



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



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

1º Acceder a phpMyAdmin:

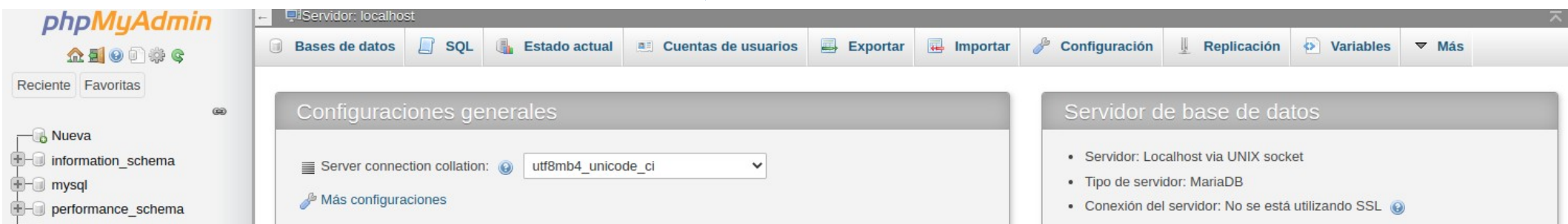
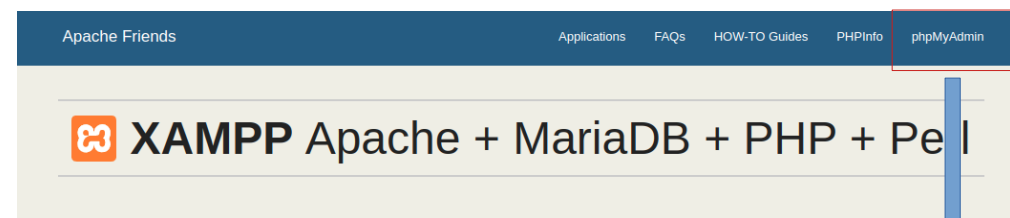
Presuponiendo que tenemos **XAMPP instalado** comprobamos que la BD esta arrancada .

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**:

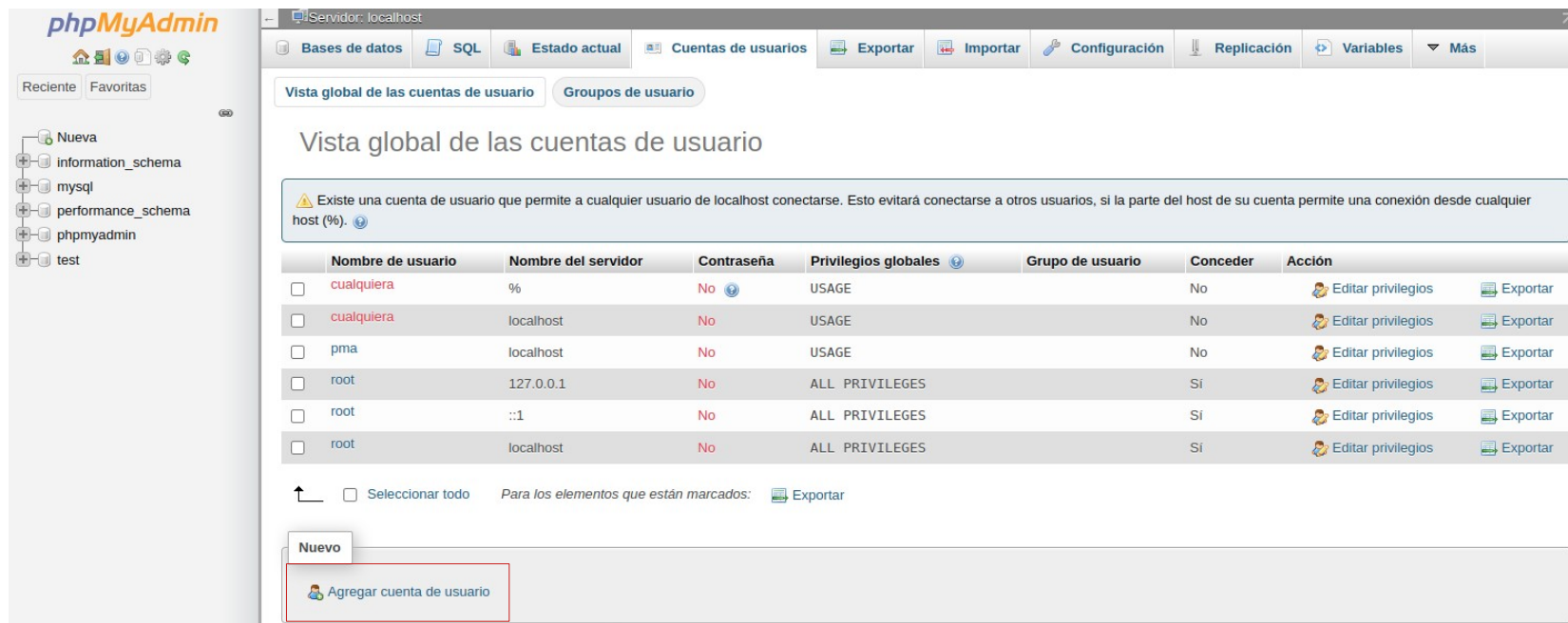
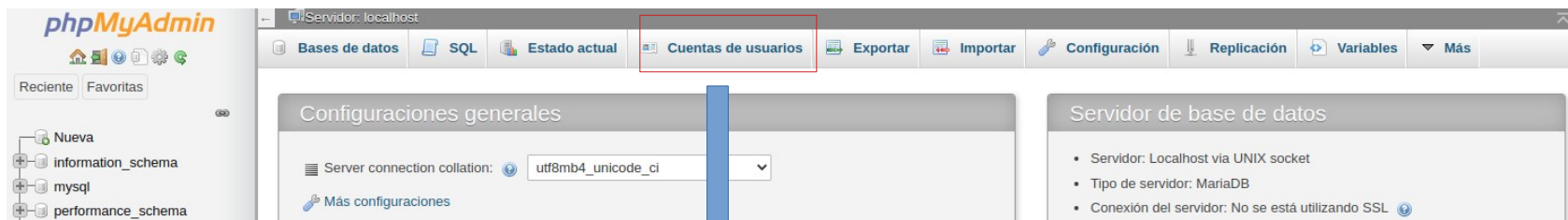


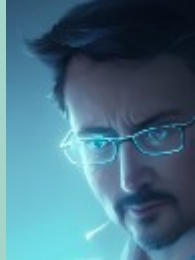


Crear la BD FUTBOL con todas sus tablas en MySQL utilizando XAMPP:

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

Continuar



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

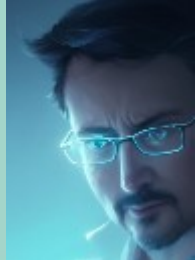
... Continuación **2º Crear un usuario nuevo y la base de datos de futbol:**

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

The screenshot shows the phpMyAdmin interface. On the left, the 'Nueva' (New) menu is open, showing a tree structure with 'futbol' selected. The main panel displays a success message: 'Ha agregado un nuevo usuario.' Below this, the SQL command is shown: `CREATE USER 'futbol'@'localhost' IDENTIFIED VIA mysql_native_password USING '***'; GRANT ALL PRIVILEGES ON *.* TO 'futbol'@'localhost' WITH GRANT OPTION; GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0; GRANT ALL PRIVILEGES ON `futbol`.* TO 'futbol'@'localhost';`. Below the SQL command, there are buttons for 'Global', 'Base de datos', 'Change password', and 'Información de la cuenta'. The 'Base de datos' button is selected. Below these buttons, there is a section for 'Privilegios globales' with a checkbox for 'Seleccionar todo' which is checked. On the right side of the screenshot, there is a dialog box asking '¿Quieres guardar la contraseña?' (Do you want to save the password?). It has a dropdown for 'Nombre de usuario' (Username) set to 'futbol' and a password field. There are 'Nunca' (Never) and 'Guardar' (Save) buttons. At the bottom of the dialog, there is a note: 'Puedes usar las contraseñas guardadas en cualquier dispositivo. Se guardan en el Gestor de contraseñas de Google para jose.ramon.profesor@gmail.com.'

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”:

phpMyAdmin

Reciente Favoritas

Nueva

- futbol
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test

Servidor: localhost » Base de datos: futbol

Estructura SQL Buscar Generar una consulta Exportar Importar Operaciones Privilegios Rutinas Eventos Disparadores

Ejecutar la(s) consulta(s) SQL en la base de datos futbol:

```

1  -- Copyright Jose Ramón Cebolla - 2023
2
3  -----
4  -- Table `ciudades`
5  -----
6  DROP TABLE IF EXISTS `ciudades` ;
7
8  CREATE TABLE IF NOT EXISTS `ciudades` (
9    `id` INT(11) NOT NULL ,
10   `nombre` VARCHAR(30) NOT NULL ,
11   `habitantes` INT(11) NULL DEFAULT NULL ,
12   PRIMARY KEY (`id`) )
13  ENGINE = InnoDB
14  DEFAULT CHARACTER SET = latin1;
15
16

```

Limpiar Formato Obtener consulta almacenada automáticamente

☐ Enlazar parámetros

Guardar esta consulta en favoritos:

Delimitador ; ☐ Mostrar esta consulta otra vez ☐ Mantener la caja de texto con la consulta ☐ Deshacer («rollback») al finalizar ☒ Habilite la revisión de las claves foráneas

Continuar



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

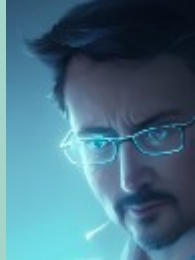
4º Comprobaciones:

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

phpMyAdmin interface showing the 'futbol' database structure. The 'Filas' column is highlighted with a red box, showing the number of rows for each table.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> ciudades	Examinar Estructura Buscar Insertar Vaciar Eliminar	58	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> equipos	Examinar Estructura Buscar Insertar Vaciar Eliminar	20	InnoDB	latin1_swedish_ci	32.0 KB	-
<input type="checkbox"/> goleadores	Examinar Estructura Buscar Insertar Vaciar Eliminar	239	InnoDB	latin1_swedish_ci	48.0 KB	-
<input type="checkbox"/> jornadas	Examinar Estructura Buscar Insertar Vaciar Eliminar	38	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> jugadores	Examinar Estructura Buscar Insertar Vaciar Eliminar	535	InnoDB	latin1_swedish_ci	64.0 KB	-
<input type="checkbox"/> partidos	Examinar Estructura Buscar Insertar Vaciar Eliminar	380	InnoDB	latin1_swedish_ci	80.0 KB	-
<input type="checkbox"/> porteros	Examinar Estructura Buscar Insertar Vaciar Eliminar	44	InnoDB	latin1_swedish_ci	16.0 KB	-
7 tablas	Número de filas	1,314	InnoDB	utf8mb4_general_ci	272.0 KB	0 B

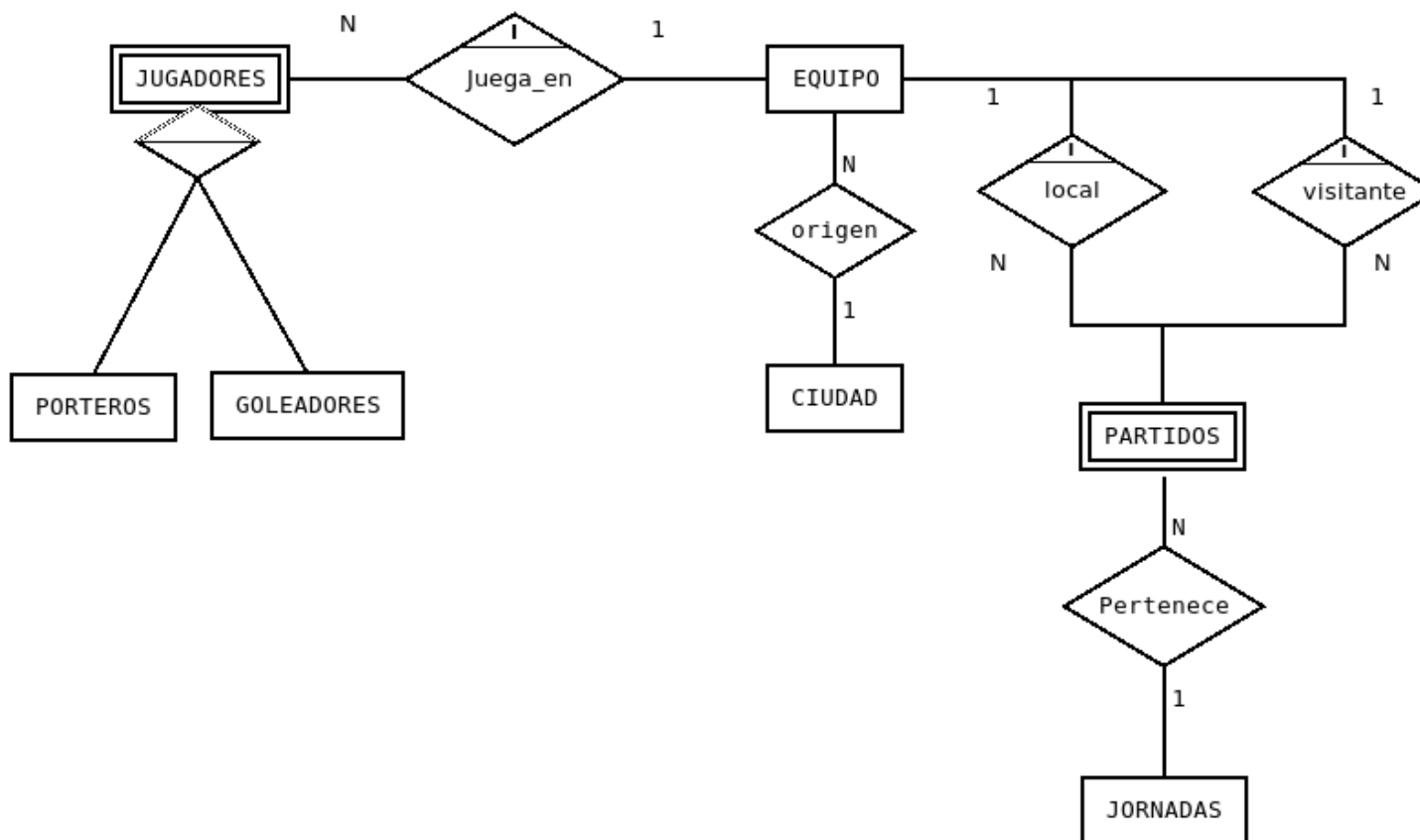
Para los elementos que están marcados: ▼



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

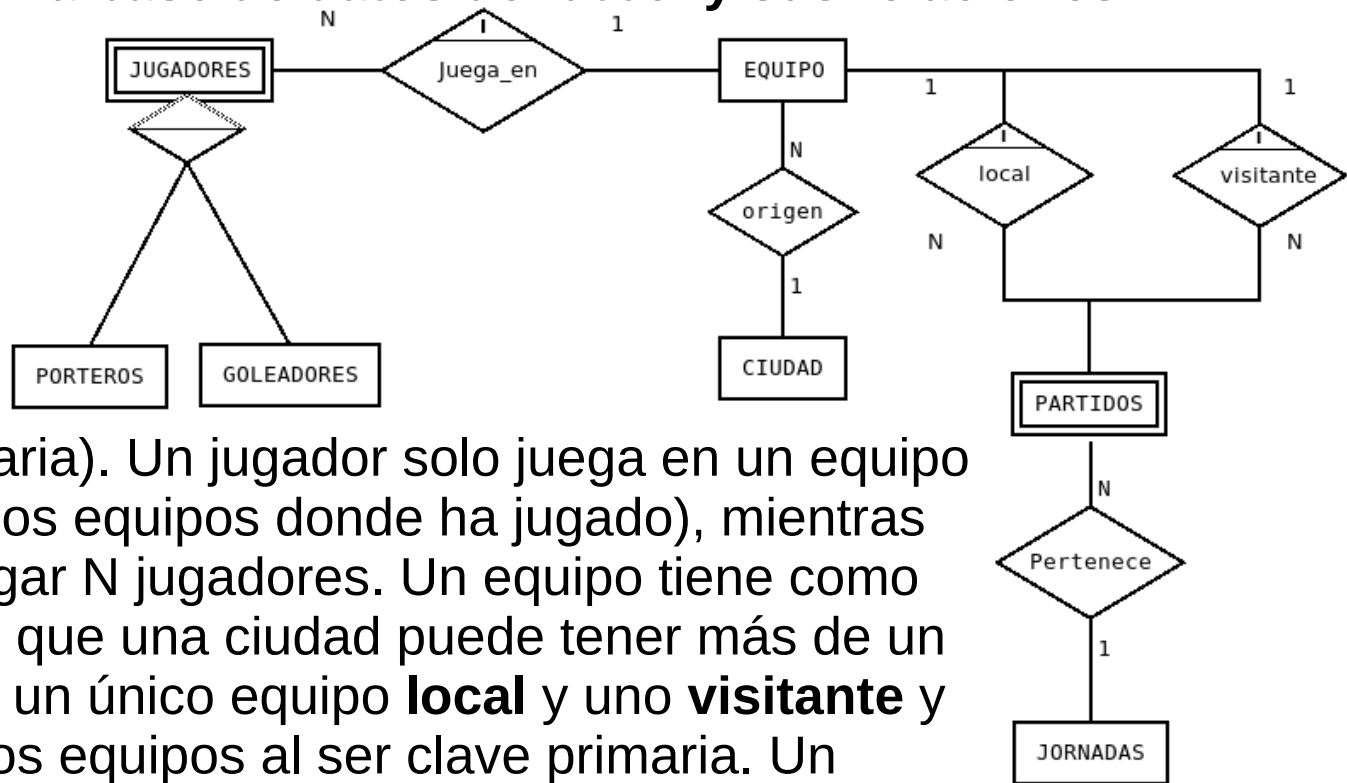
Desarrollo Web en Entorno Servidor - Joseamon.profesor@gmail.com

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

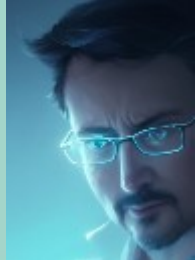
... Continuación **5º Entender la base de datos de futbol y sus relaciones:**

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.



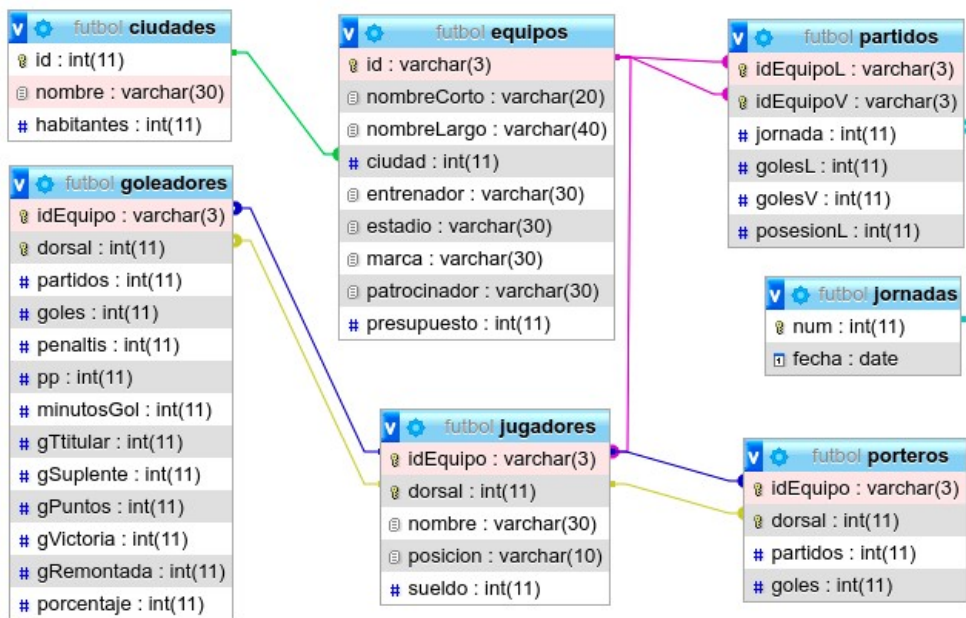
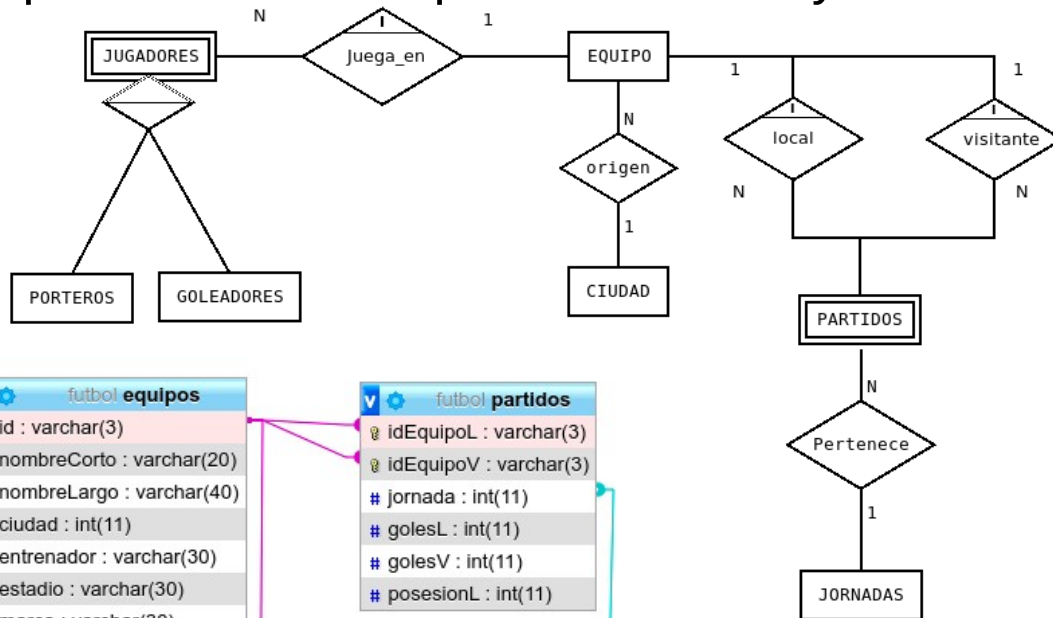
¿Te acuerdas de como traducirlo a tablas de la BD?



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;



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

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



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 - Jose Ramon.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**

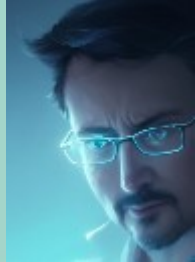
Nota: Con H2 habían 8 pasos en total. Eso significa que había un paso más, el antiguo paso 4º donde inicializábamos la BD. Como ya la tenemos creada e inicializada en MySQL ahora no nos hará falta.



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

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



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:

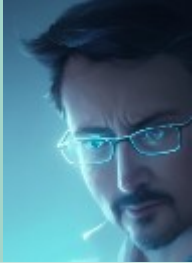
“Spring Web” , **“MySQL”** , “Spring Data JPA”, “Lombok” y añadir “DevTools” y “Validation” y le damos al intro.



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

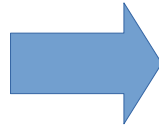
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



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

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

```
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 >
11  <groupId>com.profesor.joseramon</groupId>
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 x
pom.xml > project > build
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      </annotationProcessorPaths>
107    </configuration>
108  </plugin>
109 </plugins>
110 </build>
111 <!-- Necesario para que Lombok funcione con MapStruct -->
```



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

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



2º Configurar la Base de Datos:

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



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

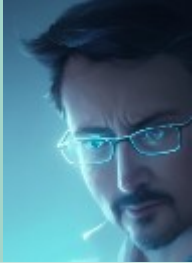
```
application.properties x
src > main > resources > application.properties
1  #Activar consola para acceder a la BD H2 via navegador
2  # localhost:puertoConfigurado/h2-console
3  spring.h2.console.enabled=true
4  #Configuración de la BD H2
5  spring.datasource.url=jdbc:h2:mem:testDwesDb
6  spring.datasource.driverClassName=org.h2.Driver
7  spring.datasource.username=dwes
8  spring.datasource.password=Simarro@1
9  spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
10
11  #En la versión 2.4.2 no hace falta, pero en la
12  # 2.6.2 hace falta para poder hacer inserts en data.sql
13  spring.jpa.hibernate.ddl-auto=none
14
15  #CONFIGURACIÓN SOLO durante las pruebas:
16  # Habilitar estadísticas hibernate
17  spring.jpa.properties.hibernate.generate_statistics=true
18  # Habilitar LOGGER de las estadísticas de hibernate
19  logging.level.org.hibernate.stat=
20  # Mostrar que consultas esta realizando Hibernate
21  spring.jpa.show-sql=true
22  # Formatear las consultas
23  spring.jpa.properties.hibernate.format_sql=true
24  # Mostrar los parametros que estan enviandose a las consultas
25  logging.level.org.hibernate.type=debug
26  #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:**

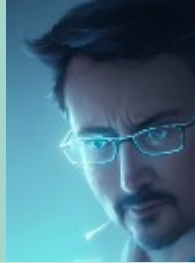
Realmente hay pocos cambios. Debemos ***modificar la url a la que apunta la base de datos, el usuario y el pw y el dialecto*** a utilizar para comunicarnos con la base de datos. Adicionalmente actualizamos el valor “ddl-auto” para que quede claro que Spring boot puede definir/crear/alterar tablas y datos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

```
application.properties X
src > main > resources > application.properties
4 # url a la base de datos MySQL
5 # futbol=nombre de la base de datos que hemos creado en MySQL
6 spring.datasource.url=jdbc:mysql://localhost:3306/futbol
7 # nombre de usuario y contraseña
8 spring.datasource.username=futbol
9 spring.datasource.password=futbolSimarro
10
11 #En H2 utilizabamos el dialecto H2Dialect, aquí MySQL8Dialect
12 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
13 #Para evitar que nombreLargo lo mapee como nombre largo y no lo encuentre
14 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
15
16 # actualizar base de datos y crear entidades
17 spring.jpa.hibernate.ddl-auto=update
18
19 #CONFIGURACIÓN SOLO durante las pruebas:
20 # Habilitar estadísticas hibernate
21 spring.jpa.properties.hibernate.generate_statistics=true
22 # Habilitar LOGGER de las estadísticas de hibernate
23 logging.level.org.hibernate.stat=debug
24 # Mostrar que consultas esta realizando Hibernate
25 spring.jpa.show-sql=true
26 # Formatear las consultas
27 spring.jpa.properties.hibernate.format_sql=true
28 # Mostrar los parametros que estan enviandose a las consultas
29 logging.level.org.hibernate.type=debug
30 #FIN CONFIGURACIÓN SOLO durante las pruebas
```

UD 3: Bases de datos y servicios REST

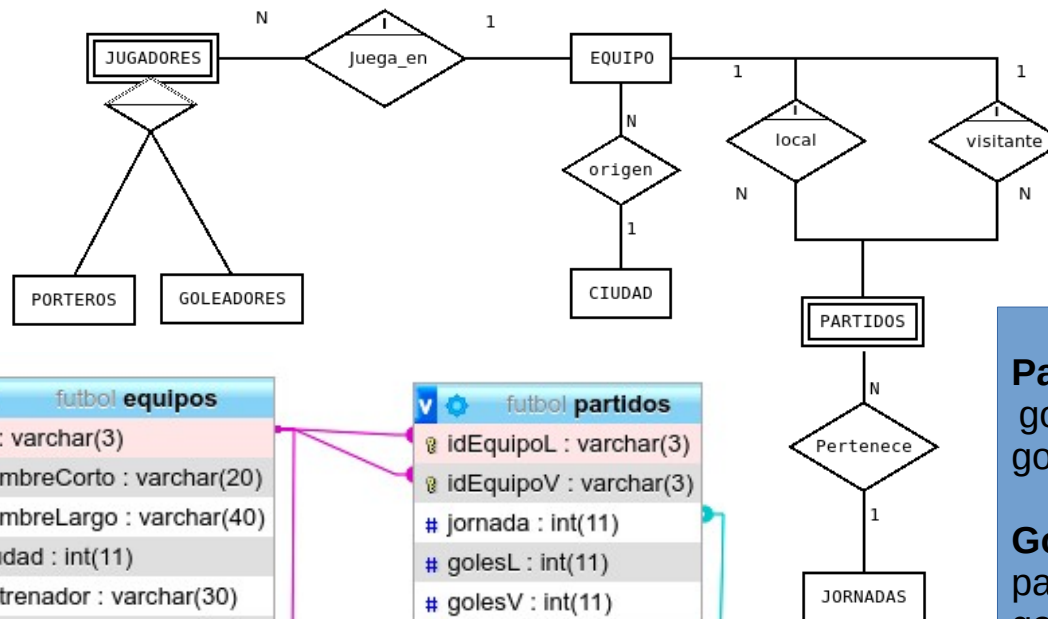
6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com



3º Crear las entidades JPA:

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



id	nombre	habitantes
int(11)	varchar(30)	int(11)

id	nombreCorto	nombreLargo	ciudad	entrenador	estadio	marca	patrocinador	presupuesto
varchar(3)	varchar(20)	varchar(40)	int(11)	varchar(30)	varchar(30)	varchar(30)	varchar(30)	int(11)

idEquipoL	idEquipoV	jornada	golesL	golesV	posesionL
varchar(3)	varchar(3)	int(11)	int(11)	int(11)	int(11)

num	fecha
int(11)	date

idEquipo	dorsal	partidos	goles	penaltis	pp	minutosGol	gTitular	gSuplente	gPuntos	gVictoria	gRemontada	porcentaje
varchar(3)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)	int(11)

idEquipo	dorsal	nombre	posicion	suelo
varchar(3)	int(11)	varchar(30)	varchar(10)	int(11)

idEquipo	dorsal	partidos	goles
varchar(3)	int(11)	int(11)	int(11)

Aclaraciones:

Partidos:

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

Goleadores:

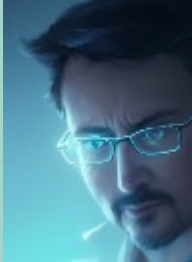
partidos= partidos jugados;
goles= goles marcados;
pp= goles en propia puerta;
minutosGol= minutos para marcar un gol;
gTitular= goles como titular;
gPuntos= goles que dieron puntos;
gVictoria= goles que dieron la victoria;
Porcentaje= porcentaje de goles del equipo;



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:

The screenshot shows an IDE with two panes. The left pane, titled 'EXPLORADOR', displays the project structure. The right pane, titled 'CiudadDb.java', shows the Java code for the entity.

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > model > db > J CiudadDb.java > ...
1  package edu.profesor.joseramon.api_rest_mysql_futbol.model.db;
2
3
4  import java.io.Serializable;
5  import javax.persistence.Column;
6  import javax.persistence.Entity;
7  import javax.persistence.Id;
8  import javax.persistence.Table;
9  import javax.validation.constraints.Size;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import lombok.AllArgsConstructor;
13 import lombok.Data;
14 import lombok.NoArgsConstructor;
15
16 @NoArgsConstructor
17 @AllArgsConstructor
18 @Data
19 @Entity
20 @Table(name = "ciudades")
21 public class CiudadDb implements Serializable{
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     private Long id;
25     @Size(min=4,message="El nombre debe de tener un tamaño mínimo de 4 caracteres")
26     private String nombre;
27     @Column(nullable = true)
28     private Long habitantes;
29 }
```

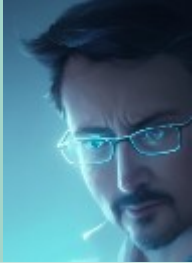
Ya podemos comprobar si la tabla esta bien enlazada contra MySQL haciendo un `clean` y un `install` con maven. Si no lanza error ¡¡Enhorabuena!!



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”:

```
J CiudadRepository.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > repository > J CiudadRepository.java > ...
1  package edu.profesor.joseramon.api_rest_mysql_futbol.repository;
2
3
4  import java.util.List;
5
6  import org.springframework.data.domain.Page;
7  import org.springframework.data.domain.Pageable;
8  import org.springframework.data.domain.Sort;
9  import org.springframework.data.jpa.repository.JpaRepository;
10 import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.CiudadDb;
11
12 public interface CiudadRepository extends JpaRepository<CiudadDb,Long>{
13     //Métodos automáticos en Spring Data JPA
14     List<CiudadDb> findAll(Sort sort);
15     //Filtrado por nombre usando equivalente a 'LIKE' usando "Containing"
16     List<CiudadDb> findByNombreContaining(String nombre,Sort sort);
17     //Paginación
18     Page<CiudadDb> findAll(Pageable pageable);
19     Page<CiudadDb> findByNombreContaining(String nombre,Pageable pageable);
20 }
```

¡¡Fijate que salvo modificar “application.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!!



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

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



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:

The screenshot shows an IDE with two main panels. The left panel, titled 'EXPLORADOR', displays the project structure. The right panel shows the code for 'CiudadService.java'.

Project Structure (EXPLORADOR):

- EDITORES ABIERTOS
 - CiudadService.java
- API_REST_MYSQL_FUTBOL
 - .mvn
 - .vscode
 - src
 - main
 - java/edu/profesor/joseramon/api_rest_mysql_futbol
 - model
 - db
 - CiudadDb.java
 - JornadaDb.java
 - dto
 - CiudadInfo.java
 - CiudadList.java
 - PaginaDto.java
 - repository
 - CiudadRepository.java
 - srv
 - CiudadService.java
 - resources
 - static
 - templates
 - application.properties

CiudadService.java Code:

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > J CiudadService.java > ...
1  package edu.profesor.joseramon.api_rest_mysql_futbol.srv;
2  import java.util.List;
3  import java.util.Optional;
4
5  import org.springframework.data.domain.Pageable;
6  import org.springframework.data.domain.Sort;
7
8  import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadInfo;
9  import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadList;
10 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.PaginaDto;
11
12 public interface CiudadService {
13     public Optional<CiudadInfo> getCiudadInfoById(Long id);
14     public List<CiudadList> findAllCiudadList();
15     public List<CiudadList> findAllCiudadList(Sort sort);
16     public List<CiudadList> findByNombreContainingCiudadList(String nombre, Sort sort);
17     public PaginaDto<CiudadList> findAllPageCiudadList(Pageable paging);
18     public PaginaDto<CiudadList> findByNombreContaining(String nombre, Pageable paging);
19 }
20
```



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:

Mappers:

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

The screenshot shows the VS Code interface. On the left, the Explorer view displays the project structure: `src/main/java/edu/profesor/joseramon/api_rest_mysql_futbol`. The `mapper` package is expanded, showing `CiudadMapper.java` and `CiudadService.java`. The main editor shows the content of `CiudadMapper.java`:

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > mapper > J CiudadMapper.java > ...
1 package edu.profesor.joseramon.api_rest_mysql_futbol.srv.mapper;
2
3
4 import java.util.List;
5
6 import org.mapstruct.Mapper;
7 import org.mapstruct.factory.Mappers;
8
9 import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.CiudadDb;
10 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadInfo;
11 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadList;
12
13 @Mapper
14 public interface CiudadMapper {
15     CiudadMapper INSTANCE= Mappers.getMapper(clazz: CiudadMapper.class);
16
17     CiudadInfo CiudadDbToCiudadInfo(CiudadDb ciudadDb);
18     CiudadList CiudadDbToCiudadList(CiudadDb ciudadDb);
19     List<CiudadList> ciudadesToCiudadList(List<CiudadDb> ciudadesDb);
20 }
```

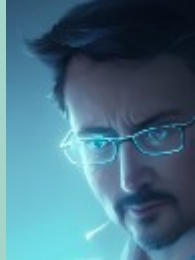
NOTA: No se han implementado todos los métodos (guardar, modificar,...) porque en esta practica se pretende analizar como se puede realizar consultas con más de una tabla.



UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

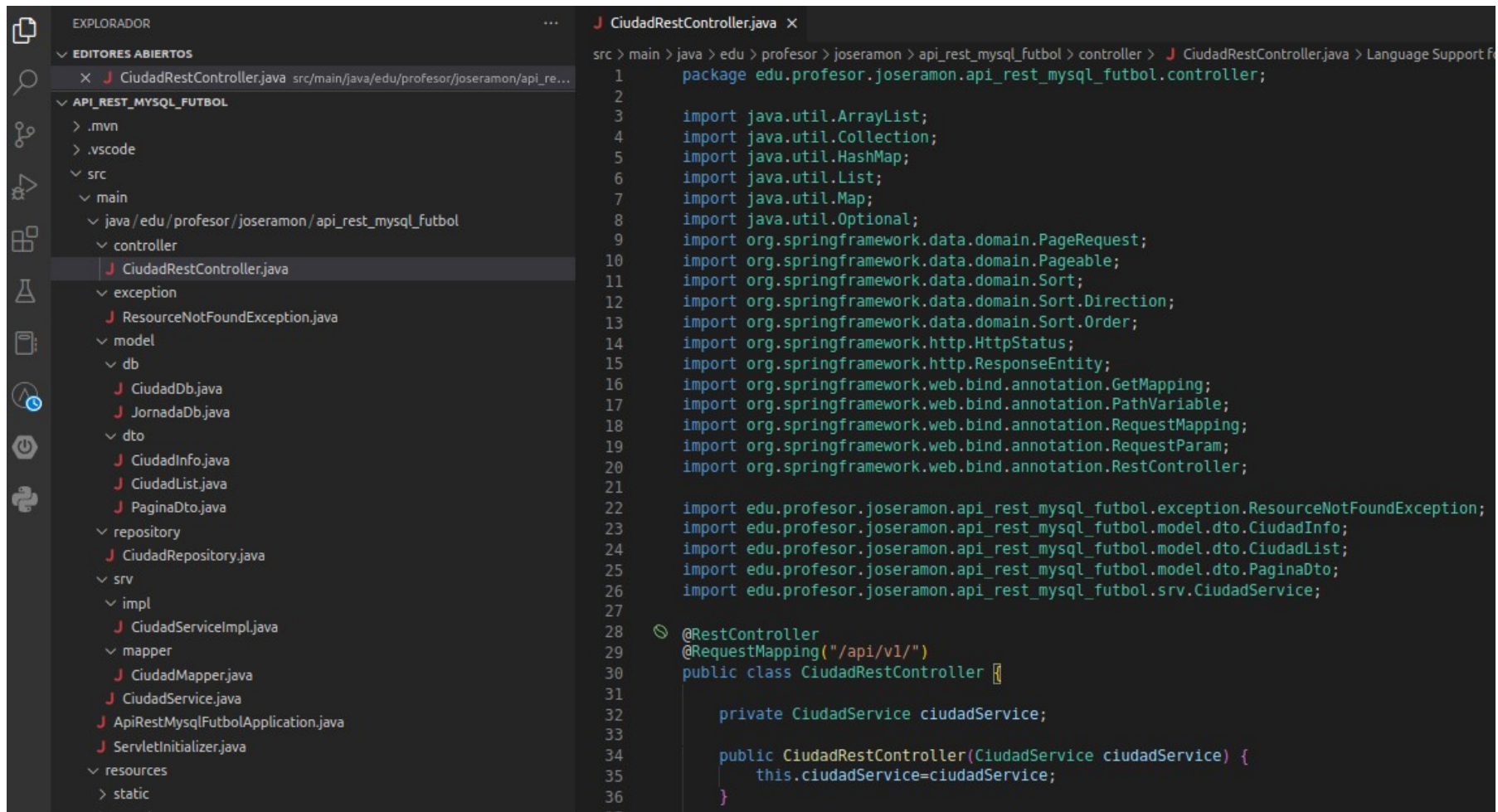
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



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

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

```
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > impl > J CiudadServiceImpl.java > ...
1  package edu.profesor.joseramon.api_rest_mysql_futbol.srv.impl;
2
3  import java.util.List;
4  import java.util.Optional;
5  import org.springframework.data.domain.Page;
6  import org.springframework.data.domain.Pageable;
7  import org.springframework.data.domain.Sort;
8  import org.springframework.stereotype.Service;
9  import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.CiudadDb;
10 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadInfo;
11 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.CiudadList;
12 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.PaginaDto;
13 import edu.profesor.joseramon.api_rest_mysql_futbol.repository.CiudadRepository;
14 import edu.profesor.joseramon.api_rest_mysql_futbol.srv.CiudadService;
15 import edu.profesor.joseramon.api_rest_mysql_futbol.srv.mapper.CiudadMapper;
16
17 @Service
18 public class CiudadServiceImpl implements CiudadService{
19     private final CiudadRepository ciudadRepository;
20
21     public CiudadServiceImpl(CiudadRepository ciudadRepository){
22         this.ciudadRepository=ciudadRepository;
23     }
24
25     public List<CiudadList> findAllCiudadList(){
26         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll());
27     }
28
29     public List<CiudadList> findAllCiudadList(Sort sort){
30         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll(sort));
31     }
32
33     public PaginaDto<CiudadList> findAllPageCiudadList(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



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:

```

src > main > resources > application.properties
1  #Cambiamos puerto para evitar solape con Xampp
2  server.port=8090
3
4  # url a la base de datos MySQL
5  # futbol=nombre de la base de datos que hemos creado en MySQL
6  spring.datasource.url=jdbc:mysql://localhost:3306/futbol
7  # nombre de usuario y contraseña
8  spring.datasource.username=futbol
9  spring.datasource.password=futbolSimarro
  
```

The screenshot shows the VS Code interface with the REST Client extension. The Explorer on the left shows the project structure with folders for DTO, Repository, Service, and Mapper. The main editor shows the test_Rest.http file with several GET requests. The right sidebar shows the response of the first request, which is a JSON array of city data.

```

12  ##
13  ##  getCiudadesByNombreContainingListOrderByName
14  GET http://localhost:8090/api/v1/ciudades/nombre/Val/orden/
15  Content-Type: application/json
16  ##
17  ##  getAllCiudades
18  GET http://localhost:8090/api/v1/ciudades HTTP/1.1
19  Content-Type: application/json
20  ##
21  ##  getAllCiudades
22  GET http://localhost:8090/api/v1/ciudades?size=5 HTTP/1.1
23  Content-Type: application/json
24  ##
25  ##  getAllCiudades
26  GET http://localhost:8090/api/v1/ciudades?sort=nombre,desc
27  Content-Type: application/json
  
```

The response of the first request is:

```

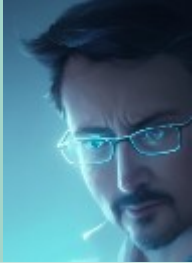
1  HTTP/1.1 200
2  Content-Type: application/json
3  Transfer-Encoding: chunked
4  Date: Sun, 29 Jan 2023 18:55:34 GMT
5  Connection: close
6
7  [
8  {
9    "id": 57,
10   "nombre": "Valladolid",
11   "habitantes": 313000
12 },
13 {
14   "id": 56,
15   "nombre": "València",
16   "habitantes": 798000
17 }
18 ]
  
```




UD 3: Bases de datos y servicios REST

6.- Consultas relacionadas en Spring Data JPA

Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com



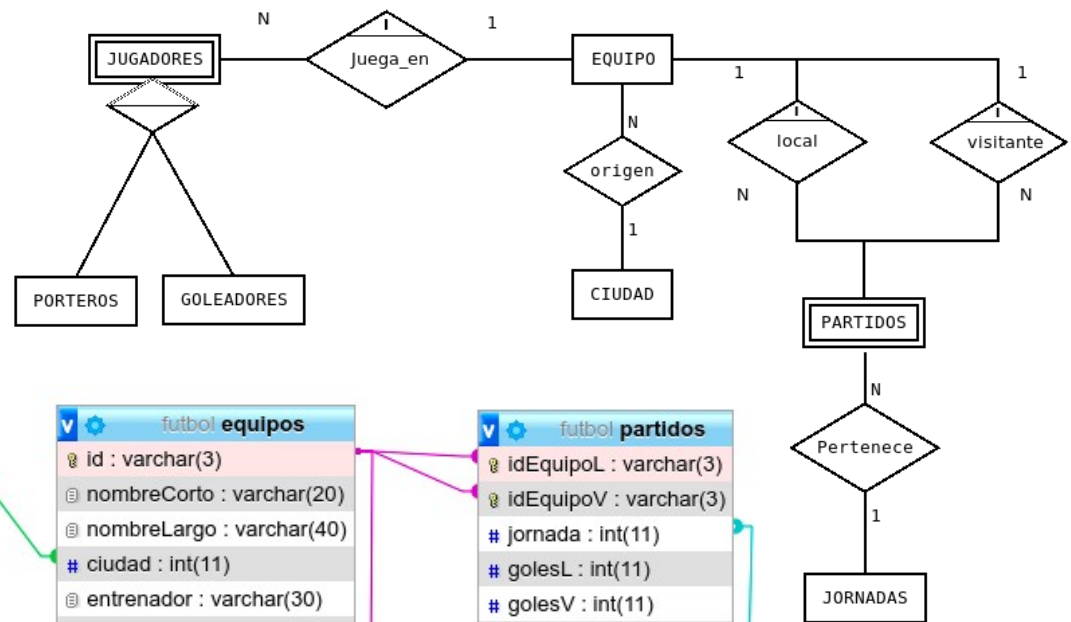
Tablas relacionadas:

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?



futbol ciudades
id : int(11)
nombre : varchar(30)
habitantes : int(11)

futbol goleadores
idEquipo : varchar(3)
dorsal : int(11)
partidos : int(11)
goles : int(11)
penaltis : int(11)
pp : int(11)
minutosGol : int(11)
gTitular : int(11)
gSuplente : int(11)
gPuntos : int(11)
gVictoria : int(11)
gRemontada : int(11)
porcentaje : int(11)

futbol equipos
id : varchar(3)
nombreCorto : varchar(20)
nombreLargo : varchar(40)
ciudad : int(11)
entrenador : varchar(30)
estadio : varchar(30)
marca : varchar(30)
patrocinador : varchar(30)
presupuesto : int(11)

futbol jugadores
idEquipo : varchar(3)
dorsal : int(11)
nombre : varchar(30)
posicion : varchar(10)
sueldo : int(11)

futbol partidos
idEquipoL : varchar(3)
idEquipoV : varchar(3)
jornada : int(11)
golesL : int(11)
golesV : int(11)
posesionL : int(11)

futbol jornadas
num : int(11)
fecha : date

futbol porteros
idEquipo : varchar(3)
dorsal : int(11)
partidos : int(11)
goles : int(11)



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:

The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a package named 'db' containing 'EquipoDb.java'. The code editor shows the following Java code:

```
14
15 @NoArgsConstructor
16 @AllArgsConstructor
17 @Data
18 @Entity
19 @Table(name = "equipos")
20 public class EquipoDb implements Serializable{
21     private static final long serialVersionUID = -818542778373595260L;
22     @Id
23     @Size(min=3,message="El id tiene un tamaño mínimo de 3")
24     private String id;
25     @Size(min=3,max=20,message="El nombre corto debe de tener un tamaño entre 3 y 20 caracteres")
26     @Column(name = "nombreCorto")
27     private String nombreCorto;
28     @Size(min=10,max=40,message="El nombre largo debe de tener un tamaño entre 10 y 40 caracteres")
29     @Column(name = "nombreLargo")
30     private String nombreLargo;
31     private Long ciudad;
32     @Size(min=10,max=30,message="El nombre del entrenador largo debe de tener un tamaño entre 10 y 30 caracteres")
33     private String entrenador;
34     @Size(min=10,max=30,message="El nombre del estadio largo debe de tener un tamaño entre 10 y 30 caracteres")
35     private String estadio;
36     @Size(min=4,max=30,message="El nombre de la marca debe de tener un tamaño entre 4 y 30 caracteres")
37     private String marca;
38     @Size(min=4,max=30,message="El nombre del patrocinador tener un tamaño entre 4 y 30 caracteres")
39     private String patrocinador;
40     private Long presupuesto;
41
42 }
```



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 línea 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
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 = -818542778373595260L;
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 - Jose Ramon - joseramon.profesor@gmail.com



Tablas relacionadas: @ManyToMany

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.

```
J EquipoRepository.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > repository > J 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:

```
J EquipoList.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > model > dto > J 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:

```
J EquipoMapper.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > mapper > J EquipoMapper.java
6  import org.mapstruct.Mapper;
7  import org.mapstruct.Mapping;
8  import org.mapstruct.factory.Mappers;
9
10 import edu.profesor.joseramon.api_rest_mysql_futbol.model.db.EquipoDb;
11 import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.EquipoList;
12
13 @Mapper
14 public interface EquipoMapper {
15     EquipoMapper INSTANCE= Mappers.getMapper(clazz: EquipoMapper.class);
16
17     @Mapping(target = "nombreCiudad", source = "ciudadDb.nombre")
18     EquipoList EquipoDbToEquipoList(EquipoDb equipoDb);
19
20     List<EquipoList> equiposDbToEquiposList(List<EquipoDb> equiposDb);
21
22 }
23
```

```
J EquipoServiceImpl.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > impl > J EquipoServiceImpl.java > ...
9  import org.springframework.data.domain.Page;
10 import org.springframework.data.domain.Pageable;
11 import org.springframework.stereotype.Service;
12
13 @Service
14 public class EquipoServiceImpl implements EquipoService{
15     private final EquipoRepository equipoRepository;
16
17     public EquipoServiceImpl(EquipoRepository equipoRepository){
18         this.equipoRepository=equipoRepository;
19     }
20
21 @Override
22 public PaginaDto<EquipoList> findAll(Pageable paging) {
23     Page<EquipoDb> paginaEquipoDb=equipoRepository.findAll(paging);
24     return new PaginaDto<EquipoList>(
25         paginaEquipoDb.getNumber(), //número de página solicitada
26         paginaEquipoDb.getSize(), //tamaño de la página
27         paginaEquipoDb.getTotalElements(), //total de elementos devueltos por la consulta
28         paginaEquipoDb.getTotalPages(), //total páginas teniendo en cuenta el tamaño de la consulta
29         EquipoMapper.INSTANCE.equiposDbToEquiposList(paginaEquipoDb.getContent()), //convertir los equipos a lista
30         paginaEquipoDb.getSort()); //ordenación de la consulta
31 }
32
```

```
J EquipoService.java x
src > main > java > edu > profesor > joseramon > api_rest_mysql_futbol > srv > J EquipoService.java > ...
1  package edu.profesor.joseramon.api_rest_mysql_futbol.srv;
2
3  import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.EquipoList;
4  import edu.profesor.joseramon.api_rest_mysql_futbol.model.dto.PaginaDto;
5  import org.springframework.data.domain.Pageable;
6
7  public interface EquipoService {
8      public PaginaDto<EquipoList> findAll(Pageable paging);
9  }
```




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:

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

35  @GetMapping("/equipos")
36  public ResponseEntity<Map<String, Object>> getAllEquipos(
37      @RequestParam(defaultValue = "0") int page,
38      @RequestParam(defaultValue = "3") int size,
39      @RequestParam(defaultValue = "id,asc") String[] sort) {
40
41      try {
42          // Crear sorts (ordenación de los datos)
43          List<Order> criteriosOrdenacion= new ArrayList<Order>();
44          //El primer criterio de ordenación siempre deberá de contener el orden(asc,desc)
45          if(sort[0].contains(",")){ //Hay más de un criterio de ordenación
46              //Tenemos un vector de ordenaciones en 'sort' y debemos leerlos
47              for (String criterioOrdenacion : sort) {
48                  String[] orden = criterioOrdenacion.split(",");
49                  if (orden.length > 1)
50                      criteriosOrdenacion.add(new Order(Direction.fromString(orden[1]), orden[0]));
51                  else // por defecto asc si no se dice nada
52                      criteriosOrdenacion.add(new Order(Direction.fromString(value: "asc"), orden[0]));
53              }
54          } else{ //Solo hay un criterio de ordenación
55              //El primer elemento del vector de sort es la dirección y el segundo el campo
56              criteriosOrdenacion.add(new Order(Direction.fromString(sort[1]), sort[0]));
57          }
58          Sort sorts= Sort.by(criteriosOrdenacion);
59
60          // Crear solicitud de página nº 'page' de tamaño 'size'
61          // utilizando el orden 'sort'
62          Pageable paging = PageRequest.of(page, size, sorts);
63
64          PaginaDto<EquipoList> paginaEquiposList=equipoService.findAll(paging);
65
66          // Rellenar datos a devolver en el servicio REST
67          List<EquipoList> equipos = paginaEquiposList.getContent();
68          Map<String, Object> response = new HashMap<>();
69          response.put("data", equipos);
70          response.put("currentPage", paginaEquiposList.getNumber());
71          response.put("pageSize", paginaEquiposList.getSize());
72          response.put("totalItems", paginaEquiposList.getTotalElements());
73          response.put("totalPages", paginaEquiposList.getTotalPages());
74          return new ResponseEntity<>(response, HttpStatus.OK);
75      } catch (Exception e) { // Si hay cualquier tipo de error
76          return new ResponseEntity<>(headers: null, HttpStatus.INTERNAL_SERVER_ERROR);
77      }
78  }
79 }
```




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: @ManyToMany

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

```
test_Rest.http x
test_Rest.http > ...
24 ###
25 ## getAllCiudades
26 Send Request
27 GET http://localhost:8090/api/v1/ciudades?sort=nombre,des
28 Content-Type: application/json
29 ###
30 ## getAllCiudades
31 Send Request
32 GET http://localhost:8090/api/v1/ciudades?nombre=Val&page
33 Content-Type: application/json
34 ###
35 ## getAllCiudades
36 Send Request
37 GET http://localhost:8090/api/v1/ciudades?sort=nombre,asc
38 Content-Type: application/json
39 ###
40 ## getAllCiudades
41 Send Request
42 GET http://localhost:8090/api/v1/ciudades?sort=habitantes
43 Content-Type: application/json
44 ###
45 ## getAllEquipos
46 Send Request
47 GET http://localhost:8090/api/v1/equipos HTTP/1.1
48 Content-Type: application/json
49
50
51
```

```
1 HTTP/1.1 200
2 Content-Type: application/json
3 Transfer-Encoding: chunked
4 Date: Mon, 30 Jan 2023 10:05:13 GMT
5 Connection: close
6
7 {
8   "totalItems": 20,
9   "data": [
10    {
11      "id": "ath",
12      "nombreLargo": "Athletic Club",
13      "nombreCiudad": "Bilbao"
14    },
15    {
16      "id": "atm",
17      "nombreLargo": "Club Atlético de Madrid",
18      "nombreCiudad": "Madrid"
19    },
20    {
21      "id": "bar",
22      "nombreLargo": "Futbol Club Barcelona",
23      "nombreCiudad": "Barcelona"
24    }
25  ],
26   "totalPages": 7,
27   "pageSize": 3,
28   "currentPage": 0
29 }
```



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.

