

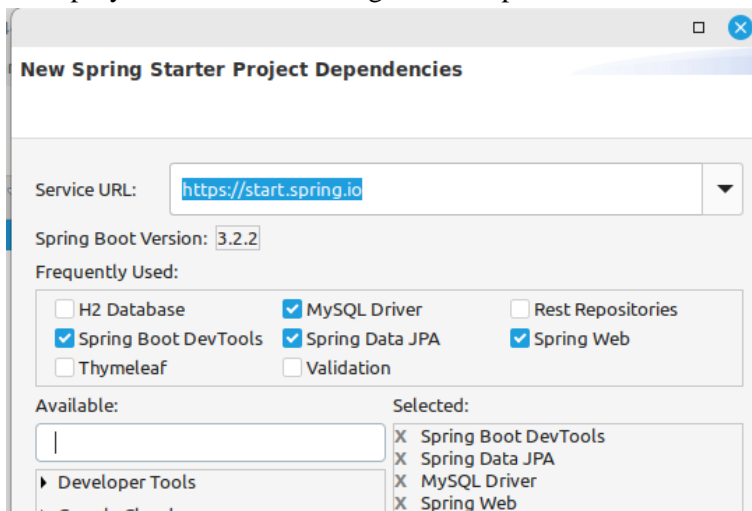
Tarea 1 – UD5

Empezando con Base de Datos

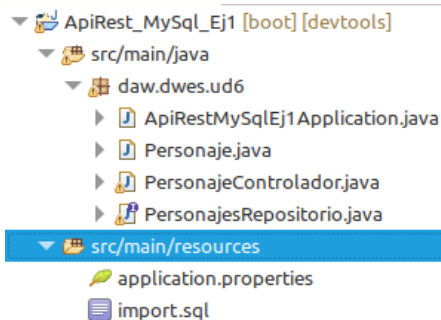
<https://github.com/AranchaC/D1-UD5.git>

Apartado 1.- API REST con MySql: cambiar la base de datos H2 por Mysql y así gozar de persistencia de los datos.

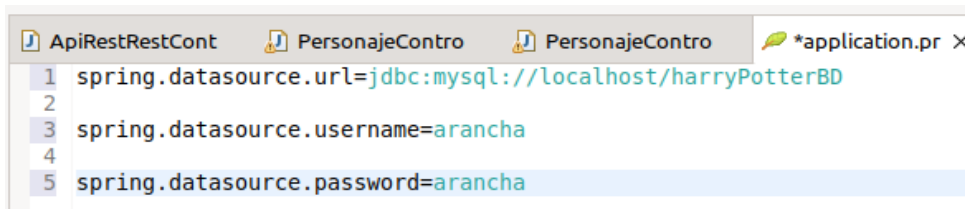
Creo proyecto nuevo con las siguientes dependencias:



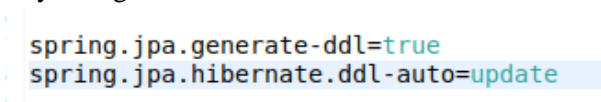
Copio los archivos .java y el imports.sql de la práctica anterior:



Añado los datos de la BBDD (url, usuario, password) en aplicaciones.properties:



Voy a elegir crear la tabla automáticamente con las siguientes líneas en applicatio.properties:



- **spring.jpa.generate-ddl=true:** Indica a Spring Boot que genere el DDL (Data Definition Language) automáticamente.
- **spring.jpa.hibernate.ddl-auto=update:** Le dice a Hibernate que actualice automáticamente la estructura de la base de datos según los cambios en las entidades de JPA.

Creo una base de datos y un usuario con permisos totales sobre ella para nuestra aplicación (con los mismos datos indicados en application.properties):

```
mysql> CREATE DATABASE harryPotterBD;
Query OK, 1 row affected (0,01 sec)

mysql> CREATE USER 'arancha'@'localhost' IDENTIFIED BY 'arancha';
Query OK, 0 rows affected (0,03 sec)

mysql> GRANT ALL PRIVILEGES ON harryPotterBD.* TO 'arancha'@'localhost';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'GRANT ALL PRIVILEGES ON harryPotterBD.* TO 'arancha'@'localhost'' at line 1
mysql> GRANT ALL PRIVILEGES ON harryPotterBD.* TO 'arancha'@'localhost';
Query OK, 0 rows affected (0,01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,01 sec)

mysql> exit;
Bye
```

Ejecuto la aplicación desde STS, y viendo que en consola no sale ningún error, visualizo en MySql que se ha autocreado la tabla en mi bbdd, con mi clase entidad como nombre de la tabla:

```
mysql> use harryPotterBD;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_harryPotterBD |
+-----+
| personaje                |
+-----+
1 row in set (0,00 sec)
```

Pero si accedemos a la tabla, está vacía por lo que vemos que no se ha cargado el import.sql:

```
mysql> show tables;
+-----+
| Tables_in_harryPotterBD |
+-----+
| personaje                |
+-----+
1 row in set (0,00 sec)

mysql> select * from personaje;
Empty set (0,00 sec)
```

Me doy cuenta de que en mi archivo imports.sql tengo que cada insert se añadan a una tabla *personajes* y que el código crea una tabla *personaje*.

Por lo que modifico el imports.sql cambiando de personajes a personaje:

```
import.sql x
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('HarryPotter', 'estudiante')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('RonWeasley', 'estudiante')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('HermioneGranger', 'estudiante')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('DracoMalfoy', 'estudiante')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('MinervaMcGonagall', 'prof')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('LunaLovegood', 'estudiant')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('SeverusSnape', 'profesor')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('GinnyWeasley', 'estudiant')
INSERT INTO personaje (nombre, rol, casa, ascendencia) VALUES ('NevilleLongbottom', 'estu
```

Ejecuto y también sale error. Gracias a los compañeros, me dicen que en la línea de `spring.jpa.hibernate.ddl-auto=create` de `application.properties`, cambie `update` por `create`.

```
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=create
```

Borro la anterior tabla *personaje*, y ejecuto de nuevo la aplicación y ahora sí que se han cargado los inserts:

```
mysql> drop table personaje;
Query OK, 0 rows affected (0,01 sec)

mysql> show tables;
Empty set (0,00 sec)

mysql> show tables;
+-----+
| Tables_in_harryPotterBD |
+-----+
| personaje                |
+-----+
1 row in set (0,00 sec)

mysql> select * from personaje;
+----+-----+-----+-----+-----+
| id | ascendencia | casa      | nombre           | rol      |
+----+-----+-----+-----+-----+
| 1  | Mestiza     | Gryffindor | HarryPotter      | estudiante |
| 2  | Sangre pura | Gryffindor | RonWeasley       | estudiante |
| 3  | Muggle     | Gryffindor | HermioneGranger  | estudiante |
| 4  | Sangre pura | Slytherin  | DracoMalfoy      | estudiante |
| 5  | Mestiza     | Gryffindor | MinervaMcGonagall | profesora |
| 6  | Sangre pura | Ravenclaw | LunaLovegood     | estudiante |
| 7  | Muggle     | Slytherin  | SeverusSnape     | profesor  |
| 8  | Sangre pura | Gryffindor | GinnyWeasley     | estudiante |
| 9  | Sangre pura | Gryffindor | NevilleLongbottom | estudiante |
| 10 | Sangre pura | Gryffindor | FredWeasley      | estudiante |
| 11 | Sangre pura | Gryffindor | GeorgeWeasley    | estudiante |
| 12 | Sangre pura | Ravenclaw | ChoChang         | estudiante |
| 13 | Sangre pura | Hufflepuff | CedricDiggory    | estudiante |
| 14 | Sangre pura | Gryffindor | SiriusBlack      | prisionero |
```

Voy a probar a insertar un nuevo personaje con Post `crearPersonaje`:

POST http://localhost:8080/personajes

Body: none form-data x-www-form-urlencoded raw binary JSON

```

1  {
2    "nombre": "AranchaChicharro",
3    "rol": "estudiante",
4    "casa": "Gryffindor",
5    "ascendencia": "Muggle"
6  }

```

Body Cookies Headers (5) Test Results Status: 201 Created Time: 125 ms Size: 267 B Save

Pretty Raw Preview Visualize

```

1  {
2    "id": 28,
3    "nombre": "AranchaChicharro",
4    "rol": "estudiante",
5    "casa": "Gryffindor",
6    "ascendencia": "Muggle"
7  }

```

arancho@daw2-01: ~

Archivo	Editar	Ver	Buscar	Terminal	Ayuda
15	Mestiza	Gryffindor	RemusLupin	profesor	
16	Sangre pura	Slytherin	BellatrixLestrange	mortífaga	
17	Sangre pura liberado	N/A	Dobby	elfo doméstico	
18	Mestiza	N/A	KingsleyShacklebolt	auror	
19	Sangre mestiza	Hufflepuff	NymphadoraTonks	auror	
20	Muggle	N/A	MadEyeMoody	auror	
21	Mestiza	Gryffindor	AlbusDumbledore	profesor	
22	Sangre pura	Ravenclaw	SybillTrelawney	profesora	
23	Sangre mestiza	Ravenclaw	FiliusFlitwick	profesor	
24	Mestiza	Slytherin	LordVoldemort	mortífago	
25	Sangre pura	Slytherin	LuciusMalfoy	mortífago	
26	Sangre pura	Slytherin	NarcissaMalfoy	mortífago	
27	Sangre mestiza	Gryffindor	PeterPettigrew	mortífago	
28	Muggle	Gryffindor	AranchaChicharro	estudiante	

Y si paro la aplicación y la vuelvo a ejecutar, este elemento nuevo no se guarda porque vuelve a crear la tabla con los datos anteriores, es decir, la tabla nueva pisa a la que ya estaba:

arancho@daw2-01: ~

Archivo	Editar	Ver	Buscar	Terminal	Ayuda
14	Sangre pura	Gryffindor	SiriusBlack	prisionero	
15	Mestiza	Gryffindor	RemusLupin	profesor	
16	Sangre pura	Slytherin	BellatrixLestrange	mortífaga	
17	Sangre pura liberado	N/A	Dobby	elfo doméstico	
18	Mestiza	N/A	KingsleyShacklebolt	auror	
19	Sangre mestiza	Hufflepuff	NymphadoraTonks	auror	
20	Muggle	N/A	MadEyeMoody	auror	
21	Mestiza	Gryffindor	AlbusDumbledore	profesor	
22	Sangre pura	Ravenclaw	SybillTrelawney	profesora	
23	Sangre mestiza	Ravenclaw	FiliusFlitwick	profesor	
24	Mestiza	Slytherin	LordVoldemort	mortífago	
25	Sangre pura	Slytherin	LuciusMalfoy	mortífago	
26	Sangre pura	Slytherin	NarcissaMalfoy	mortífago	
27	Sangre mestiza	Gryffindor	PeterPettigrew	mortífago	

27 rows in set (0,00 sec)

```

mysql>
32741 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enable
32741 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on o

```

Y entonces, ahora en application.properties cambio el *create* por *update*.

```

spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update

```

Voy a volver a crear dos nuevos personajes con post `crearPersonaje`:

- AranchaChicharro
- EvaChicharro

POST

http://localhost:8080/personajes

POST

http://localhost:8080/personajes

Params

Authorization

Headers (8)

Body

Params

Authorization

Headers (8)

Body

none

form-data

x-www-form-urlencoded

none

form-data

x-www-form-urlencoded

```

1 {
2   "nombre": "AranchaChicharro",
3   "rol": "etudiante",
4   "casa": "Gryffindor",
5   "ascendencia": "Muggle"
6 }

```

```

1 {
2   "nombre": "EvaChicharro",
3   "rol": "etudiante",
4   "casa": "Slytherin",
5   "ascendencia": "Muggle"
6 }

```

Body

Cookies

Headers (5)

Test Results

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

Pretty

Raw

Preview

Visualize

JSON

```

1 {
2   "id": 28,
3   "nombre": "AranchaChicharro",
4   "rol": "etudiante",
5   "casa": "Gryffindor",
6   "ascendencia": "Muggle"
7 }

```

```

1 {
2   "id": 29,
3   "nombre": "EvaChicharro",
4   "rol": "etudiante",
5   "casa": "Slytherin",
6   "ascendencia": "Muggle"
7 }

```

Compruebo en lista:

23	FiliusFlitwick	Ravenclaw	profesor	Sangre mestiza
24	LordVoldemort	Slytherin	mortífago	Mestiza
25	LuciusMalfoy	Slytherin	mortífago	Sangre pura
26	NarcissaMalfoy	Slytherin	mortífago	Sangre pura
27	PeterPettigrew	Gryffindor	mortífago	Sangre mestiza
28	AranchaChicharro	Gryffindor	etudiante	Muggle
29	EvaChicharro	Slytherin	etudiante	Muggle

29 rows in set (0,00 sec)

mysql>

Paro la ejecución de la api y la vuelvo a ejecutar, y vemos que ahora sí que se han guardado los personajes que he añadido nuevos:

21	AlbusDumbledore	Gryffindor	profesor	Mestiza
22	SybillTrelawney	Ravenclaw	profesora	Sangre pura
23	FiliusFlitwick	Ravenclaw	profesor	Sangre mestiza
24	LordVoldemort	Slytherin	mortífago	Mestiza
25	LuciusMalfoy	Slytherin	mortífago	Sangre pura
26	NarcissaMalfoy	Slytherin	mortífago	Sangre pura
27	PeterPettigrew	Gryffindor	mortífago	Sangre mestiza
28	AranchaChicharro	Gryffindor	etudiante	Muggle
29	EvaChicharro	Slytherin	etudiante	Muggle

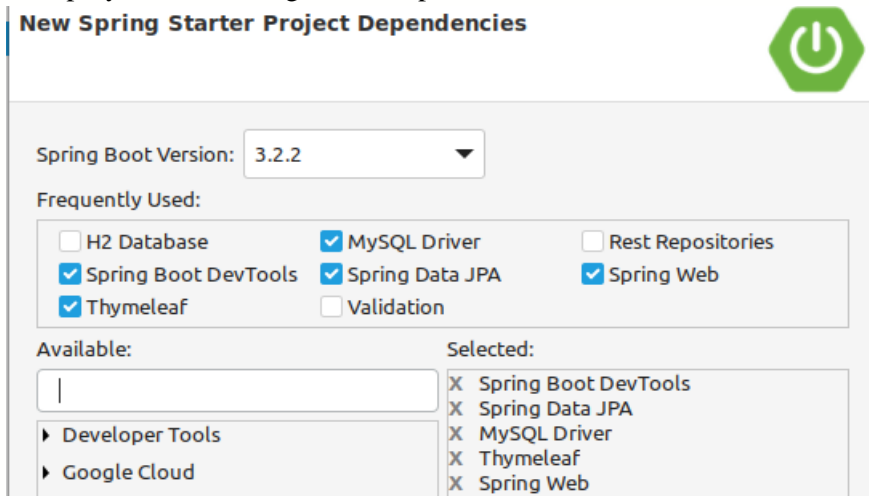
29 rows in set (0,00 sec)

34mysql>

60+01:00 INFO 49924 --- [restartedMain] o.a.c.c.c.[Tomcat].[/] : In

2.- Continuar el QUIZZ de la UD5 haciendo que las puntuaciones se guarden en base de datos.

Creo proyecto con las siguientes dependencias:



Primero voy a crear la bbdd para esta api y voy a aprovechar el usuario del ej anterior, pero tendré que darle permisos a esta bbdd nueva:

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE jugadoresQUIZZ_HP;
Query OK, 1 row affected (0,01 sec)

mysql> USE jugadoresQUIZZ_HP;
Database changed
mysql> GRANT ALL PRIVILEGES ON jugadoresQUIZZ_HP.* TO 'arancha'@'localhost';
Query OK, 0 rows affected (0,01 sec)

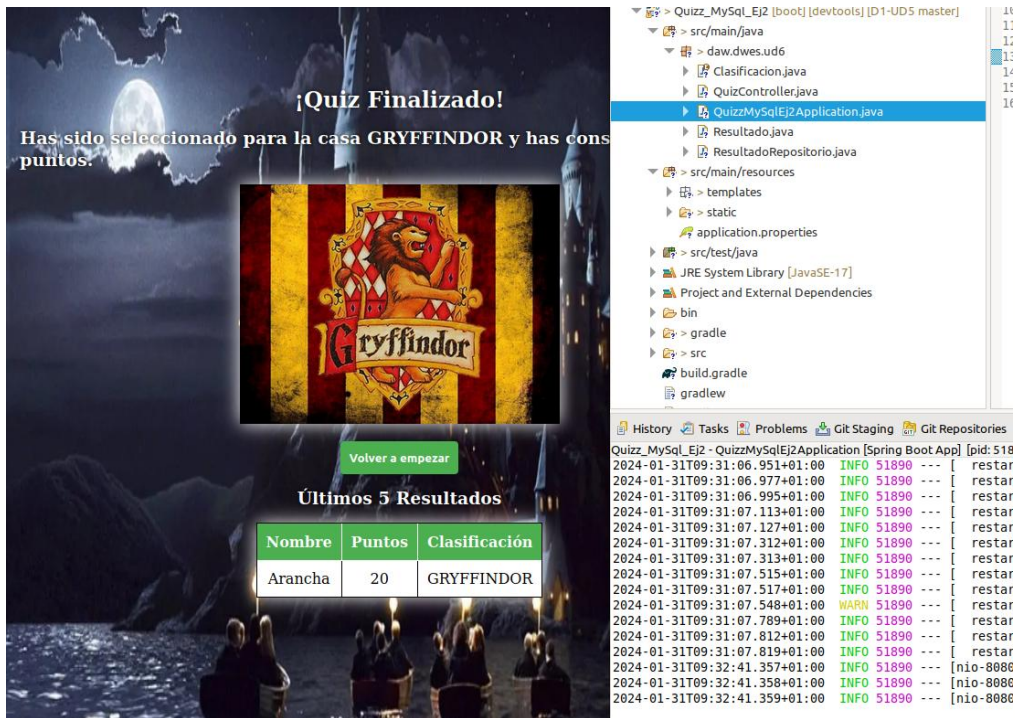
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,01 sec)

mysql>
```

Modifico el application.properties con los datos de esta bbdd, igual que en el ej1:

```
application.pro x Personaje.java application.pro ApiRestMySQLEj1
1 spring.datasource.url=jdbc:mysql://localhost/jugadoresQUIZZ_HP
2
3 spring.datasource.username=arancha
4 spring.datasource.password=arancha
5
6 spring.jpa.generate-ddl=true
7 spring.jpa.hibernate.ddl-auto=create
```

Copio los archivos del proyecto del quizz del tema anterior y ejecuto la api, asegurándome de que no hay ningún error en consola ni en el navegador:



Como en el caso de esta api, mi objeto entidad, *Resultado* no tiene atributo identificador, voy a crear un id automático, igual que hicimos en el reto anterior. También añado la anotación *@Entity* a la clase *Resultado*:

```

1  @Entity
2  public class Resultado {
3
4      private Clasificacion clasificacion;
5      private int puntos;
6      private String nombre;
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.IDENTITY)
10     private long id;
  
```

En mi clase *ResultadoRepositorio* la convierto en interface (creando clase nueva), y que extienda de *JpaRepository* sobre mi entidad y la clave, que será *Resultado* y *Long* (sobre esto me he fijado en ejemplo de ej anterior)

Y creo función *findByNombre* (que aún no se si será necesario):

```

application.properties ResultadoRepositorio.java X *QuizController.java Resultado.java
1  package daw.dwes.ud6;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4
5  public interface ResultadoRepositorio extends JpaRepository<Resultado, Long> {
6
7      Resultado findByNombre(String nombre);
8
9
10 }
  
```

Ahora voy al **Controlador**.

Cambio la anotación `@Controller` de la clase por `@RestController`. Me aseguro de que hay variable de tipo `ResultadoRepository`, y creo constructor con anotación `@Autowired` para iniciar esta variable:

```
@RestController
public class QuizController {

    private final ResultadoRepository resultadoRepository;

    @Autowired
    public QuizController(ResultadoRepository resultadoRepository) {
        this.resultadoRepository = resultadoRepository;
    }
}
```

Ahora tengo que cambiar la forma de guardar los datos y acceder a ellos, pues ahora vamos a usar la bbdd. Me dirijo al método de la última página, donde gestiono esta parte de lógica, y usando las funciones de `JpaRepository`, ahora aplicadas a mi `ResultadoRepository`.

```
@PostMapping("/paginaNombre")
public String paginaNombre(
    @RequestParam(name = "nombre") String nombre,
    HttpSession session,
    Model model) {

    // Obtener el objeto Resultado de la sesión
    Resultado resultado = obtenerResultado(session);

    // Actualizar el nombre en el objeto Resultado
    resultado.setNombre(nombre);
    // Agregar el objeto Resultado actualizado al modelo
    model.addAttribute("resultado", resultado);

    // Guardar el resultado en el repositorio con .save:
    resultadoRepository.save(resultado);

    // Obtener todos los resultados del repositorio con .findAll():
    List<Resultado> todosLosResultados = resultadoRepository.findAll();

    // Seleccionar los últimos 5 resultados (o menos si hay menos de 5)

    //si la lista tiene > 5 elementos, la posición es size-5, si no, es la pos:
    int inicio = todosLosResultados.size() > 5 ? todosLosResultados.size() - 5
    //la posición será la última de la lista.
    int fin = todosLosResultados.size();
    List<Resultado> ultimosResultados = todosLosResultados.subList(inicio, fin);


    // Agregar la lista de últimos resultados al modelo
    model.addAttribute("ultimosResultados", ultimosResultados);

    return "finalResultado";
}
```

- `resultadoRepository.save(resultado);` :: con esta función guardo el objeto/entidad resultado, que se compone de nombre, puntuación y clasificación en la bbdd.
- `List<Resultado> todosLosResultados = resultadoRepository.findAll();` :: creo una lista de todosLosResultados y le asigno todos los valores del repositorio mediante la función `.findAll()`.

Ahora que no tengo ningún error en el código, ejecuto la api y tras ver que no hay ningún error en la consola, pruebo en el navegador a hacer una vez el quiz y compruebo en MySQL que se ha creado la tabla en la bbdd correspondiente y que se ha introducido el jugador del quizz del navegador con su resultado:

¡Quiz Finalizado!
Has sido seleccionado para la casa RAVENCLAW y has conseguido 17 puntos.



[Volver a empezar](#)

Últimos 5 Resultados

Nombre	Puntos	Clasificación
rodri	17	RAVENCLAW

```
mysql> CREATE DATABASE jugadoresQUIZZ_HP;
Query OK, 1 row affected (0,01 sec)

mysql> USE jugadoresQUIZZ_HP;
Database changed

mysql> GRANT ALL PRIVILEGES ON jugadoresQUIZZ_HP.* TO 'root'@'localhost';
Query OK, 0 rows affected (0,01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,01 sec)


mysql> show tables;
+-----+
| Tables_in_jugadoresQUIZZ_HP |
+-----+
| resultado                    |
+-----+
1 row in set (0,00 sec)

mysql> select * from resultado;
+-----+-----+-----+-----+
| clasificacion | puntos | id | nombre |
+-----+-----+-----+-----+
| 1             | 17     | 1 | rodri  |
+-----+-----+-----+-----+
1 row in set (0,00 sec)

mysql>
```

En mi caso, la clasificación es de tipo enum y vemos que en sql aparece de tipo int. Voy a probar a poner más jugadores intentando que salga uno de cada clasificación:

¡Quiz Finalizado!
Has sido seleccionado para la casa SLYTHERIN y has conseguido 13 puntos.



[Volver a empezar](#)

Últimos 5 Resultados

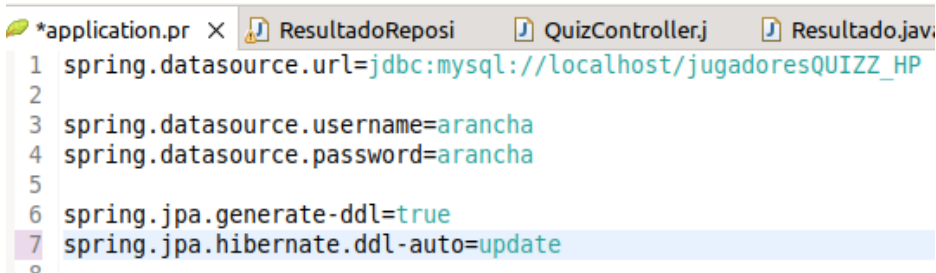
Nombre	Puntos	Clasificación
Rodri	20	GRYFFINDOR
Arancha	19	RAVENCLAW
Eva	7	HUFFLEPUFF
Pepito	13	SLYTHERIN

```
mysql> show tables;
+-----+
| Tables_in_jugadoresQUIZZ_HP |
+-----+
| resultado                    |
+-----+
1 row in set (0,00 sec)

mysql> select * from resultado;
+-----+-----+-----+-----+
| clasificacion | puntos | id | nombre |
+-----+-----+-----+-----+
| 0             | 20     | 1 | Rodri  |
| 1             | 19     | 2 | Arancha|
| 3             | 7      | 3 | Eva    |
| 2             | 13     | 4 | Pepito |
+-----+-----+-----+-----+
4 rows in set (0,00 sec)

mysql>
```

Ahora algo importante, para que se queden estos datos guardados en la bbdd cuando detengamos la api y la volvamos a ejecutar, en *application.properties* cambiamos la anotación **create** por **update**, pues si no, vuelve a hacer la acción de crear tabla y sustituiría a la que teníamos. Y al poner *update* en vez de *create*, la actualizamos:

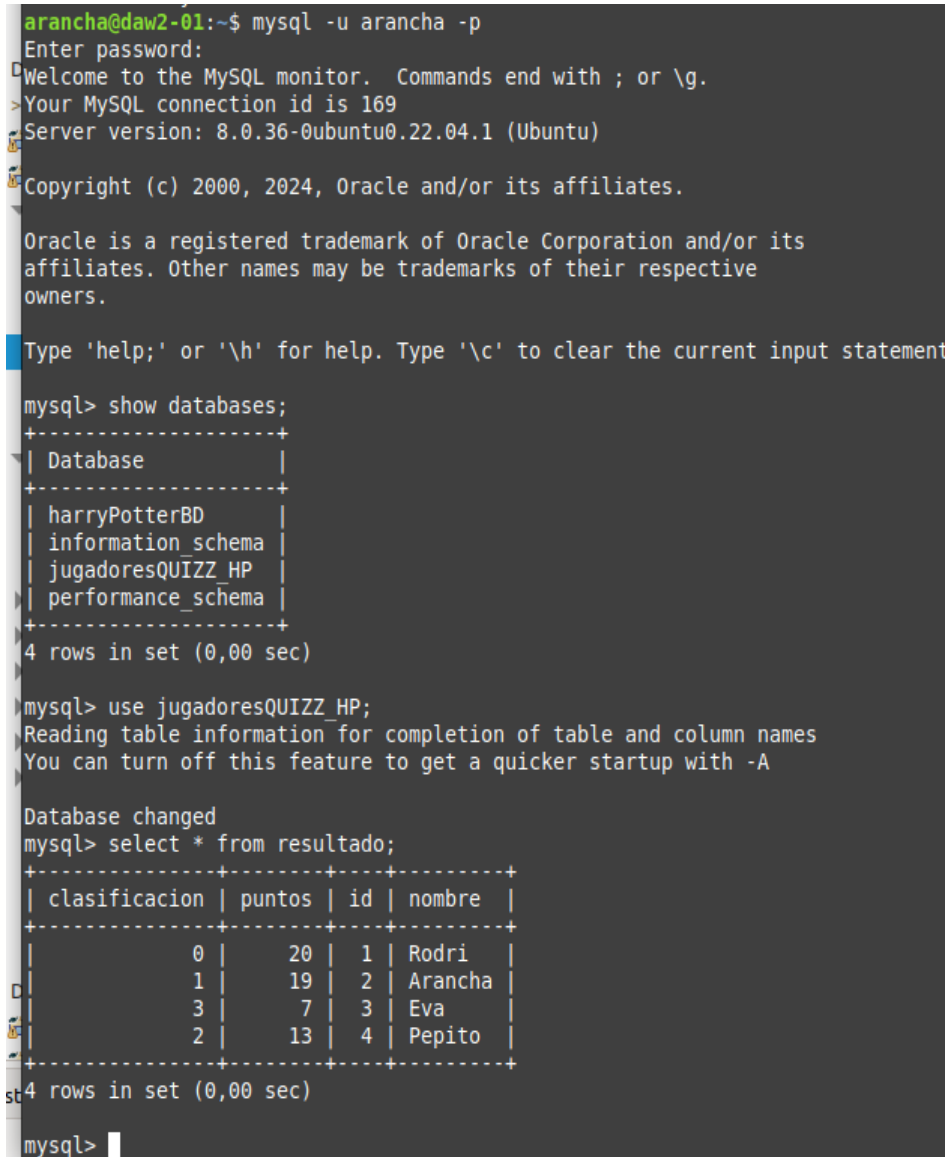


```

1 spring.datasource.url=jdbc:mysql://localhost/jugadoresQUIZZ_HP
2
3 spring.datasource.username=arancha
4 spring.datasource.password=arancha
5
6 spring.jpa.generate-ddl=true
7 spring.jpa.hibernate.ddl-auto=update

```

Ahora hago la prueba. Detengo la api y la vuelvo a ejecutar, y compruebo en MySQL (saliendo y entrando nuevamente) que ahora sí que tenemos los resultados anteriores guardados:



```

arancha@daw2-01:~$ mysql -u arancha -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
> Your MySQL connection id is 169
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

mysql> show databases;
+-----+
| Database |
+-----+
| harryPotterBD |
| information schema |
| jugadoresQUIZZ HP |
| performance_schema |
+-----+
4 rows in set (0,00 sec)

mysql> use jugadoresQUIZZ_HP;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

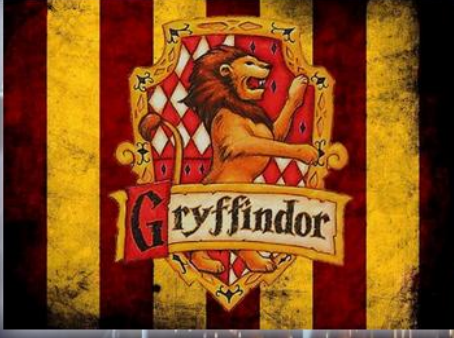
Database changed
mysql> select * from resultado;
+-----+-----+-----+-----+
| clasificacion | puntos | id | nombre |
+-----+-----+-----+-----+
| 0 | 20 | 1 | Rodri |
| 1 | 19 | 2 | Arancha |
| 3 | 7 | 3 | Eva |
| 2 | 13 | 4 | Pepito |
+-----+-----+-----+-----+
4 rows in set (0,00 sec)

mysql>

```

Introducimos más jugadores para verificar que se añaden a la tabla anterior:

Quiz Finalizado.
do seleccionado para la casa GRYFFINDOR
guido 27 puntos.



Volver a empezar

Últimos 5 Resultados

Nombre	Puntos	Clasificación
Eva	7	HUFFLEPUFF
Pepito	13	SLYTHERIN
Carol	19	RAVENCLAW
Sara	8	HUFFLEPUFF
Ana	27	GRYFFINDOR

```

performance schema |
+-----+
4 rows in set (0,00 sec)

mysql> use jugadoresQUIZZ_HP;
Reading table information for completion of
You can turn off this feature to get a quick

Database changed
mysql> select * from resultado;
+-----+
| clasificacion | puntos | id | nombre |
+-----+
| 0 | 20 | 1 | Rodri |
| 1 | 19 | 2 | Arancha |
| 3 | 7 | 3 | Eva |
| 2 | 13 | 4 | Pepito |
+-----+
4 rows in set (0,00 sec)

mysql> select * from resultado;
+-----+
| clasificacion | puntos | id | nombre |
+-----+
| 0 | 20 | 1 | Rodri |
| 1 | 19 | 2 | Arancha |
| 3 | 7 | 3 | Eva |
| 2 | 13 | 4 | Pepito |
| 1 | 19 | 5 | Carol |
| 3 | 8 | 6 | Sara |
| 0 | 27 | 7 | Ana |
+-----+
7 rows in set (0,00 sec)

mysql>
  
```

Al ver los materiales, he visto que existe la anotación `@Enumerated` que se usa para indicar que un tipo enumerado se almacene como String de lo contrario se almacena como ordinal, como ha pasado en mi tabla.

Voy incluir esta anotación en mi clase entidad *Resultado*, en el atributo *clasificacion*:

```

application.pro  ResultadoReposi  *Resultado.java  "2
8
9 @Entity
10 public class Resultado {
11
12 @Enumerated
13 private Clasificacion clasificacion;
14 private int puntos;
15 private String nombre;
16
17 @Id
18 @GeneratedValue(strategy = GenerationType.IDENTITY)
19 private Long id;
  
```

Ejecute la API de nuevo y compruebo los datos de la tabla en MySQL, pero no se actualiza, seguramente sea porque la columna ya esté creada y reconocida como int:

```

mysql> select id, nombre, puntos, clasificacion from resultado;
+-----+
| id | nombre | puntos | clasificacion |
+-----+
| 1 | Rodri | 20 | 0 |
| 2 | Arancha | 19 | 1 |
| 3 | Eva | 7 | 3 |
| 4 | Pepito | 13 | 2 |
| 5 | Carol | 19 | 1 |
| 6 | Sara | 8 | 3 |
| 7 | Ana | 27 | 0 |
| 8 | Ana | 22 | 0 |
+-----+
8 rows in set (0,00 sec)
  
```


Voy a probar a eliminar la tabla, poniendo en application.properties otra vez create:

```
4 spring.datasource.password=arancha
5
6 spring.jpa.generate-ddl=true
7 spring.jpa.hibernate.ddl-auto=create
```

Pero tampoco salía la clasificación como string, viendo otra vez los materiales hay un añadido a la anotación `@Enumerated` que es para especificar el tipo de dato enumerado. Habría que añadir `EnumType` y en mi caso `String`:

```
@Entity
public class Resultado {
    @Enumerated(EnumType.STRING)
    private Clasificacion clasificacion;
    private int puntos;
    private String nombre;
```

Y vuelvo a iniciar la api, con create en properties, y realizo el quiz varias veces para generar usuarios, y ahora sí que salen los valores de la columna clasificación como string. (*Importante! Volver a cambiar de create a update en properties*):



The image shows a MySQL terminal window and a web application interface. The terminal window displays the following SQL query and its results:

```
mysql> select id, nombre, puntos, clasificacion
+-----+-----+-----+-----+
| id | nombre | puntos | clasificacion |
+-----+-----+-----+-----+
| 1 | Arancha | 0 | NULL |
| 2 | Arancha | 20 | GRYFFINDOR |
| 3 | Rodri | 13 | SLYTHERIN |
| 4 | Eva | 18 | RAVENCLAW |
+-----+-----+-----+-----+
4 rows in set (0,01 sec)
```

The web application interface shows a button labeled "Volver a empezar" and a table titled "Últimos 5 Resultados". The table has three columns: "Nombre", "Puntos", and "Clasificación". The data in the table is as follows:

Nombre	Puntos	Clasificación
Arancha	0	
Arancha	20	GRYFFINDOR
Rodri	13	SLYTHERIN
Eva	18	RAVENCLAW

3.- Para el bien (dos de los tres puntos)

- **Documentar las pruebas de qué pasa si no existe la tabla y qué ajustes sirven para que se cree automáticamente y si da algún problema si ya está creada.**

Esta parte se ha ido documentando a medida que se ha hecho la documentación anterior con pruebas realizadas.

La clave está en el archivo `application.properties`. Si dejamos el estado de `generate-ddl` en `create`, cada vez que ejecutamos la api la tabla se vuelve a crear, sustituyendo la tabla anterior, si la hubiera, por una nueva y por lo tanto se borran los datos. Para que esto no pase, y se actualicen los datos, tenemos que dejar en `generate-ddl` en `update`.

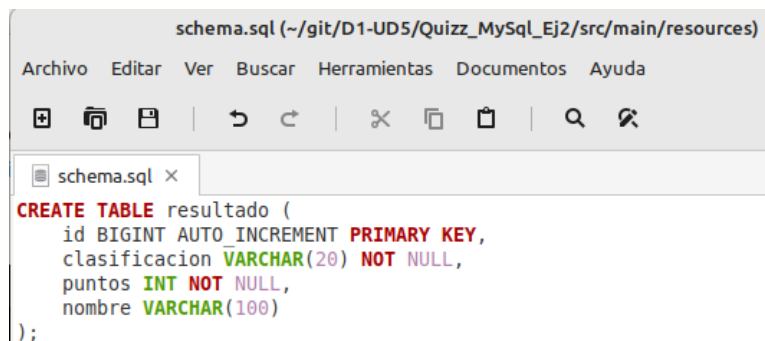
Con `create`, la tabla se crea automáticamente.

Ver la creación mediante archivo `schema.sql` y la carga inicial de datos de las distintas formas posibles: archivo `data` o `import.sql` si no existen.

Para hacer estas pruebas voy a borrar la tabla.

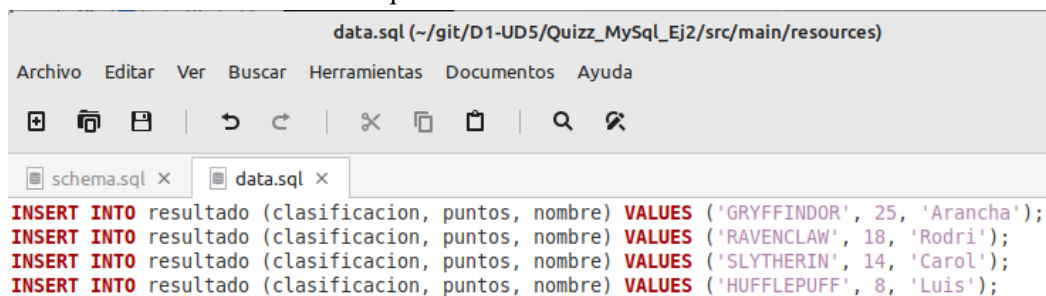
```
mysql> drop table resultado;  
Query OK, 0 rows affected (0,02 sec)
```

Y en mi proyecto, en la carpeta `resources` voy a crear archivo `schema.sql` con las líneas de sql para crear la tabla:



```
schema.sql (~/.git/D1-UD5/Quizz_MySql_Ej2/src/main/resources)  
Archivo Editar Ver Buscar Herramientas Documentos Ayuda  
schema.sql x  
CREATE TABLE resultado (  
  id BIGINT AUTO INCREMENT PRIMARY KEY,  
  clasificacion VARCHAR(20) NOT NULL,  
  puntos INT NOT NULL,  
  nombre VARCHAR(100)  
);
```

Y para cargar los datos iniciales, lo voy a hacer desde `data.sql`, pues este tipo de archivo genera las las consultas después de crear el esquema de la bbdd, y si lo hiciera desde `imports.sql`, ejecutaría las consultas antes de crear el esquema:



```
data.sql (~/.git/D1-UD5/Quizz_MySql_Ej2/src/main/resources)  
Archivo Editar Ver Buscar Herramientas Documentos Ayuda  
schema.sql x data.sql x  
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('GRYFFINDOR', 25, 'Arancha');  
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('RAVENCLAW', 18, 'Rodri');  
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('SLYTHERIN', 14, 'Carol');  
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('HUFFLEPUFF', 8, 'Luis');
```

En `properties`, en la línea de `hibernate.ddl-auto` pongo el estado en `none`:

```
spring.jpa.generate-ddl=true  
spring.jpa.hibernate.ddl-auto=none
```

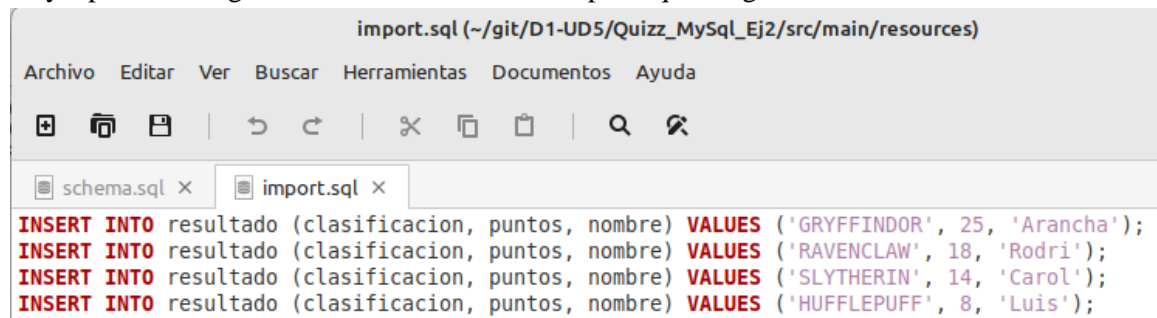

Y ejecutamos la api, se crea la tabla pero no se importan los datos:

```
mysql> select id, nombre, puntos, clasificacion from resultado;
Empty set (0,00 sec)

mysql> show tables;
+-----+
| Tables_in_jugadoresQUIZZ_HP |
+-----+
| resultado                    |
+-----+
1 row in set (0,00 sec)

mysql> select * from resultado;
Empty set (0,00 sec)
```

Voy a probar a cargar los datos con el archivo import.sql en lugar de data:



```
import.sql (~/git/D1-UD5/Quizz_MySql_Ej2/src/main/resources)

Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda

[+] [🔍] [📄] [🔄] [↶] [↷] [✂] [📋] [📄] [🔍] [🔧]

schema.sql x  import.sql x

INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('GRYFFINDOR', 25, 'Arancha');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('RAVENCLAW', 18, 'Rodri');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('SLYHERIN', 14, 'Carol');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('HUFFLEPUFF', 8, 'Luis');
```

Y vuelvo a empezar, borro la tabla de bbdd, detengo la api y la vuelvo a ejecutar. Y tampoco se cargan los datos, la tabla está vacía:

```
mysql> drop table resultado;
Query OK, 0 rows affected (0,01 sec)

mysql> show tables;
Empty set (0,00 sec)

mysql> show tables;
+-----+
| Tables_in_jugadoresQUIZZ_HP |
+-----+
| resultado                    |
+-----+
1 row in set (0,00 sec)

mysql> select * from resultado;
Empty set (0,00 sec)

mysql> select * from resultado;
Empty set (0,00 sec)

mysql>
```

Como solución, voy a poner los imports en el mismo schema.sql, seguidamente de la creación de la tabla:

```
*schema.sql (~/.git/D1-UD5/Quizz_MySql_Ej2/src/main/resources)
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda

*schema.sql x  import.sql x

CREATE TABLE resultado (
  id BIGINT AUTO INCREMENT PRIMARY KEY,
  clasificacion VARCHAR(20) NOT NULL,
  puntos INT NOT NULL,
  nombre VARCHAR(100)
);

INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('GRYFFINDOR', 25, 'Arancha');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('RAVENCLAW', 18, 'Rodri');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('SLYTHERIN', 14, 'Carol');
INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('HUFFLEPUFF', 8, 'Luis');
```

Y de nuevo, detengo la api, borro tabla de la bbdd y ejecuto api, y tampoco se añaden los datos:

```
mysql> drop table resultado;
Query OK, 0 rows affected (0,02 sec)

mysql> show tables;
+-----+
| Tables_in_jugadoresQUIZZ_HP |
+-----+
| resultado                    |
+-----+
1 row in set (0,01 sec)

mysql> select * from resultado;
Empty set (0,00 sec)

mysql>
```

Como verificación extra, accedo a la info de la tabla y se puede ver que la tabla se crea correctamente, con los tipos de datos correctos:

```
mysql> describe resultado;
+-----+
| Field          | Type                                                                 | Null | Key | Default | Extra          |
+-----+
| id             | bigint                                                             | NO   | PRI | NULL    | auto_increment |
| clasificacion  | enum('GRYFFINDOR','RAVENCLAW','SLYTHERIN','HUFFLEPUFF')           | YES  |     | NULL    |                |
| nombre         | varchar(255)                                                       | YES  |     | NULL    |                |
| puntos         | int                                                                | NO   |     | NULL    |                |
+-----+
4 rows in set (0,00 sec)

mysql>
```

Pero desde casa, al crear la bbdd desde ahí, pruebo algo diferente.

En application.properties la línea generate-ddl pongo el estado en **false**, pues esta línea le indica a Spring Boot que debe generar el esquema de la base de datos (DDL) en función de las entidades JPA detectadas en la aplicación. Cuando se establece en true, Spring Boot generará automáticamente el DDL necesario para crear las tablas de la base de datos en función de las clases de entidad JPA. Y si se establece en false, se tendrá que crear el esquema manualmente.

```
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=create
```

Por lo que vuelvo a la idea inicial, en el archivo **schema.sql** pongo las líneas sql para crear la tabla y en el archivo **import.sql** los inserts:

```

schema.sql
C: > Users > savia > git > D1-UD5 > Quizz_MySql_Ej2 > src > main > resources > schema.sql
1 CREATE TABLE resultado (
2     id BIGINT AUTO_INCREMENT PRIMARY KEY,
3     clasificacion VARCHAR(20) NOT NULL,
4     puntos INT NOT NULL,
5     nombre VARCHAR(100)
6 );

import.sql
C: > Users > savia > git > D1-UD5 > Quizz_MySql_Ej2 > src > main > resources > import.sql
1 INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('GRYFFINDOR', 25, 'Arancha');
2 INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('RAVENCLAW', 18, 'Eva');
3 INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('SLYTHERIN', 14, 'Rodri');
4 INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('HUFFLEPUFF', 8, 'Pepito');
5
  
```

Ejecuto la api y compruebo en MySQL que se genera tanto la tabla como los valores:

```

MariaDB [jugadoresquizz_hp]> show tables;
+-----+
| Tables_in_jugadoresquizz_hp |
+-----+
| resultado                    |
+-----+
1 row in set (0.001 sec)

MariaDB [jugadoresquizz_hp]> select * from resultado;
+-----+-----+-----+-----+
| puntos | id | nombre | clasificacion |
+-----+-----+-----+-----+
|      25 | 1 | Arancha | GRYFFINDOR    |
|      18 | 2 | Eva     | RAVENCLAW     |
|      14 | 3 | Rodri   | SLYTHERIN     |
|       8 | 4 | Pepito  | HUFFLEPUFF    |
+-----+-----+-----+-----+
4 rows in set (0.001 sec)

MariaDB [jugadoresquizz_hp]>
  
```

- **Activar y documentar la depuración para ver las consultas SQL que está generando el ORM automáticamente:**

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format_sql=true

Estas líneas de configuración se utilizan para controlar la visualización y el formato de las consultas SQL generadas por Hibernate.

1. **spring.jpa.show-sql=true:** indica a Spring Boot que muestre las consultas SQL generadas por Hibernate en la consola de registro. Esto puede ser útil para depurar y comprender qué consultas se están ejecutando en la base de datos.
2. **spring.jpa.properties.hibernate.format_sql=true:** le indica a Hibernate que formatee las consultas SQL generadas para que sean más legibles cuando se imprimen en la consola de registro.. Esto es especialmente útil cuando se trabaja con consultas SQL complejas o largas.

Añado estas líneas a mi application.properties:

```

1  spring.datasource.url=jdbc:mysql://localhost/jugadoresquizz_hp
2
3  spring.datasource.username=arancha
4  spring.datasource.password=arancha
5
6  spring.jpa.generate-ddl=false
7  spring.jpa.hibernate.ddl-auto=update
8
9  spring.jpa.show-sql=true
10 spring.jpa.properties.hibernate.format_sql=true

```

Ejecuto la api (tras borrar la tabla nuevamente), y vemos el resultado en consola:

```

Quiz_MySql_Ej2 - QuizMySqlEj2Application [Spring Boot App] C:\Users\savia\OneDrive\Escritorio\sts-4.21.0.RELEASE\plugins\org.eclipse
2024-02-01T20:11:52.097+01:00 INFO 16608 --- [ restartedMain] o.h.c.internal.RegionFactoryInitiator
2024-02-01T20:11:52.386+01:00 INFO 16608 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo
2024-02-01T20:11:52.477+01:00 WARN 16608 --- [ restartedMain] org.hibernate.dialect.Dialect
2024-02-01T20:11:53.308+01:00 INFO 16608 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator
Hibernate:
    drop table if exists resultado
Hibernate:
    create table resultado (
        puntos integer not null,
        id bigint not null auto_increment,
        nombre varchar(255),
        clasificacion enum ('GRYFFINDOR','RAVENCLAW','SLYTHERIN','HUFFLEPUFF'),
        primary key (id)
    ) engine=InnoDB
Hibernate: INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('GRYFFINDOR', 25, 'Arancha')
Hibernate: INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('RAVENCLAW', 18, 'Eva')
Hibernate: INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('SLYTHERIN', 14, 'Rodri')
Hibernate: INSERT INTO resultado (clasificacion, puntos, nombre) VALUES ('HUFFLEPUFF', 8, 'Pepito')
2024-02-01T20:11:53.389+01:00 INFO 16608 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBe
2024-02-01T20:11:53.649+01:00 WARN 16608 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguratio
2024-02-01T20:11:54.035+01:00 INFO 16608 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer
2024-02-01T20:11:54.095+01:00 INFO 16608 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer

```

En esta línea, Hibernate está generando la consulta SQL para crear la tabla resultado con sus columnas. La columna clasificacion está definida como un tipo enum con los valores 'GRYFFINDOR', 'RAVENCLAW', 'SLYTHERIN', y 'HUFFLEPUFF'. Esta línea por lo tanto indica que esta consulta se ha mapeado correctamente.

Y las siguientes líneas indican que los insert con los valores correspondientes se han volcado correctamente.

Es decir, vemos las consultas de los dos archivos sql que se están ejecutando, de esta manera vemos más fácilmente que estos archivos sql se ejecutan correctamente.

**** Para hacer más pruebas voy a generar consultas en el controlador:**

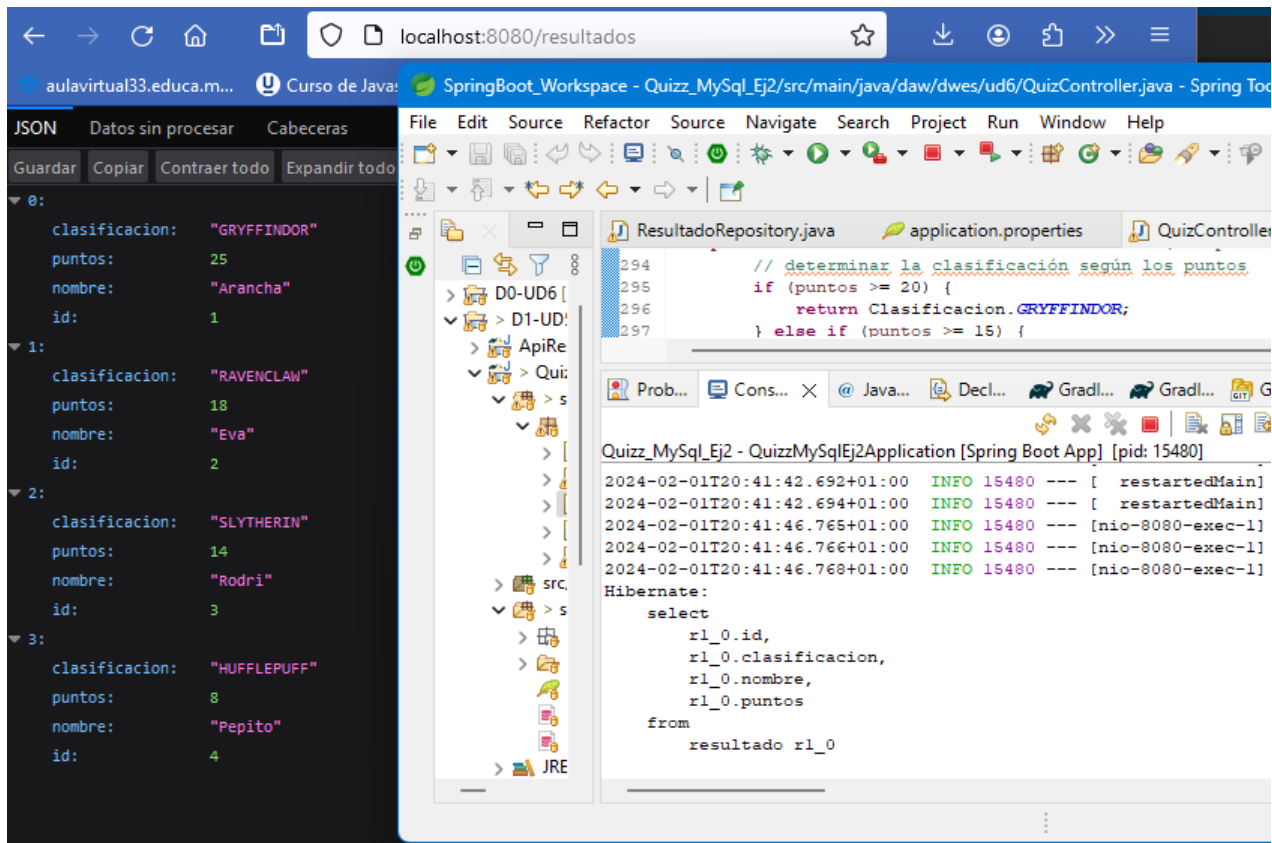
Método obtener todos los resultados con .findAll(): de tipo List.

```

@GetMapping("/resultados")
public List<Resultado> obtenerTodosLosResultados() {
    return resultadoRepositorio.findAll();
}

```

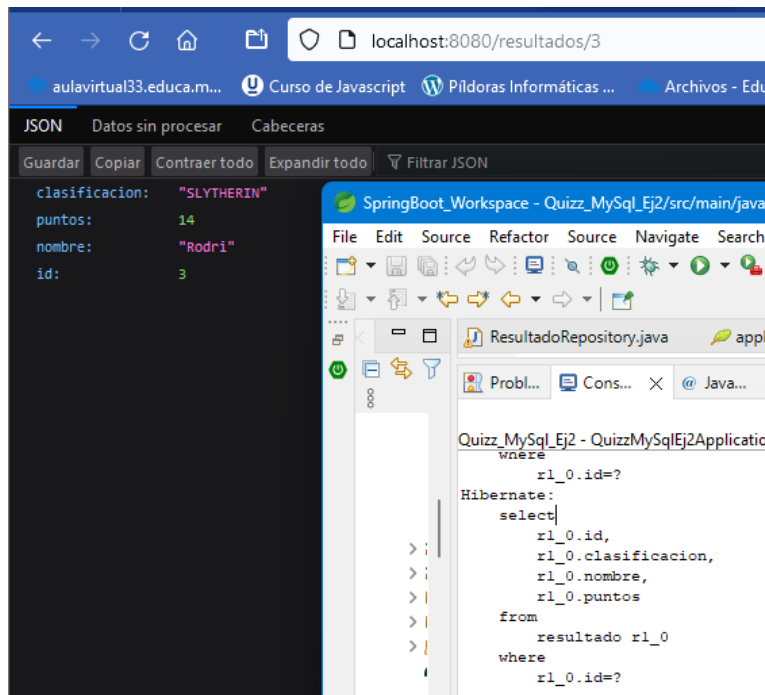
Ejecuto en navegador y veo resultado en navegador y consola:



Método GET buscar un resultado con `.findById()`: tipo Optional.

```

@GetMapping("/resultados/{id}")
public Optional<Resultado> buscarResultadoId(@PathVariable ("id") Long id){
    return resultadoRepositorio.findById(id);
} //BuscarId
  
```

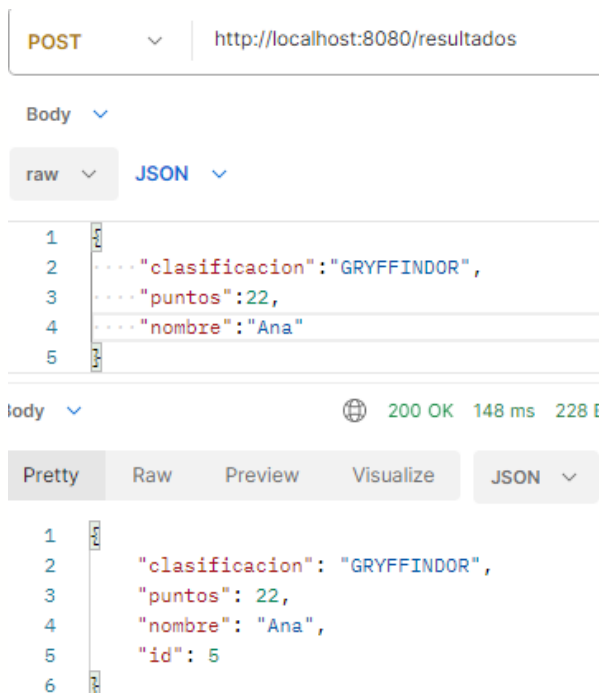
Método POST agregar nuevo resultado con `.save()`. De tipo Resultado.

```

@PostMapping("/resultados")
public Resultado agregarResultado(@RequestBody Resultado nuevoResultado)
    return resultadoRepositorio.save(nuevoResultado);
} //agregar

```

Creo resultado con postman:



Y veo resultado en consola de STS:

```

2024-02-01T20:55:08.167+01:00 INFO 15480 --- [nio-8080-e
2024-02-01T20:55:08.167+01:00 INFO 15480 --- [nio-8080-e
2024-02-01T20:55:08.168+01:00 INFO 15480 --- [nio-8080-e
Hibernate:
    insert
    into
        resultado
        (clasificacion, nombre, puntos)
    values
        (?, ?, ?)

```

Y en mySql:

```

MariaDB [jugadoresquizz_hp]> select * from resultado;
+-----+-----+-----+-----+
| puntos | id | nombre | clasificacion |
+-----+-----+-----+-----+
| 25     | 1 | Arancha | GRYFFINDOR    |
| 18     | 2 | Eva     | RAVENCLAW     |
| 14     | 3 | Rodri   | SLYTHERIN     |
| 8       | 4 | Pepito  | HUFFLEPUFF    |
| 22     | 5 | Ana     | GRYFFINDOR    |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)

```

4.- Para el excelente (dos de los tres puntos, además del bien)

- Implementar borrado y/o modificación de clasificaciones. Filtrado.

Método PUT modificar resultado con .save() y .findById(). De tipo Response Entity para manejar la respuesta.

- Primero con .findById(id) le asigno el elemento correspondiente a variable resultadoOptional,
- Segundo, si no existe (si no .isPresent()) return notFound.
- Tercero, si si existe, con .save guardo el resultado en el repositorio y return noContent.

```

@PutMapping("/resultados/{id}")
public ResponseEntity<Object> actualizarResultado(
    @PathVariable (name="id") Long id,
    @RequestBody Resultado resultadoActualizado) {
    Optional<Resultado> resultadoOptional = resultadoRepositorio.findById(id);

    if (!resultadoOptional.isPresent()) {
        return ResponseEntity.notFound().build();
    }

    resultadoActualizado.setId(id);
    resultadoRepositorio.save(resultadoActualizado);
    return ResponseEntity.noContent().build();
}

```

En postman modifiko el elemento con id 5, compruebo en MySql que se modifica el elemento y en consola de STS se ve la consulta con formato, pero no se muestran los valores:

Body

raw JSON

```
1 {
2   ... \"clasificacion\": \"HUFFLEPUFF\",
3   ... \"puntos\": 7,
4   ... \"nombre\": \"Ana\"
5 }
```

Body 204 No Content 80 n

Pretty Raw Preview Visualize Text

1

MySQL [jugadoresquizz_hp]> select * from resultado;

puntos	id	nombre	clasificacion
25	1	Arancha	GRYFFINDOR
18	2	Eva	RAVENCLAW
14	3	Rodri	SLYTHERIN
8	4	Pepito	HUFFLEPUFF
22	5	Ana	GRYFFINDOR

5 rows in set (0.001 sec)

MySQL [jugadoresquizz_hp]> select * from resultado;

puntos	id	nombre	clasificacion
25	1	Arancha	GRYFFINDOR
18	2	Eva	RAVENCLAW
14	3	Rodri	SLYTHERIN
8	4	Pepito	HUFFLEPUFF
7	5	Ana	HUFFLEPUFF

5 rows in set (0.001 sec)

```
Quiz_MySql_Ej2 - QuizMySqlEj2Application [Spring Boot App] [pid: 15
2024-02-01T21:16:46.568+01:00 INFO 15480 --- [ rest
2024-02-01T21:16:50.183+01:00 INFO 15480 --- [nio-808
2024-02-01T21:16:50.183+01:00 INFO 15480 --- [nio-808
2024-02-01T21:16:50.184+01:00 INFO 15480 --- [nio-808
Hibernate:
select
  r1_0.id,
  r1_0.clasificacion,
  r1_0.nombre,
  r1_0.puntos
from
  resultado r1_0
where
  r1_0.id=?
Hibernate:
update
  resultado
set
  clasificacion=?,
  nombre=?,
  puntos=?
where
  id=?
```

Método delete borrar resultado con `.deleteById()` y `.findById()`: de tipo `ResponseEntity` para manejar la respuesta.

Los pasos seguidos son los mismos que en el método `put`, pero si el elemento existe, lo borramos de la bbdd con `.deleteById(id)`.

```
@DeleteMapping("/resultados/{id}")
public ResponseEntity<Object> eliminarResultado(@PathVariable("id") Long id) {
    Optional<Resultado> resultadoOptional = resultadoRepositorio.findById(id);

    if (!resultadoOptional.isPresent()) {
        return ResponseEntity.notFound().build();
    }
    resultadoRepositorio.deleteById(id);
    return ResponseEntity.noContent().build();
} //eliminar
```

En postman borro el elemento con id 5, nombre Ana, compruebo en MySql que se elimina y en la consola STS la consulta con formato, pero no se ven los valores:

The screenshot displays three components related to a database deletion operation:

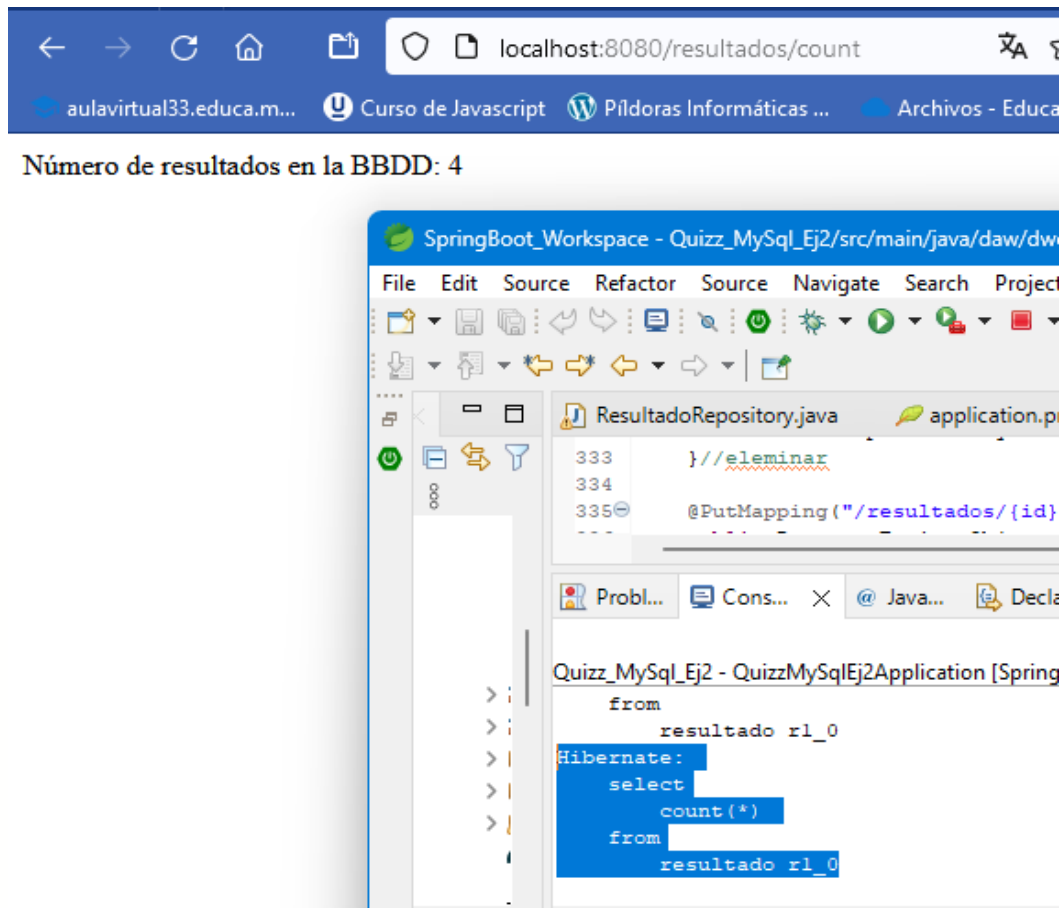
- Postman:** A DELETE request to `http://localhost:8080/resultados/5` is shown. The response is `204 No Content`.
- MySQL Terminal:** Two queries are executed on the `jugadoresquizz_hp` database. The first query shows 5 rows, including Ana (id 5). The second query, after deletion, shows 4 rows, with Ana removed.
- Spring Boot Console:** The log shows the Hibernate SQL for selecting and deleting the record with id=5. The delete statement is:


```
delete
from
  resultado
where
  id=5
```

Método contar resultados con .count(): de tipo ResponseEntity

Utilizo el método count() para obtener el número total de resultados en el repositorio y se lo asignamos a variable de tipo long, y lo devolvemos como parte de la respuesta.

```
@GetMapping("/resultados/count")
public ResponseEntity<Object> contarResultados() {
    long totalResultados = resultadoRepositorio.count();
    return ResponseEntity.ok("Número de resultados en la BBDD: "
        + totalResultados);
}
```



- Usar el monitor de qué consultas se hacen en tiempo real a la base de datos:
<https://mkyong.com/spring-boot/spring-boot-show-hibernate-sql-query/>

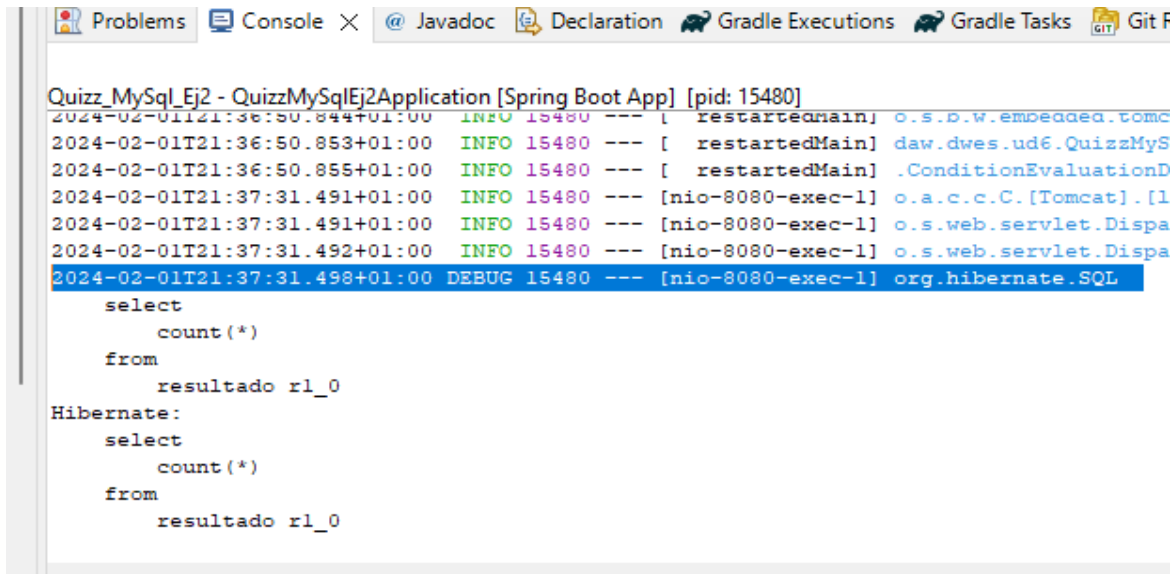
Añado estas líneas en application.properties:

```
#show sql statement
logging.level.org.hibernate.SQL=debug

#show sql values
logging.level.org.hibernate.type.descriptor.sql=trace
```

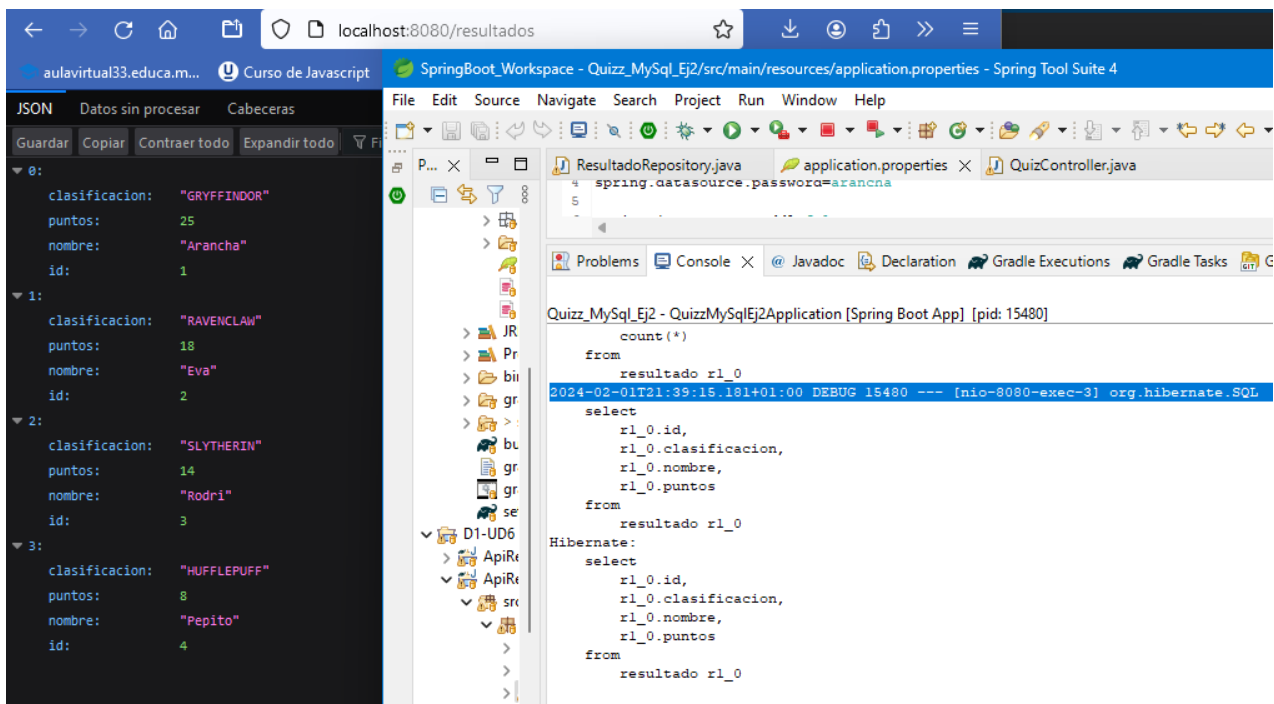
Y al probar a hacer consultas, se puede ver una línea de debug que aparece el tiempo exacto en el que se ha realizado la consulta.

Pruebo con varias consultas diferentes:



Quizz_MySql_Ej2 - QuizzMySqlEj2Application [Spring Boot App] [pid: 15480]

```
2024-02-01T21:36:50.844+01:00 INFO 15480 --- [ restartedMain] o.s.b.w.embedded.tomcat
2024-02-01T21:36:50.853+01:00 INFO 15480 --- [ restartedMain] daw.dwes.ud6.QuizzMyS
2024-02-01T21:36:50.855+01:00 INFO 15480 --- [ restartedMain] .ConditionEvaluationD
2024-02-01T21:37:31.491+01:00 INFO 15480 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[1
2024-02-01T21:37:31.491+01:00 INFO 15480 --- [nio-8080-exec-1] o.s.web.servlet.Dispa
2024-02-01T21:37:31.492+01:00 INFO 15480 --- [nio-8080-exec-1] o.s.web.servlet.Dispa
2024-02-01T21:37:31.498+01:00 DEBUG 15480 --- [nio-8080-exec-1] org.hibernate.SQL
    select
      count(*)
    from
      resultado rl_0
Hibernate:
    select
      count(*)
    from
      resultado rl_0
```



localhost:8080/resultados

aulavirtual33.educa.m... Curso de Javascript SpringBoot_Workspace - Quizz_MySql_Ej2/src/main/resources/application.properties - Spring Tool Suite 4

JSON Datos sin procesar Cabeceras

Guardar Copiar Contraer todo Expandir todo

```
0: {
  "clasificacion": "GRYFFINDOR",
  "puntos": 25,
  "nombre": "Arancha",
  "id": 1
}
1: {
  "clasificacion": "RAVENCLAW",
  "puntos": 18,
  "nombre": "Eva",
  "id": 2
}
2: {
  "clasificacion": "SLYHERIN",
  "puntos": 14,
  "nombre": "Rodri",
  "id": 3
}
3: {
  "clasificacion": "HUFFLEPUFF",
  "puntos": 8,
  "nombre": "Pepito",
  "id": 4
}
```

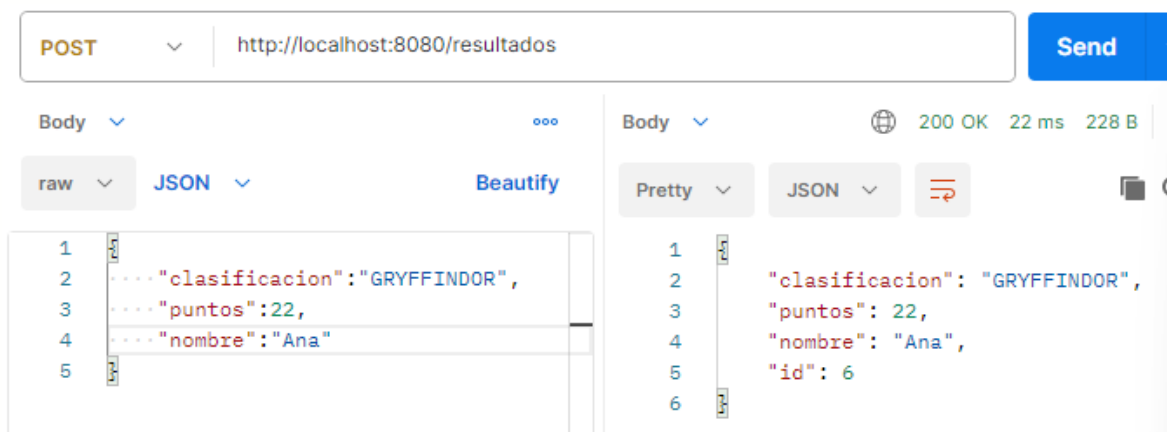
File Edit Source Navigate Search Project Run Window Help

ResultadoRepository.java application.properties QuizController.java

```
spring.datasource.password=arancha
```

Quizz_MySql_Ej2 - QuizzMySqlEj2Application [Spring Boot App] [pid: 15480]

```
count(*)
from
  resultado rl_0
2024-02-01T21:39:15.181+01:00 DEBUG 15480 --- [nio-8080-exec-3] org.hibernate.SQL
    select
      rl_0.id,
      rl_0.clasificacion,
      rl_0.nombre,
      rl_0.puntos
    from
      resultado rl_0
Hibernate:
    select
      rl_0.id,
      rl_0.clasificacion,
      rl_0.nombre,
      rl_0.puntos
    from
      resultado rl_0
```



POST http://localhost:8080/resultados Send

Body raw JSON Beautify

```
1 {
2   "clasificacion": "GRYFFINDOR",
3   "puntos": 22,
4   "nombre": "Ana"
5 }
```

Body Pretty JSON

```
1 {
2   "clasificacion": "GRYFFINDOR",
3   "puntos": 22,
4   "nombre": "Ana",
5   "id": 6
6 }
```

```

2024-02-01T21:40:29.514+01:00 DEBUG 15480 --- [nio-8080-exec-6] org.hibernate.SQL
    insert
    into
        resultado
        (clasificacion, nombre, puntos)
    values
        (?, ?, ?)
Hibernate:
    insert
    into
        resultado
        (clasificacion, nombre, puntos)
    values
        (?, ?, ?)

```

La línea `logging.level.org.hibernate.type.descriptor.sql=trace`, creo que es la que hace que se muestren los valores. Cambio el estado `trace` a `true`, y sale error en la consola, pero me da las opciones siguientes:

```

<terminated> Quizz_MySql_Ej2 - QuizzMySQLEj2Application [Spring Boot App] C:\Users\savia\OneDrive\Escritorio\sts-4.21.0.RELEASE\plugins'
Failed to bind properties under 'logging.level.org.hibernate.type.descriptor.sql' to org.springframework.b

Property: logging.level.org.hibernate.type.descriptor.sql
Value: "true"
Origin: class path resource [application.properties] - 17:49
Reason: failed to convert java.lang.String to org.springframework.boot.logging.LogLevel (caused by jav

Action:

Update your application's configuration. The following values are valid:

DEBUG
ERROR
FATAL
INFO
OFF
TRACE
WARN

```

Pruebo con el estado `debug` y tampoco se muestran los valores, siguen mostrándose interrogaciones:

```

2024-02-04T02:17:26.542+01:00 DEBUG 9656 --- [nio-80
    insert
    into
        resultado
        (clasificacion, nombre, puntos)
    values
        (?, ?, ?)
Hibernate:
    insert
    into
        resultado
        (clasificacion, nombre, puntos)
    values
        (?, ?, ?)

```

CONCLUSIONES

He dejado los dos proyectos usados para esta práctica en mi repositorio de github:

<https://github.com/AranchaC/D1-UD5.git>

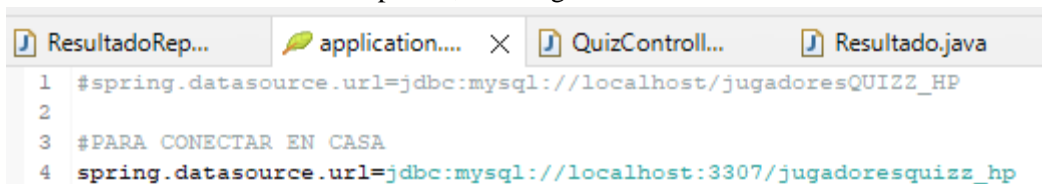
-**ApiRest_MySql_Ej1**: para el ejercicio 1.

-**Quizz_MySql_Ej2**: para los ejercicios 2 , 3 y 4.

Me ha gustado mucho hacer esta práctica, sobretudo el hecho de adaptar Apis que ya teníamos hechas pues hemos podido ver que hay muchas maneras de realizar las Apis y sus funciones. Como bbdd me gusta, el probar a hacer consultas con mi propia api y sus valores es muy interesante.

He aprendido mucho sobre las funciones del repositorio en spring gracias a sus dependencias.

También he aprendido mucho a ver los logs de la consola de STS y manejar los mensajes de error, sobre todo en casa al conectar con mysql de allí. Pues como en casa uso Windows y tengo mysql con xampp, lo uso en el puerto 3307 y como no es puerto por defecto, tuve que especificar en applicatio.properties, en la ruta de conexión con la bbdd el puerto, de la siguiente manera:

A screenshot of an IDE window showing a configuration file. The window has several tabs at the top: 'ResultadoRep...', 'application...', 'QuizControll...', and 'Resultado.java'. The 'application...' tab is active. The code in the editor is as follows:

```
1 #spring.datasource.url=jdbc:mysql://localhost/jugadoresQUIZZ_HP
2
3 #PARA CONECTAR EN CASA
4 spring.datasource.url=jdbc:mysql://localhost:3307/jugadoresquizz_hp
```

Pero llegar a esa conclusión me costó mucho, muchas consultas con chatGpt, muchas consultas con los archivos de configuración, tanto de sts como de xamp, y consultas con los logs tanto de la consola de sts como de mysql.

También me ha gustado ver las diferentes formas de crear la tabla, de forma automática y manual con archivo schema.sql.

La forma automática está muy bien, pues se puede personalizar/especificar el nombre y tipo de datos a mapear a la bbdd.

Sobre otros problemas encontrados, los he ido describiendo en el documento pues lo he ido redactando sobre la marcha.