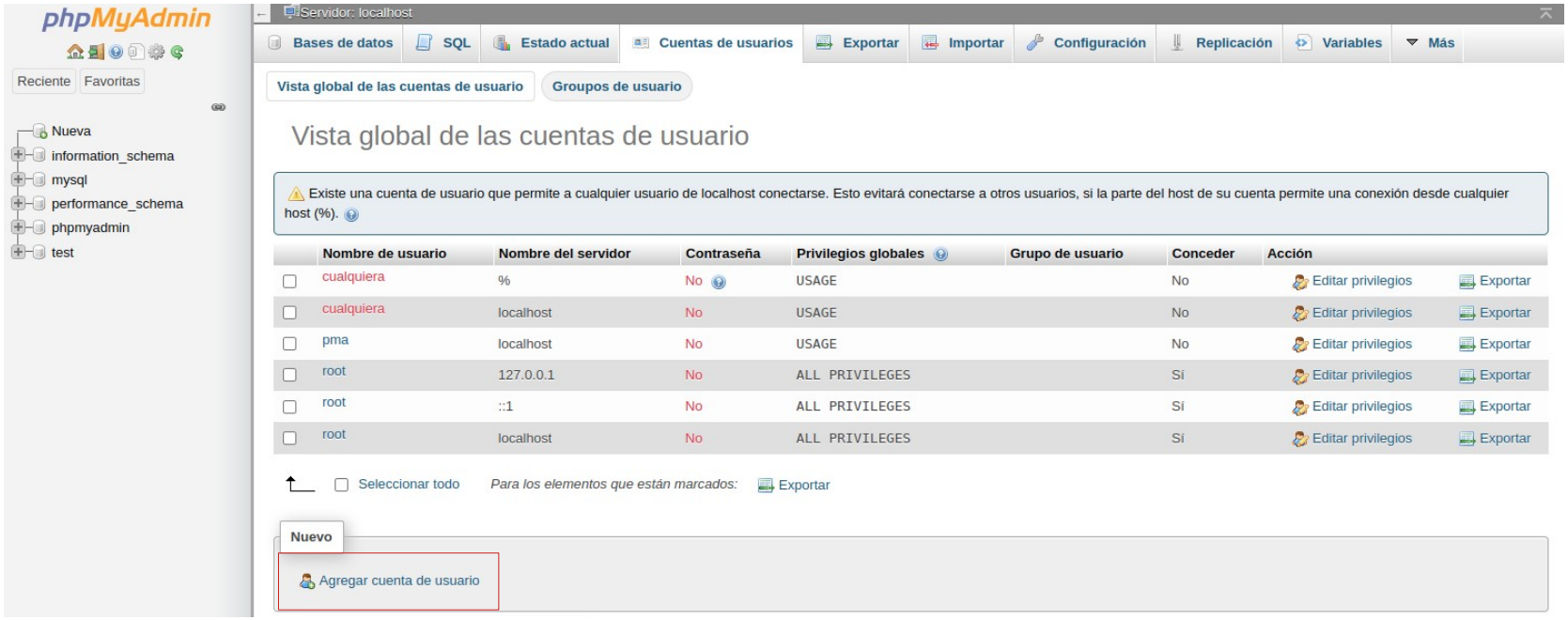


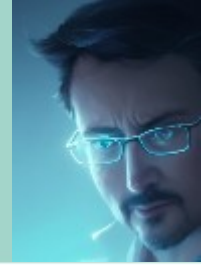
Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

2º Crear un usuario nuevo y la base de datos de futbol:

Entramos en “Cuentas de usuarios” y seleccionamos “Agregar cuenta de usuario”:



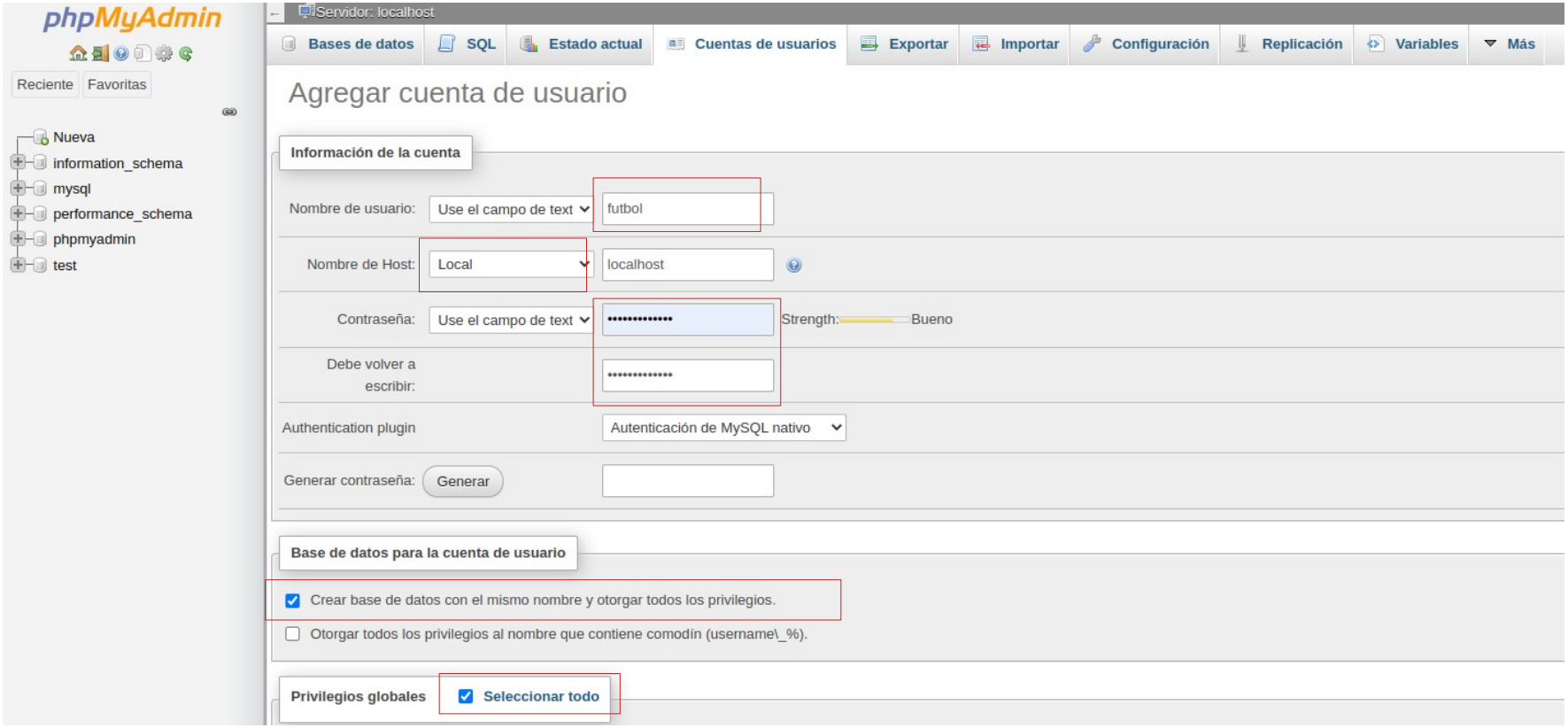


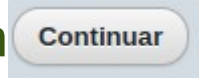


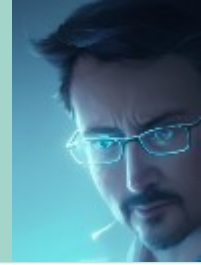
Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

… **Continuación 2º Crear un usuario nuevo y la base de datos de futbol:**

Creamos el ***usuario 'futbol'***, para el servidor local (nombre de host), asignamos la contraseña "futbolSimarro" y seleccionamos "Crear base de datos con el mismo nombre" asignándole todos los permisos "Privilegios globales: Seleccionar todo":



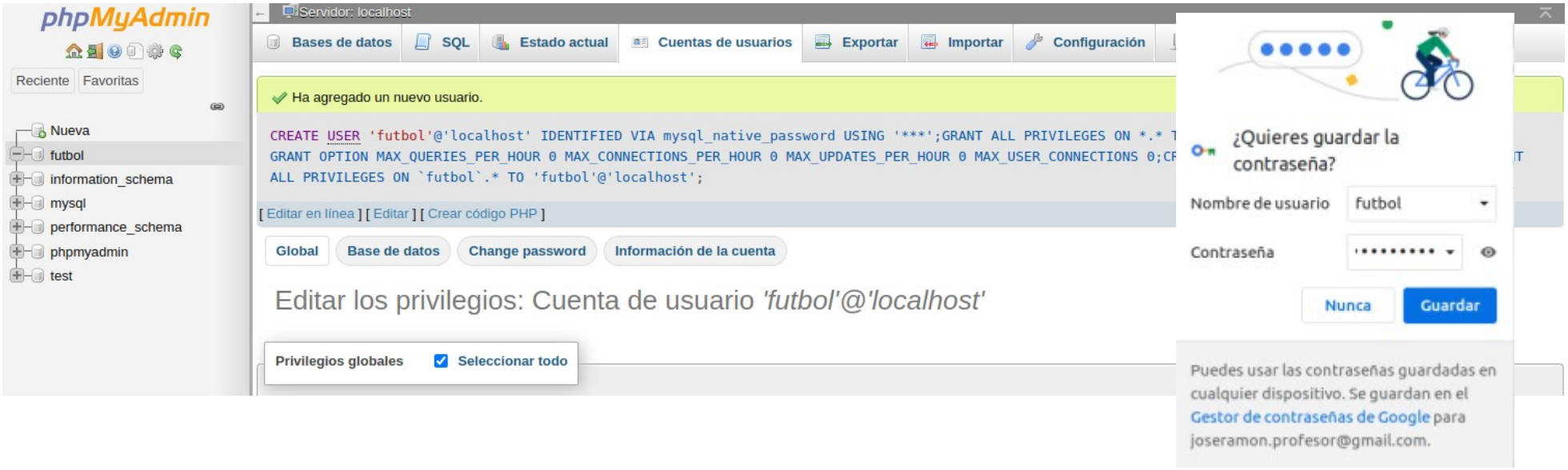
Si nos vamos al final de la página ya podemos pulsar en 



Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

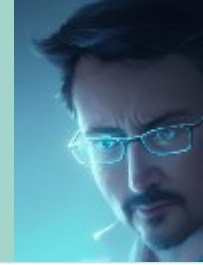
… **Continuación 2º Crear un usuario nuevo y la base de datos de futbol:**

Si todo ha ido bien ya podemos ver nuestro usuario creado y no estaría mal guardar la constraseña para no tener que volverla a teclear:



Si quisieras hacerlo por consola seria equivalente a:

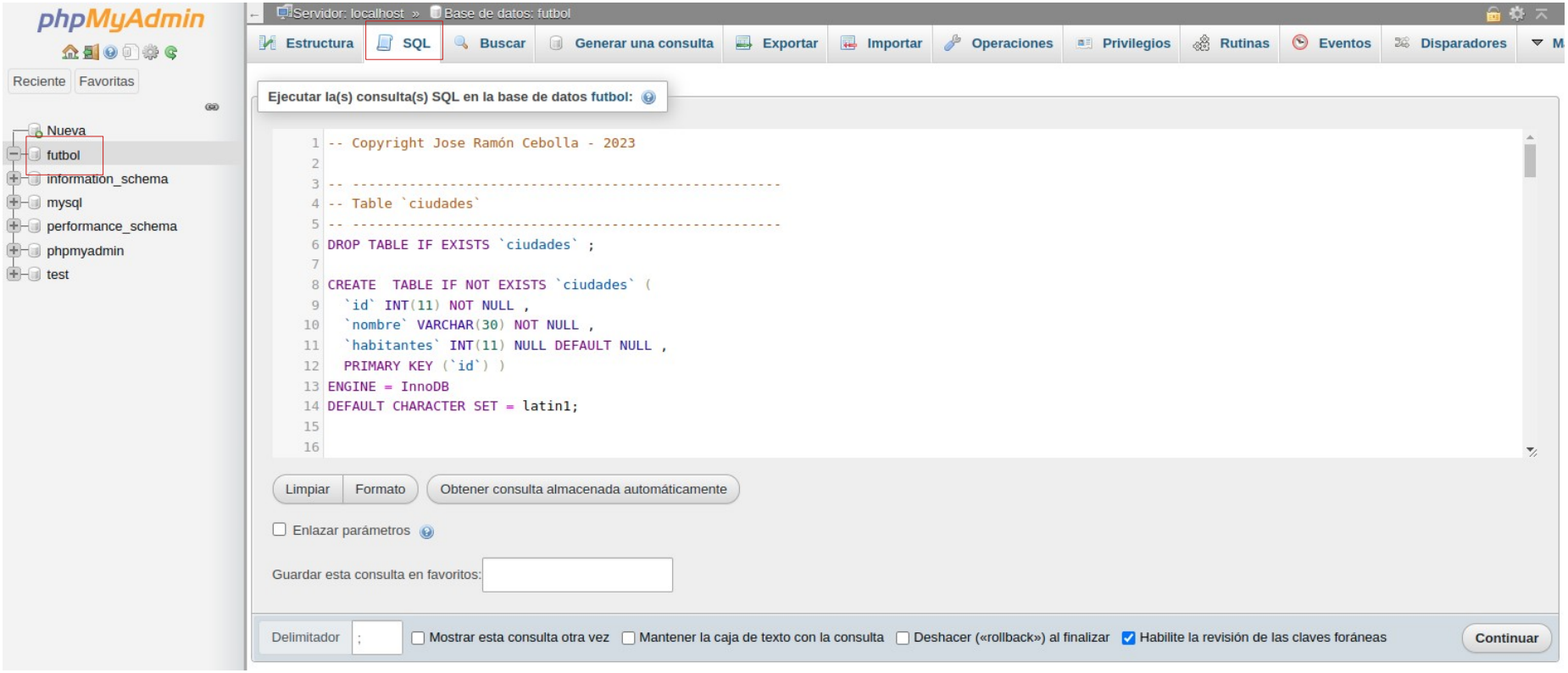
```
CREATE USER 'futbol'@'localhost' IDENTIFIED VIA mysql_native_password USING '***';GRANT ALL PRIVILEGES ON *.* TO 'futbol'@'localhost' REQUIRE NONE WITH GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;CREATE DATABASE IF NOT EXISTS `futbol`;GRANT ALL PRIVILEGES ON `futbol`.* TO 'futbol'@'localhost';
```



Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

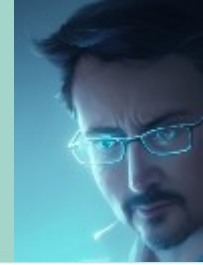
3º Ejecutar script de creación de tablas e inserciones de datos:

Entrando en el ***usuario "futbol"*** , ya podemos ejecutar nuestro script de creación de las tablas y la inserción de los datos iniciales copiando el sql del DRIVE en la ***subpestaña "SQL"*** y pulsando "Continuar":



UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

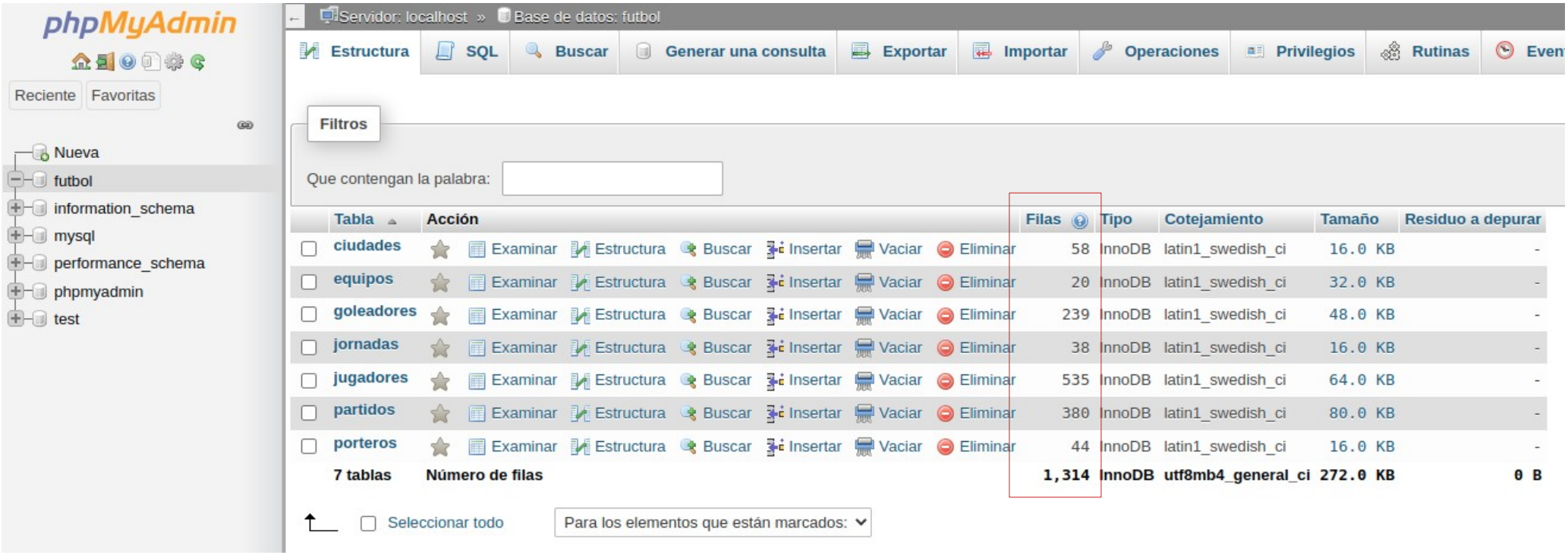
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

4º Comprobaciones:

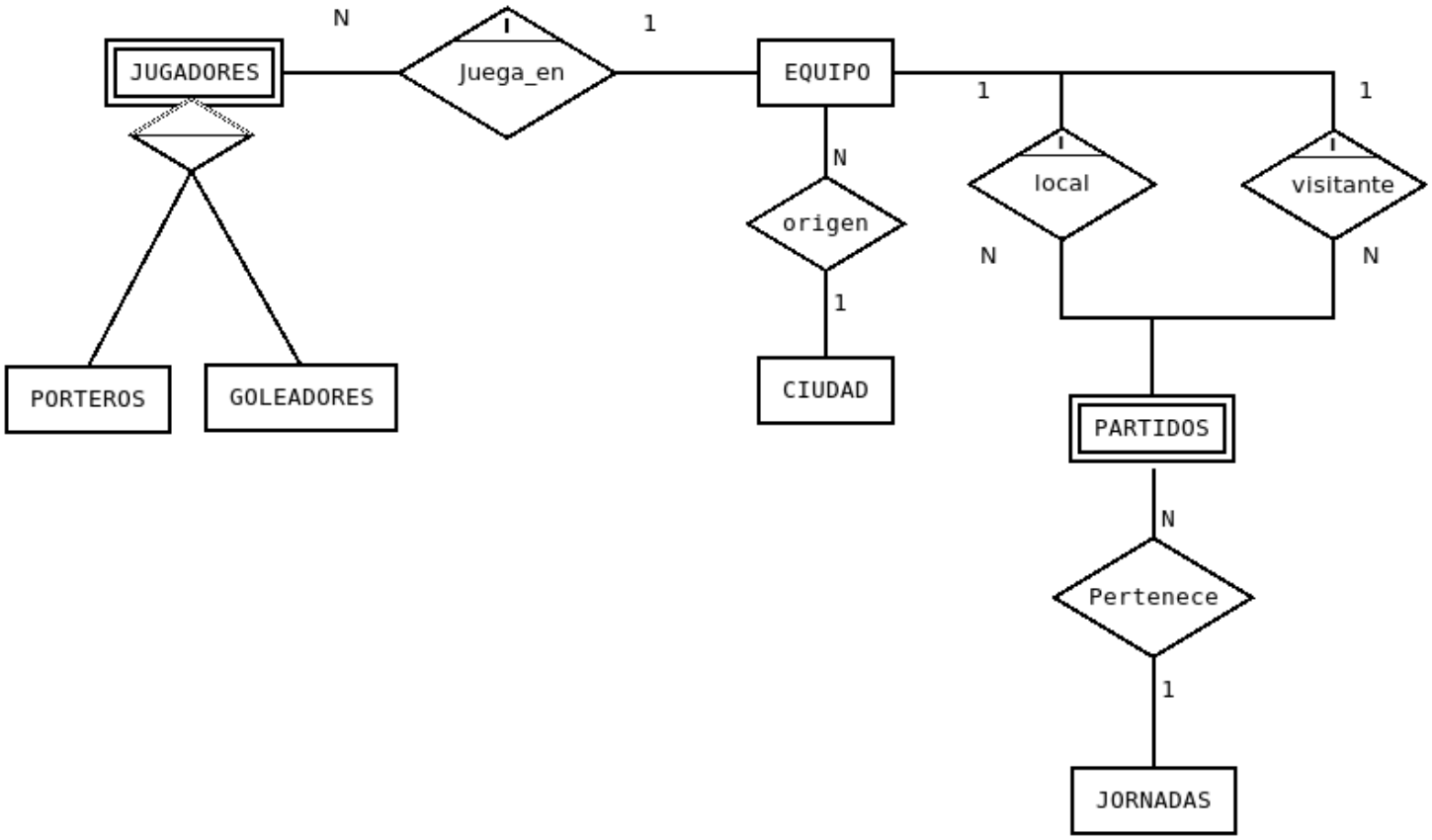
Entrando en el ***usuario "futbol"*** , deberíamos de poder ver las siguientes tablas y comprobar que tienen datos comprobando las filas:



Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

5º Entender la base de datos de futbol y sus relaciones:

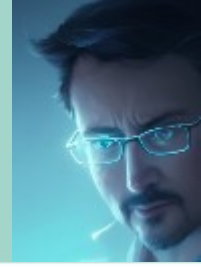
La visión global de la Bd de futbol que vamos a crear es:



¿Te acuerdas de como interpretar el Diagrama E-R?

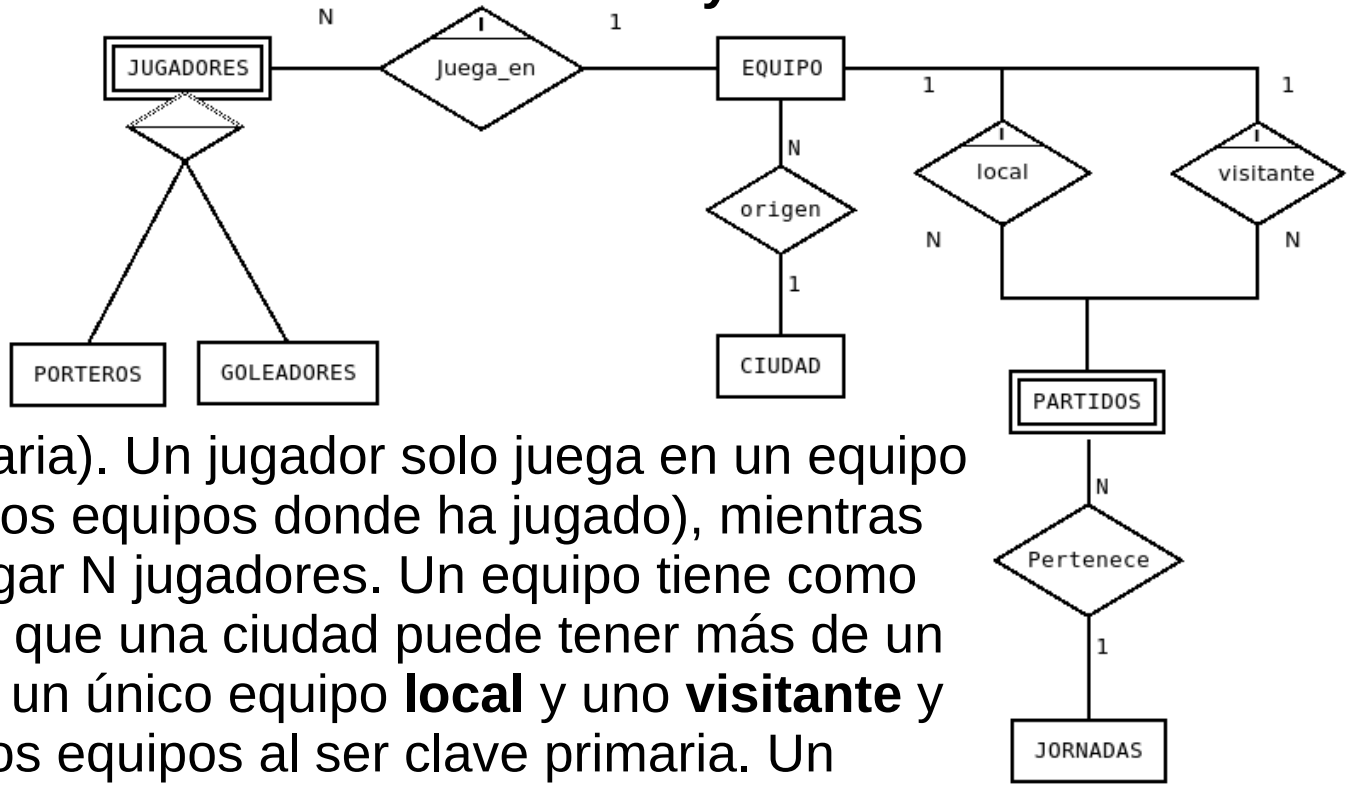
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

... **Continuación 5º Entender la base de datos de futbol y sus relaciones:**



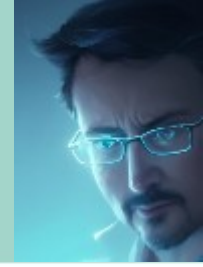
Un **jugador** puede especializarse en **portero** o **goleador**.

Para identificar a un jugador se necesita a su **equipo** (porque utilizaremos el dorsal y el equipo como clave primaria). Un jugador solo juega en un equipo (no guardamos historico de los equipos donde ha jugado), mientras que en un equipo pueden jugar N jugadores. Un equipo tiene como origen una **ciudad**, mientras que una ciudad puede tener más de un equipo. En un **partido** juega un único equipo **local** y uno **visitante** y es necesario especificar estos equipos al ser clave primaria. Un equipo puede jugar en más de un partido como local o como visitante. Por último, un partido se realiza en una **jornada**, mientras que en una jornada tiene lugar N partidos.

¿Te acuerdas de como traducirlo a tablas de la BD?

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

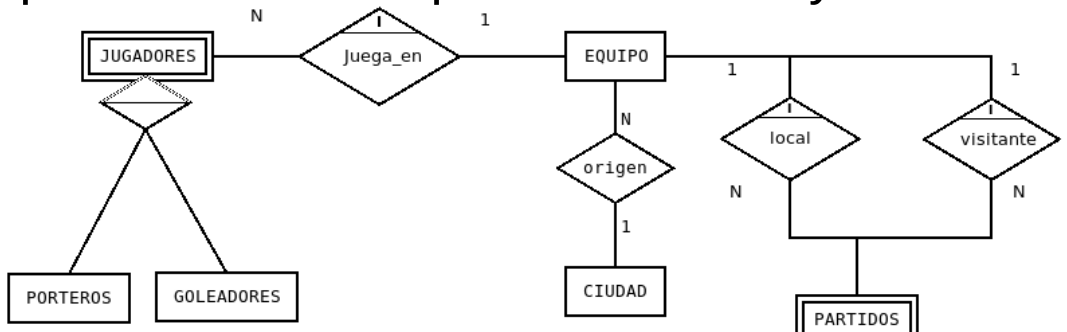
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

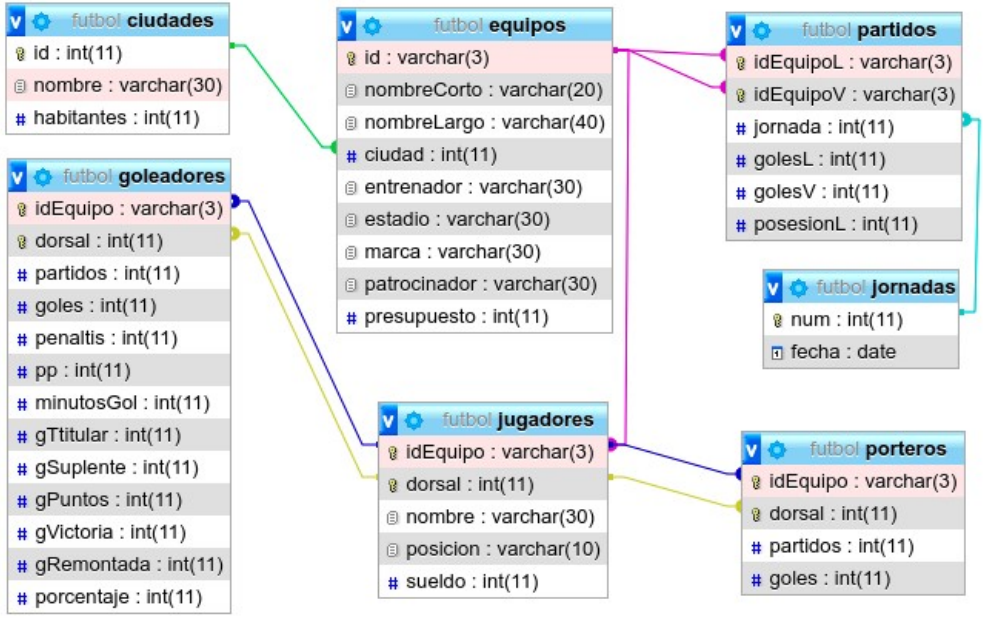


Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

… **Continuación 5º Entender la base de datos de futbol y sus relaciones:**

Realmente ***es buena práctica tener una clave primaria de 1 solo atributo y no claves primarias compuestas por varios campos***, pero en este ejemplo lo haremos así para que el alumno sepa como crear y utilizar esas claves primarias compuestas por varios atributos en Spring Boot:





Aclaraciones:

Partidos:

golesL= goles equipo local;
golesV= goles equipo visitante;

Goleadores:

partidos= partidos jugados;
goles= goles marcados;
pp= goles en propia puerta;
minutosGol= minutos para marcar un gol;
gTitular= goles como titular;
gPuntos= goles que dieron puntos;
gVictoria= goles que dieron la victoria;
Porcentaje= porcentaje de goles del equipo;

Puede resultarte muy útil ***comprobar el script con el código de creación de la bd desde phpMyAdmin para ver como hemos implementado estas relaciones entre tablas en MySQL***.

Pasando a nuestro proyecto Spring Boot, hasta ahora solo hemos hecho consultas sobre una única tabla, ***no hemos hecho consultas que impliquen varias tablas relacionadas entre si utilizando Spring Boot***.

¿ Como podemos relacionar 2 tablas con Spring Data en una consulta?

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Para esta 2º parte de la práctica vamos a crear otro proyecto REST que hará un CRUD sobre nuestra Base de Datos de futbol en MySQL.

Los pasos a seguir para crear un API RESTful:

1º Crear el proyecto Spring Boot
2º Configurar la Base de Datos
3º Crear las entidades JPA
4º Crear el repositorio para cada entidad JPA
5º Crear la capa de servicios
6º Crear la capa de control (controller)
7º Pruebas

Nota: Con H2 habían 8 pasos en total. Eso significa que había un paso más, el antiguo paso 4º donde inicializábamos la BD. Como ya la tenemos creada e inicializada en MySQL ahora no nos hará falta.

1º Crear el proyecto Spring Boot:

Para empezar vamos a crear nuestro proyecto REST llamado **“api_rest_mysql_futbol”** igual como hicimos en “dwes_futbol_rest”:

Java Projects\Añadir (+) \Spring Boot \Proyecto Maven \ versión 2.7.8 \Java

e introducimos los siguientes datos:

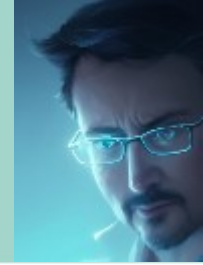
Group Id: edu.alumno.NombreAlumno
Artifact Id: api_rest_mysql_futbol
Package: war
Versión de Java: 11

Dependencias:

Deberemos seleccionar las mismas dependencias del otro proyecto:
“Spring Web” ,***“MySQL”*** ,”Spring Data JPA”, “Lombok” y añadir “DevTools” y “Validation” y le damos al intro.

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



... continuación 1º Crear el proyecto Spring Boot:

La gran diferencia del pom.xml generado en este proyecto es que hasta ahora teniamos la dependencia h2 de com.h2database y la hemos cambiado por la de mysql-connector-j de com.mysql .

pom.xml

home > joseramon > ProyectosVSCode > dwes_futbol_rest > pom.xml >

```
38          <dependency>
39              <groupId>com.h2database</groupId>
40              <artifactId>h2</artifactId>
41              <scope>runtime</scope>
42          </dependency>
```

pom.xml > project > dependencies > dependency

```
41          </dependency>
42          <dependency>
43              <groupId>com.mysql</groupId>
44              <artifactId>mysql-connector-j</artifactId>
45              <scope>runtime</scope>
46          </dependency>
47          <dependency>
```

Para continuar deberemos de copiar las propiedades que indican la versión de lombok y mapstruct. Lo más cómodo es que vayamos al proyecto “dwes_futbol_rest” y copiemos esas propiedades:

pom.xml > project > properties > org.projectlombok.version

```
12      <artifactId>api_rest_mysql_futbol</artifactId>
13      <version>0.0.1-SNAPSHOT</version>
14      <packaging>war</packaging>
15      <name>api_rest_mysql_futbol</name>
16      <description>Demo project for Spring Boot</description>
17      <properties>
18          <java.version>11</java.version>
19          <org.projectlombok.version>1.18.24</org.projectlombok.version>
20          <org.mapstruct.version>1.4.1.Final</org.mapstruct.version>
21      </properties>
22      <dependencies>
23          <dependency>
24              <groupId>org.springframework.boot</groupId>
25              <artifactId>spring-boot-starter-data-jpa</artifactId>
26          </dependency>
```

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

… continuación 1º Crear el proyecto Spring Boot:

Para continuar deberemos de copiar la dependencia del starter de Tomcat y MapStruct y plugins como el compilador de maven.

Lo más cómodo es que vayamos al proyecto “dwes_futbol_rest” y copiemos de su fichero pom.xml esas 2 dependencias y el subapartado build entero para que quede como en la imagen:

pom.xml

```
59              <artifactId>spring-boot-starter-test</artifactId>
60              <scope>test</scope>
61          </dependency>
62          <dependency>
63              <groupId>org.springframework.boot</groupId>
64              <artifactId>spring-boot-starter-tomcat</artifactId>
65              <scope>provided</scope>
66          </dependency>
67          <!--  DTOs  -->
68          <dependency>
69              <groupId>org.mapstruct</groupId>
70              <artifactId>mapstruct</artifactId>
71              <version>${org.mapstruct.version}</version>
72              <scope>compile</scope>
73          </dependency>
74      </dependencies>
75
76      <build>
77          <plugins>
78              <plugin>
79                  <groupId>org.springframework.boot</groupId>
80                  <artifactId>spring-boot-maven-plugin</artifactId>
81                  <configuration>
82                      <excludes>
83                          <exclude>
84                              <groupId>org.projectlombok</groupId>
85                              <artifactId>lombok</artifactId>
86                          </exclude>
87                      </excludes>
88                  </configuration>
89              </plugin>
90              <plugin>
91                  <groupId>org.apache.maven.plugins</groupId>
92                  <artifactId>maven-compiler-plugin</artifactId>
93                  <version>3.8.1</version>
94                  <configuration>
95                      <verbose>true</verbose>
96                      <source>1.8</source>
97                      <target>1.8</target>
98                      <encoding>UTF-8</encoding>
99                      <showWarnings>true</showWarnings>
100                     <annotationProcessorPaths>
101                         <path>
102                             <groupId>org.projectlombok</groupId>
103                             <artifactId>lombok</artifactId>
104                             <version>${org.projectlombok.version}</version>
105                         </path>
106                         <!-- Necesario para que Lombok funcione con MapStruct -->
```

2º Configurar la Base de Datos:

Teniendo en cuenta el fichero “aplication.properties” del proyecto “dwes_futbol_rest” que utilizaba una base de datos H2

¿Que cambios haremos para pasar a una Base de Datos MySQL?

application.properties

src > main > resources > application.properties

```
#Activar consola para acceder a la BD H2 via navegador
# localhost:puertoConfigurado/h2-console
spring.h2.console.enabled=true
#Configuración de la BD H2
spring.datasource.url=jdbc:h2:mem:testDwesDb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=dwes
spring.datasource.password=Simarro@1
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

#En la versión 2.4.2 no hace falta, pero en la
# 2.6.2 hace falta para poder hacer inserts en data.sql
spring.jpa.hibernate.ddl-auto=none

#CONFIGURACIÓN SOLO durante las pruebas:
# Habilitar estadísticas hibernate
spring.jpa.properties.hibernate.generate_statistics=true
# Habilitar LOGGER de las estadísticas de hibernate
logging.level.org.hibernate.stat=
# Mostrar que consultas esta realizando Hibernate
spring.jpa.show-sql=true
# Formatear las consultas
spring.jpa.properties.hibernate.format_sql=true
# Mostrar los parametros que estan enviandose a las consultas
logging.level.org.hibernate.type=debug
#FIN CONFIGURACIÓN SOLO durante las pruebas
```

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

... Continuación 2º Configurar la Base de Datos:

Realmente hay pocos cambios. Debemos ***modificar la url a la que apunta la base de datos, el usuario y el pw y el dialecto*** a utilizar para comunicarnos con la base de datos. Adicionalmente actualizamos el valor "ddl-auto" para que quede claro que Spring boot puede definir/crear/alterar tablas y datos en la base de datos:

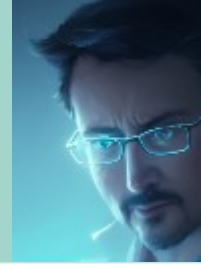
application.properties

src > main > resources > application.properties

```
# url a la base de datos MySQL
# futbol=nombre de la base de datos que hemos creado en MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/futbol
# nombre de usuario y contraseña
spring.datasource.username=futbol
spring.datasource.password=futbolSimarro

#En H2 utilizabamos el dialecto H2Dialect, aquí MySQL8Dialect
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
#Para evitar que nombreLargo lo mapee como nombre_largo y no lo encuentre
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

# actualizar base de datos y crear entidades
spring.jpa.hibernate.ddl-auto=update

#CONFIGURACIÓN SOLO durante las pruebas:
# Habilitar estadísticas hibernate
spring.jpa.properties.hibernate.generate_statistics=true
# Habilitar LOGGER de las estadísticas de hibernate
logging.level.org.hibernate.stat=debug
# Mostrar que consultas esta realizando Hibernate
spring.jpa.show-sql=true
# Formatear las consultas
spring.jpa.properties.hibernate.format_sql=true
# Mostrar los parametros que estan enviandose a las consultas
logging.level.org.hibernate.type=debug
#FIN CONFIGURACIÓN SOLO durante las pruebas
```

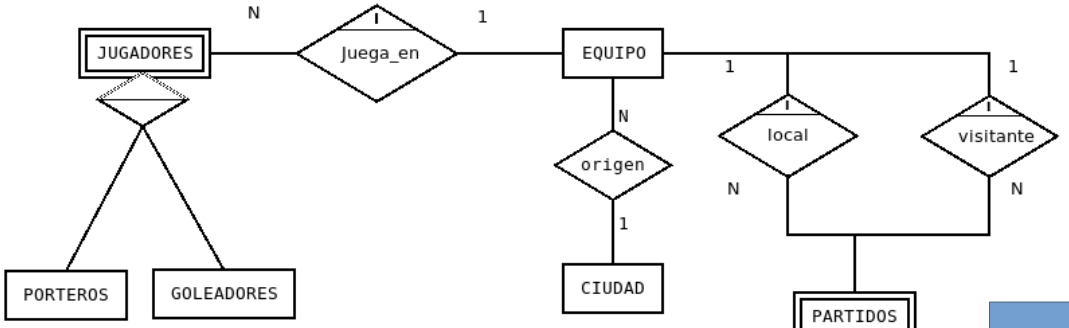
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

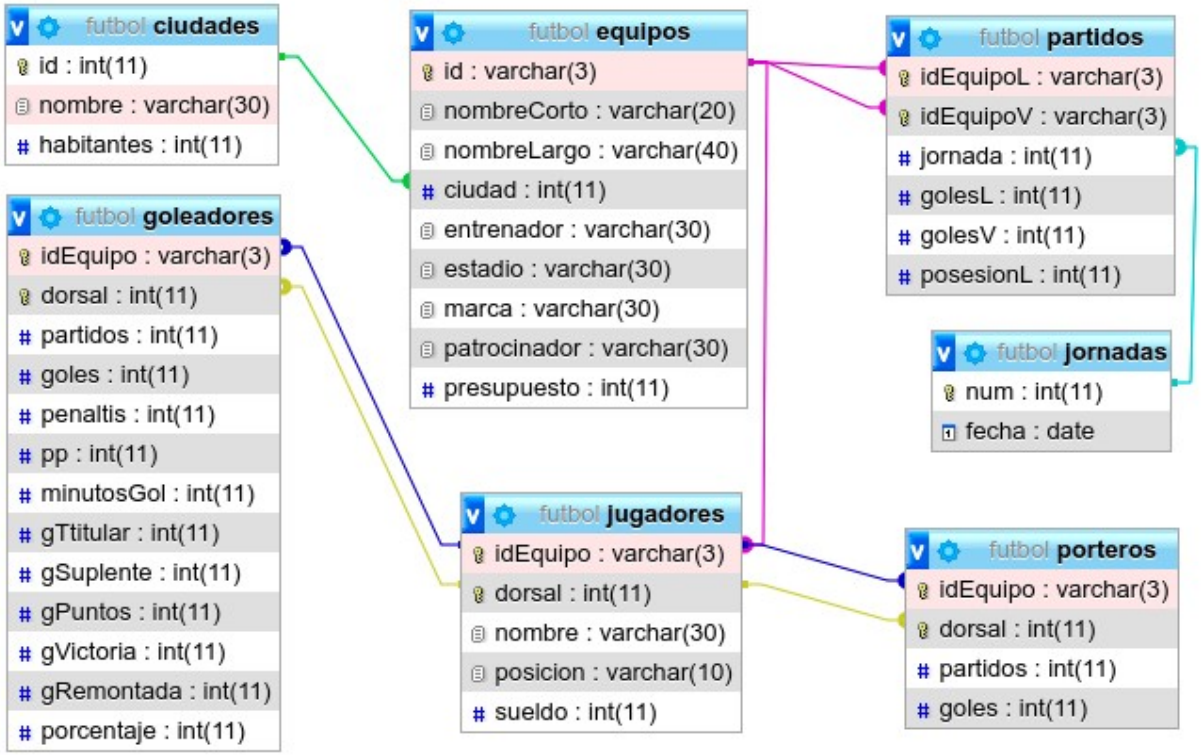
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



3º Crear las entidades JPA:

Recordemos la visión global de la Bd de futbol creada en MySQL:





Aclaraciones:

Partidos:

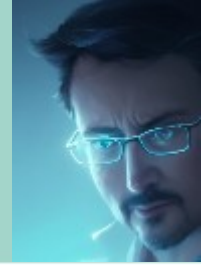
golesL= goles equipo local;
golesV= goles equipo visitante;

Goleadores:

partidos= partidos jugados;
goles= goles marcados;
pp= goles en propia puerta;
minutosGol= minutos para marcar un gol;
gTitular= goles como titular;
gPuntos= goles que dieron puntos;
gVictoria= goles que dieron la victoria;
Porcentaje= porcentaje de goles del equipo;

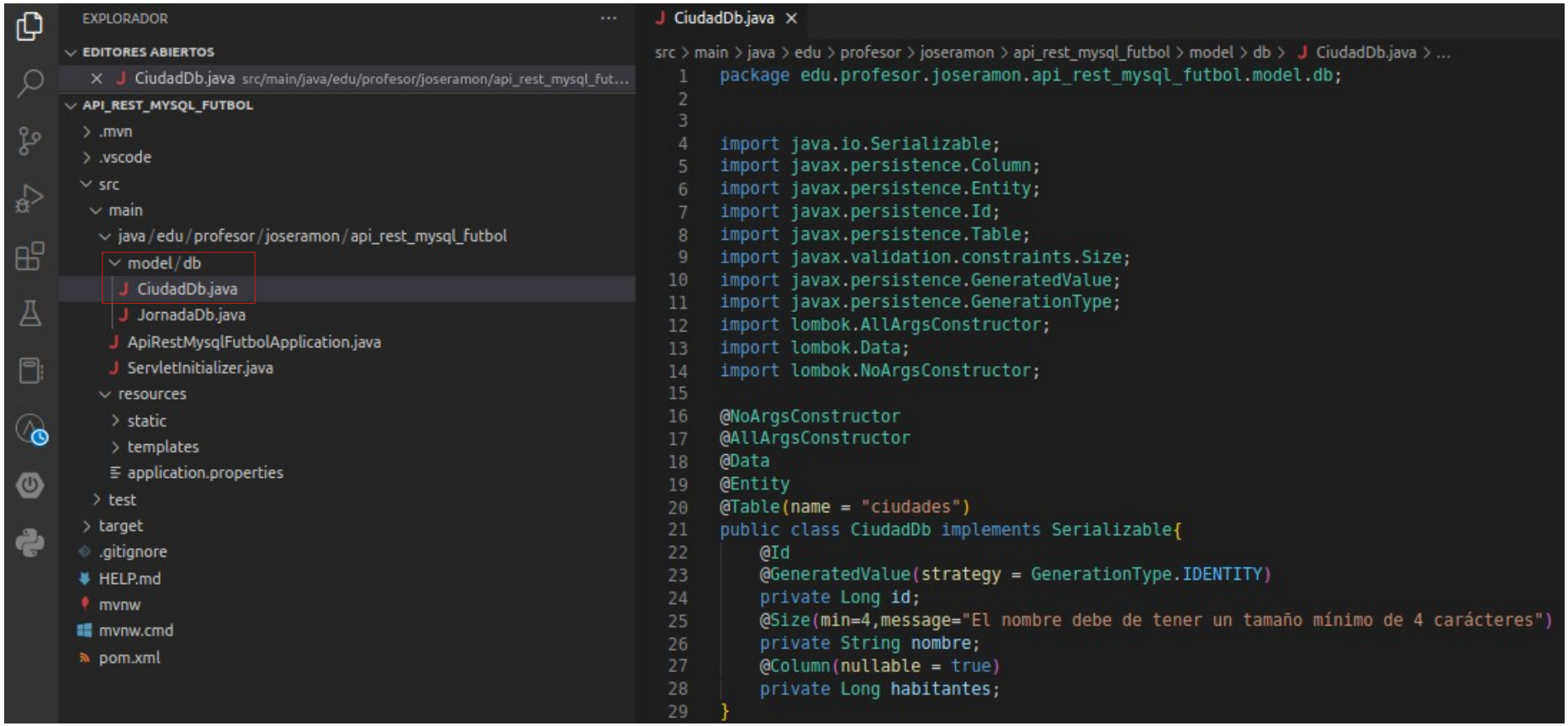
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



… continuación 3º Crear las entidades JPA:

Empezaremos de momento por copiar la entidad del proyecto anterior "CiudadDb.java" dentro del package api_rest_mysql_futbol.**model.db** y más adelante ya veremos como creamos el resto de tablas y como relacionarlas:



Ya podemos comprobar si la tabla esta bien enlazada contra MySQL haciendo un clean y un install con maven. Si no lanza error ¡¡Enhorabuena!!

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

4º Crear el repositorio para cada entidad JPA:

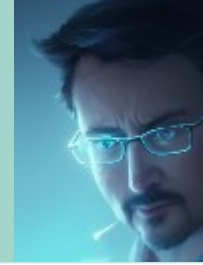
Para hacer **CRUD (Create Read Update Delete)** sobre la **tabla "Ciudades"** volvemos a copiar del proyecto anterior CiudadRepository donde extendemos de la interface JpaRepository de Spring. En las nuevas versiones de Spring Boot ya no hace falta indicar la anotación "@Repository":

CiudadRepository.java

src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > repository > CiudadRepository.java > ...

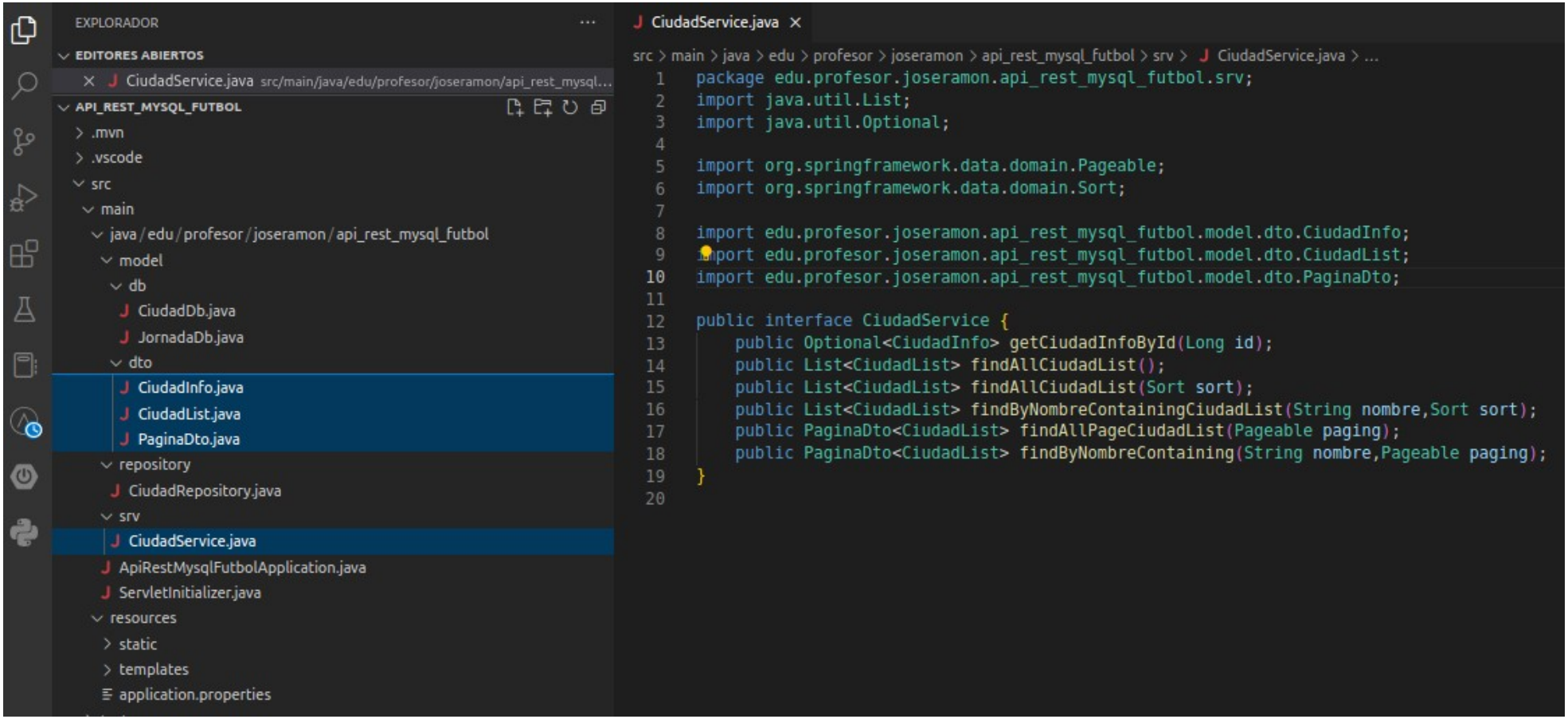
```java
package edu.profesor.joseramon.api_rest_mysql_futbol.repository;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.repository.JpaRepository;
import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.CiudadDb;

public interface CiudadRepository extends JpaRepository<CiudadDb,Long>{
    //Métodos automáticos en Spring Data JPA
    List<CiudadDb> findAll(Sort sort);
    //Filtrado por nombre usando equivalente a 'LIKE' usando "Containing"
    List<CiudadDb> findByNombreContaining(String nombre,Sort sort);
    //Paginación
    Page<CiudadDb> findAll(Pageable pageable);
    Page<CiudadDb> findByNombreContaining(String nombre,Pageable pageable);
}
```

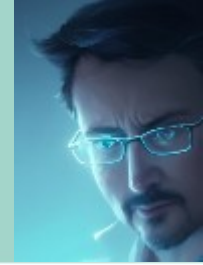
¡¡Fijate que salvo modificar "aplication.properties" no nos ha hecho falta modificar nada en las clases Java para que en vez de utilizar la Base de Datos H2 las consultas se realicen sobre una Base de Datos MySQL!!



5º Crear la capa de servicios:

Para crear la capa de servicios realizamos el mismo proceso, copiamos CiudadInfo,CiudadLista,PaginaDto y CiudadService del proyecto anterior y los dejamos en los subpaquetes correspondientes:

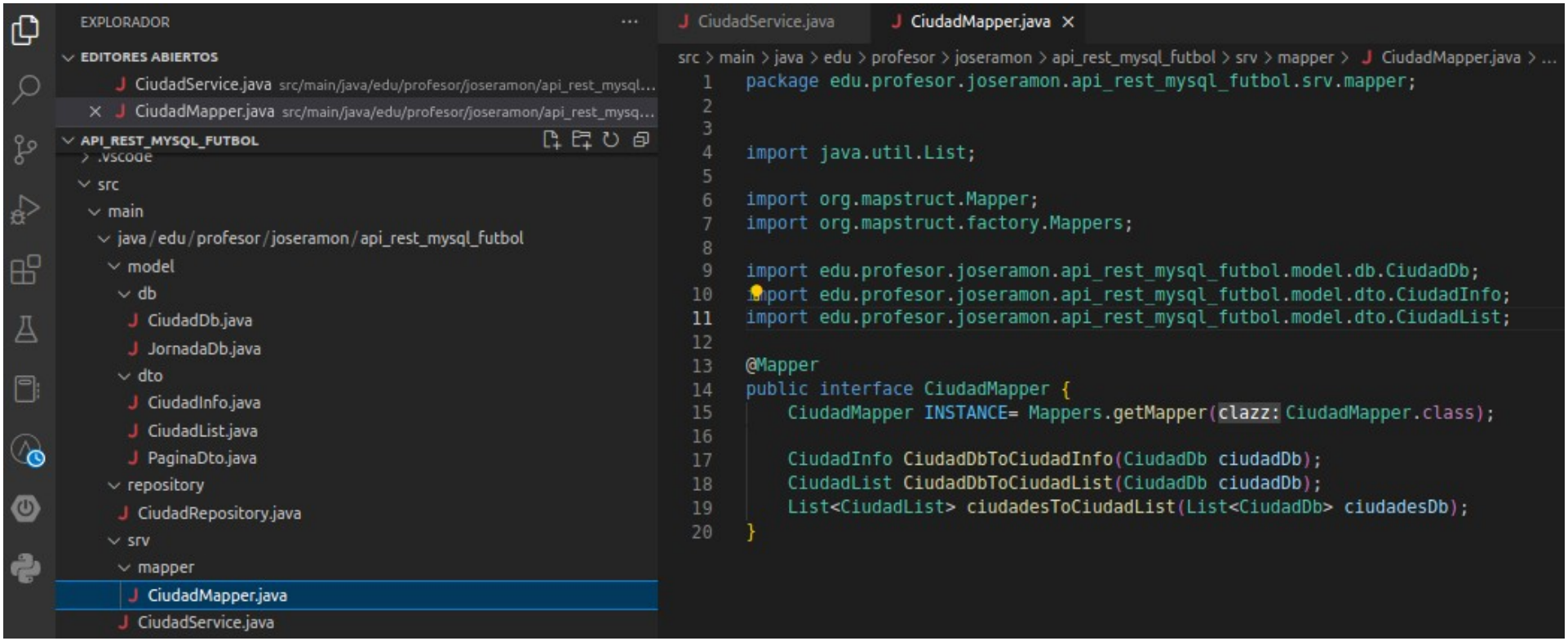




... continuación 5º Crear la capa de servicios:

Mappers:

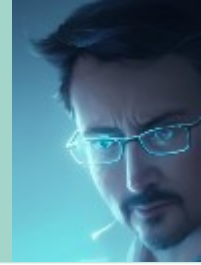
Para poder realizar la implementación del servicio necesitaremos copiar CiudadMapper.java:



NOTA: No se han implementado todos los métodos (guardar, modificar,…) porque en esta practica se pretende analizar como se puede realizar consultas con más de una tabla.

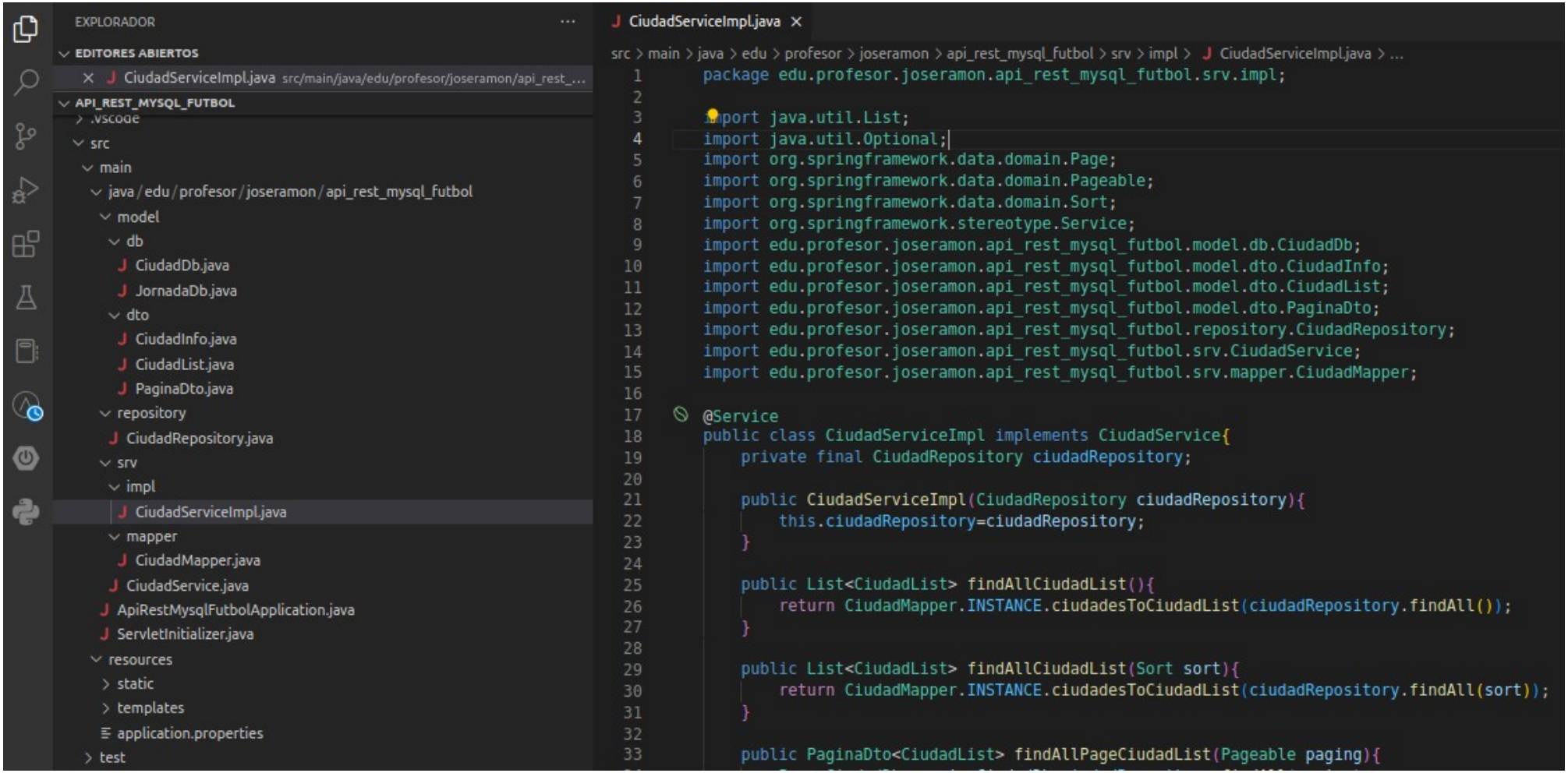
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



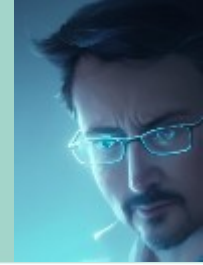
... continuación 5º Crear la capa de servicios:

Y por último implementaremos los métodos del servicio copiandolos de CiudadServiceImpl.java:



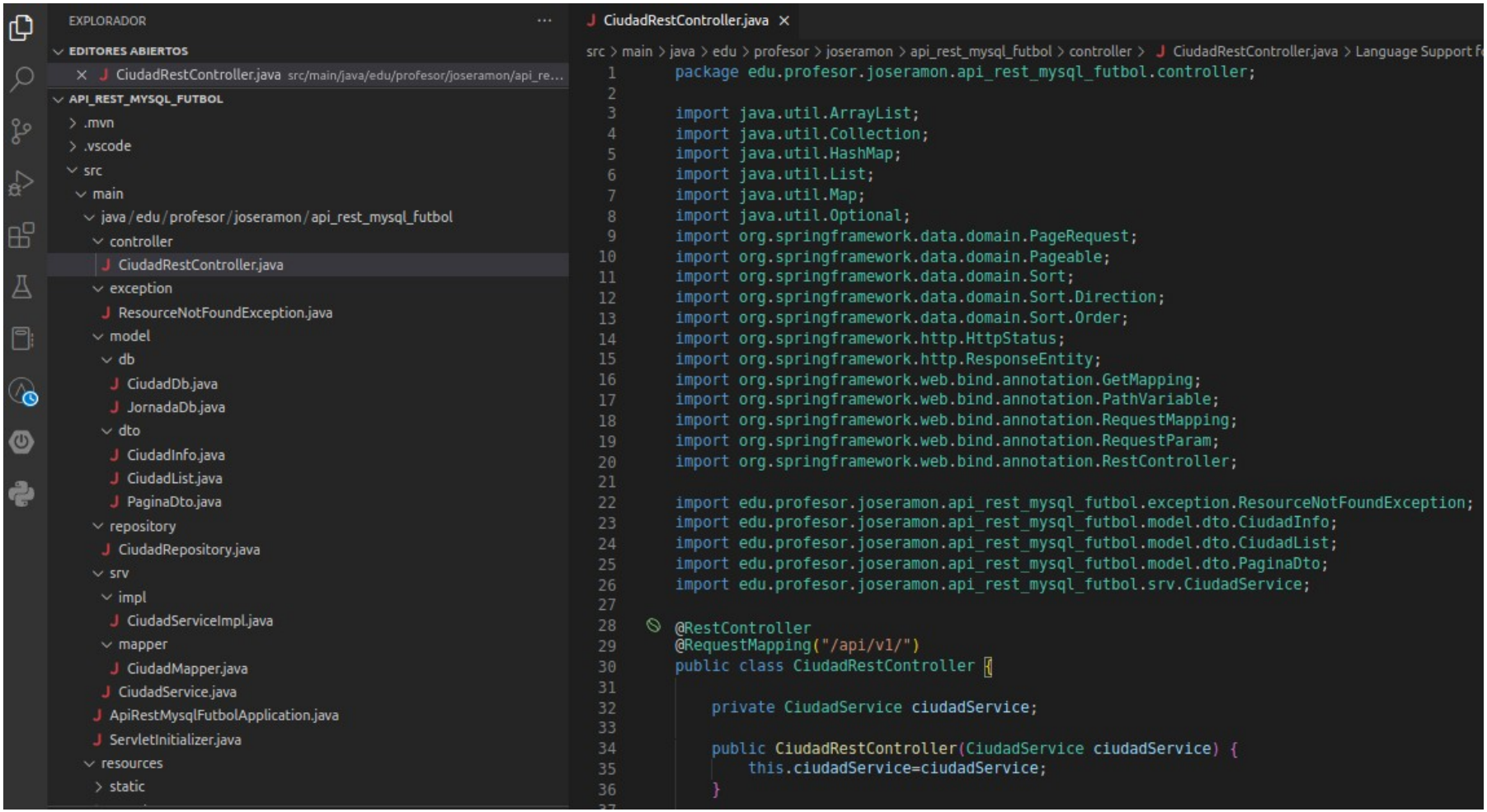
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



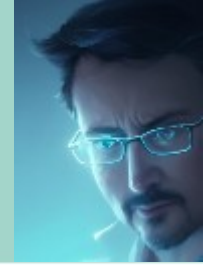
6º Crear la capa de control (controller):

Ahora implementamos los métodos de consulta sobre ciudad copiando CiudadRestController.java y la excepción personalizada:



UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



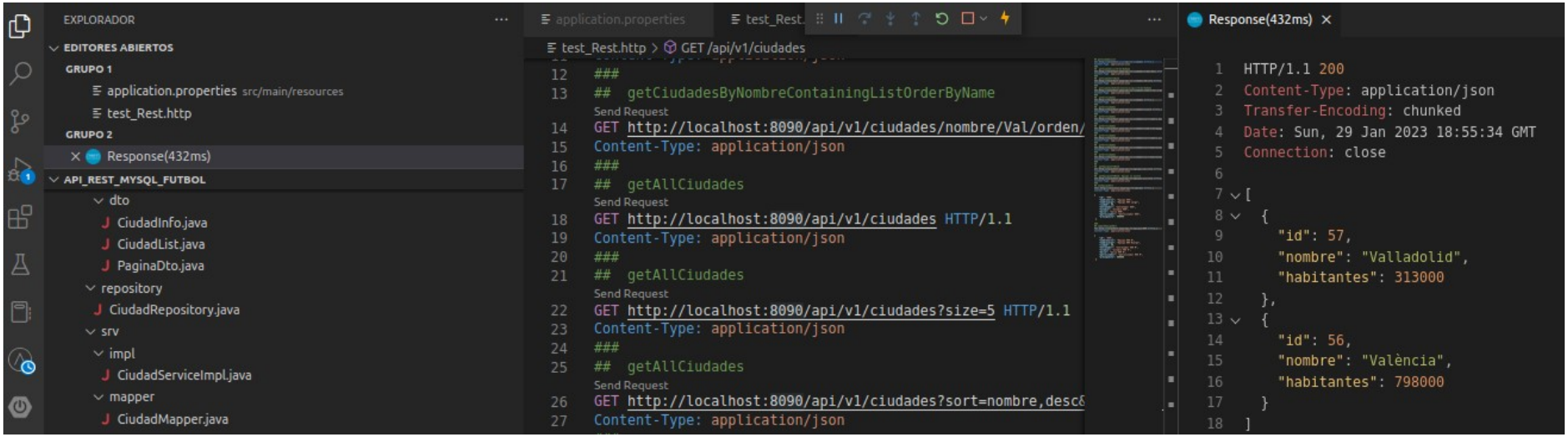
7º Pruebas:

Utilizando la extensión de VSCode “REST Client” podemos copiar el fichero test_Rest.http . Para que funcione correctamente tendremos que asegurarnos que el puerto de Xampp no sea el 8080. Si es el caso tendremos que cambiar el puerto de Spring boot a otro puerto (por ejemplo 8090), realizar los cambios en el fichero test_Rest.http y comprobar que funciona correctamente sobre MySQL:

application.properties | test_Rest.http

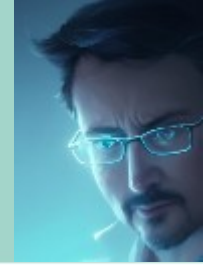
src > main > resources > application.properties

```
#Cambiamos puerto para evitar solape con Xampp
server.port=8090

# url a la base de datos MySQL
# futbol=nombre de la base de datos que hemos creado en MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/futbol
# nombre de usuario y contraseña
spring.datasource.username=futbol
spring.datasource.password=futbolSimarro
```



UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

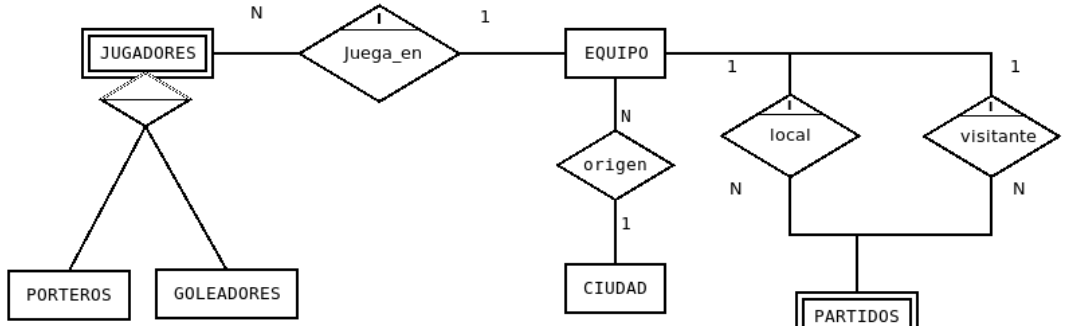


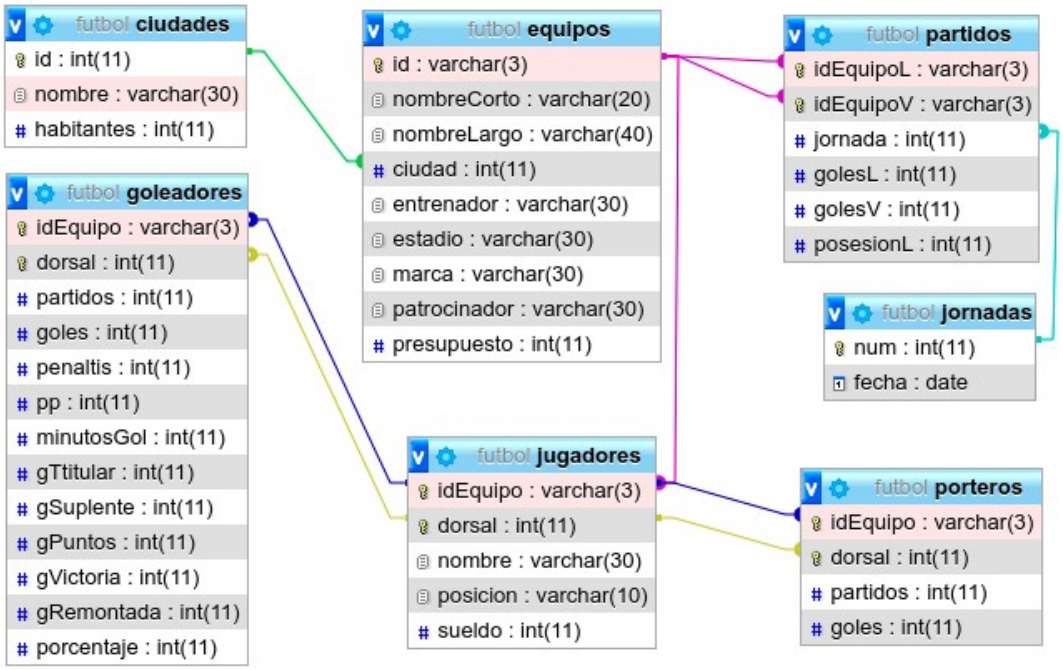
Tablas relacionadas:

Si nos fijamos en la visión global de la Base de Datos, los equipos pertenecen a una ciudad y en una ciudad pueden haber más de un equipo.

Nos han pedido sacar un ***listado de Equipos que indique el nombre de su ciudad***:

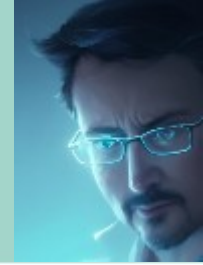
¿Como podemos hacerlo?





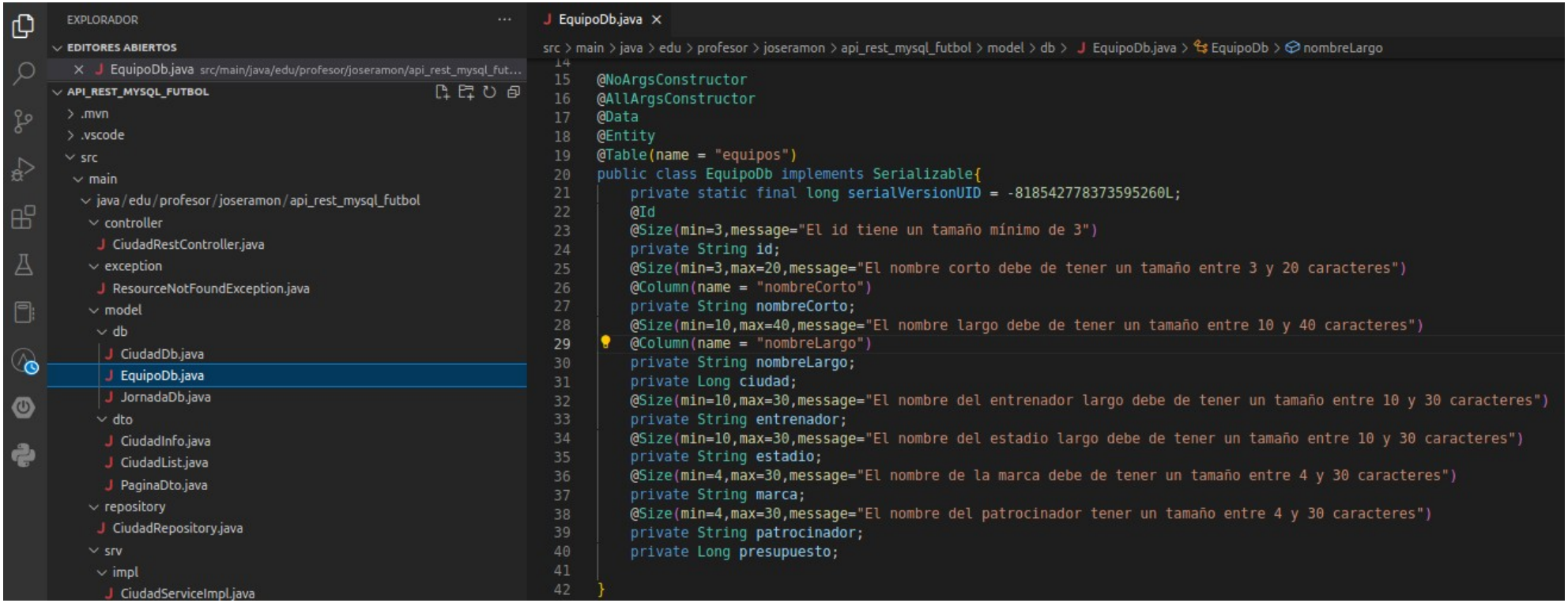
UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Tablas relacionadas:

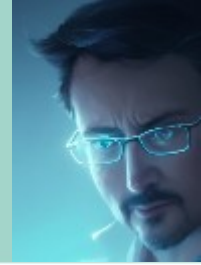
Empezamos reutilizando la definición de “EquipoDb.java” de un proyecto anterior. Copiala y comprueba si da errores haciendo un clean y un install de maven:



Si te fijas la ciudad es un numérico (clave ajena), ¿Como podemos enlazar el equipo a su ciudad para poder mostrar el nombre?

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Tablas relacionadas: @ManyToOne

Quitamos el ***numérico ciudad***, que realmente es una ***clave ajena que apunta a la tabla ciudad*** y lo sustituimos por una linea en la que hacemos ***referencia al objeto ciudadDb*** al que pertenece el equipo. Vamos a ***enlazar EquipoDb con CiudadDb***:

Puesto que una ciudad puede tener muchos equipos y un equipo solo puede tener una ciudad la notación que debemos utilizar estando en EquipoDb es ***@ManyToOne*** porque muchos equipos pueden hacer referencia a la misma ciudad a través de su clave ajena, ***@joinColumn***, llamada 'ciudad' en MySQL:

EquipoDb.java

src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > model > db > EquipoDb.java > EquipoDb > ciudadDb

```
14  import javax.persistence.JoinColumn;
15  import javax.persistence.ManyToOne;
16
17  @NoArgsConstructor
18  @AllArgsConstructor
19  @Data
20  @Entity
21  @Table(name = "equipos")
22  public class EquipoDb implements Serializable{
23      private static final long serialVersionUID = -8185427783735952 60L;
24      @Id
25      @Size(min=3,message="El id tiene un tamaño mínimo de 3")
26      private String id;
27      @Size(min=3,max=20,message="El nombre corto debe de tener un tamaño entre 3 y 20 caracteres")
28      @Column(name = "nombreCorto")
29      private String nombreCorto;
30      @Size(min=10,max=40,message="El nombre largo debe de tener un tamaño entre 10 y 40 caracteres")
31      @Column(name = "nombreLargo")
32      private String nombreLargo;
33      @ManyToOne
34      @JoinColumn(name = "ciudad")
35      private CiudadDb ciudadDb;
36      @Size(min=10,max=30,message="El nombre del entrenador largo debe de tener un tamaño entre 10 y 30 caracteres")
37      private String entrenador;
38      @Size(min=10,max=30,message="El nombre del estadio largo debe de tener un tamaño entre 10 y 30 caracteres")
39      private String estadio;
40      @Size(min=4,max=30,message="El nombre de la marca debe de tener un tamaño entre 4 y 30 caracteres")
41      private String marca;
42      @Size(min=4,max=30,message="El nombre del patrocinador tener un tamaño entre 4 y 30 caracteres")
43      private String patrocinador;
44      private Long presupuesto;
45
46  }
```

¿Como continuamos ahora que tenemos CiudadDb en EquipoDb?

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Tablas relacionadas: @ManyToOne

Para mostrar un listado de equipos con el nombre de su ciudad, primero debemos crear una clase de repositorio que extienda de JpaRepository para acceder a los datos de la entidad "EquipoDb". Después, creamos una clase de servicio para realizar operaciones en el repositorio y finalmente un controlador para gestionar las peticiones HTTP y devolver la respuesta en formato JSON.

EquipoRepository.java

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > repository > EquipoRepository.java > ...
1   package edu.profesor.joseramon.api_rest_mysql_futbol.repository;
2   import org.springframework.data.jpa.repository.JpaRepository;
3
4   import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.EquipoDb;
5
6   public interface EquipoRepository extends JpaRepository<EquipoDb, String> {
7   }
8
```

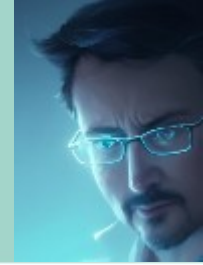
Antes de pasar al servicio tenemos que crear un EquipoList, que contenga el id, el nombre largo y el nombre de la ciudad:

EquipoList.java

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > model > dto > EquipoList.java > EquipoList > nombreCiudad
1   package edu.profesor.joseramon.api_rest_mysql_futbol.model.dto;
2
3   import javax.validation.constraints.Size;
4
5   import lombok.AllArgsConstructor;
6   import lombok.Data;
7   import lombok.NoArgsConstructor;
8
9   @NoArgsConstructor
10  @AllArgsConstructor
11  @Data
12  public class EquipoList {
13      @Size(min=3,message="El id tiene un tamaño mínimo de 3")
14      private String id;
15      @Size(min=10,max=40,message="El nombre largo debe de tener un tamaño entre 10 y 40 caracteres")
16      private String nombreLargo;
17      private String nombreCiudad;
18  }
```

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



Tablas relacionadas: @ManyToOne

También nos hará falta un mapper para poder convertir de EquipoDb a EquipoList teniendo en cuenta que debemos convertir el atributo ciudadDb de tipo CiudadDb a un String nombreCiudad en EquipoList:

EquipoMapper.java

src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > mapper > EquipoMapper.java

```java
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.factory.Mappers;

import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.EquipoDb;
import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.EquipoList;

@Mapper
public interface EquipoMapper {
    EquipoMapper INSTANCE= Mappers.getMapper(clazz: EquipoMapper.class);

    @Mapping(target = "nombreCiudad", source = "ciudadDb.nombre")
    EquipoList EquipoDbToEquipoList(EquipoDb equipoDb);

    List<EquipoList> equiposDbToEquiposList(List<EquipoDb> equiposDb);

}
```

EquipoServiceImpl.java

src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > impl > EquipoServiceImpl.java > ...

```java
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

@Service
public class EquipoServiceImpl implements EquipoService{
    private final EquipoRepository equipoRepository;

    public EquipoServiceImpl(EquipoRepository equipoRepository){
        this.equipoRepository=equipoRepository;
    }

    @Override
    public PaginaDto<EquipoList> findAll(Pageable paging) {
        Page<EquipoDb> paginaEquipoDb=equipoRepository.findAll(paging);
        return new PaginaDto<EquipoList>(
                    paginaEquipoDb.getNumber(),//número de página solicitada
                    paginaEquipoDb.getSize(), //tamaño de la página
                    paginaEquipoDb.getTotalElements(),//total de elementos devueltos por la cor
                    paginaEquipoDb.getTotalPages(), //total páginas teniendo en cuenta el tamañ
                    EquipoMapper.INSTANCE.equiposDbToEquiposList(paginaEquipoDb.getContent()),/
                    paginaEquipoDb.getSort()); //ordenación de la consulta
    }
}
```

EquipoService.java

src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > EquipoService.java > ...

```java
package edu.profesor.joseramon.api_rest_mysql_futbol.srv;

import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.EquipoList;
import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.PaginaDto;
import org.springframework.data.domain.Pageable;

public interface EquipoService {
    public PaginaDto<EquipoList> findAll(Pageable paging);
}
```

UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

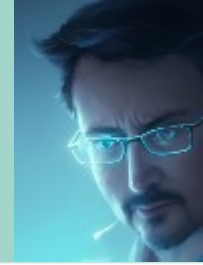
Tablas relacionadas: @ManyToOne

Tan solo falta el controller al estilo y semejanza que el método getAllCiudades de CiudadRestController pero sin el parámetro “nombre” porque de momento no queremos filtrar por el nombre:

EquipoRestController.java

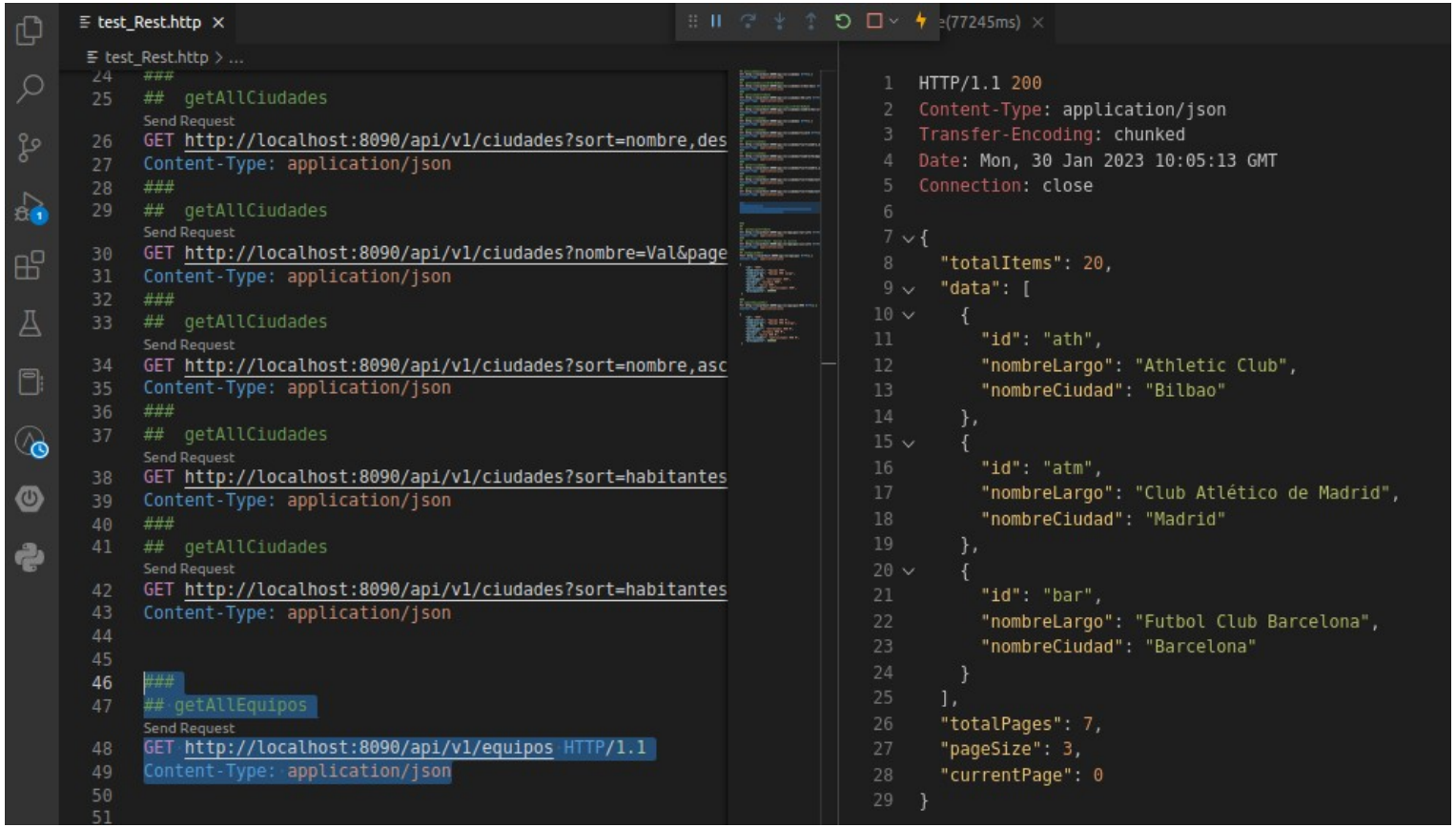
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > controller > EquipoRestController.java > Language Support for Java

```java
@GetMapping("/equipos")
public ResponseEntity<Map<String, Object>> getAllEquipos(
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "3") int size,
    @RequestParam(defaultValue = "id,asc") String[] sort) {

  try {
    // Crear sorts (ordenación de los datos)
    List<Order> criteriosOrdenacion= new ArrayList<Order>();
    //El primer criterio de ordenacion siempre deberá de contener el orden(asc,desc)
    if(sort[0].contains(",")){ //Hay más de un criterio de ordenación
     //Tenemos un vector de ordenaciones en 'sort' y debemos leerlos
     for (String criterioOrdenacion : sort) {
       String[] orden = criterioOrdenacion.split(",");
       if (orden.length > 1)
         criteriosOrdenacion.add(new Order(Direction.fromString(orden[1]), orden[0]));
       else // por defecto asc si no se dice nada
         criteriosOrdenacion.add(new Order(Direction.fromString(value: "asc"), orden[0]));
      }
    } else{ //Solo hay un criterio de ordenación
        //El primer elemento del vector de sort es la dirección y el segundo el campo
        criteriosOrdenacion.add(new Order(Direction.fromString(sort[1]), sort[0]));
    }
    Sort sorts= Sort.by(criteriosOrdenacion);

    // Crear solicitud de página nº 'page' de tamaño 'size'
    // utilizando el orden 'sorts'
    Pageable paging = PageRequest.of(page, size, sorts);

    PaginaDto<EquipoList> paginaEquiposList=equipoService.findAll(paging);

    // Rellenar datos a devolver en el servicio REST
    List<EquipoList> equipos = paginaEquiposList.getContent();
    Map<String, Object> response = new HashMap<>();
    response.put("data", equipos);
    response.put("currentPage", paginaEquiposList.getNumber());
    response.put("pageSize", paginaEquiposList.getSize());
    response.put("totalItems", paginaEquiposList.getTotalElements());
    response.put("totalPages", paginaEquiposList.getTotalPages());
    return new ResponseEntity<>(response, HttpStatus.OK);
  } catch (Exception e) { // Si hay cualquier tipo de error
    return new ResponseEntity<>(headers: null, HttpStatus.INTERNAL_SERVER_ERROR);
  }
 }
}
```



Tablas relacionadas: @ManyToOne

Podemos probarlo creando una nueva llamada para ejecutar getAllEquipos en test_Rest.http:



UD 3: Bases de datos y servicios REST
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

EJERCICIO:

Sube la aplicación final al moodle.

Para ello:

1º Haz un “Run As \Maven Clean” para dejar solo los fichero fuentes y quitar momentaneamente los necesarios para ejecutar la aplicación (dependencias).

2º Comprime la carpeta de tu aplicación y ponle como nombre UD3_practica7_nombreAlumno.tar.gz al fichero comprimido donde nombreAlumno es el nombre del alumno que entrega la práctica.

3º Súbela al moodle.

IMPORTANTE: No comprimir en RAR, porque Ubuntu no lo lee bien y en clase tenemos Ubuntu. Si tuviesemos Windows, podemos comprimir en ZIP.