

## Parcial 2

Jorge Esteban Herrera Jimenez – 833060

Corporación Universitaria Minuto de Dios

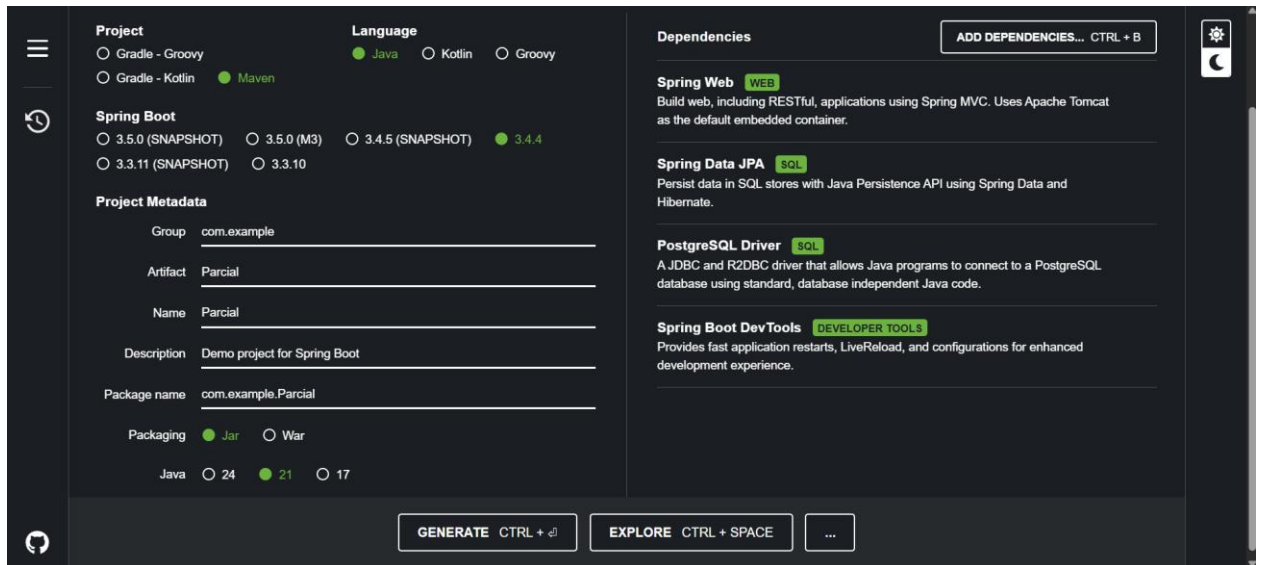
Arquitectura de Software

Ing. De Sistemas

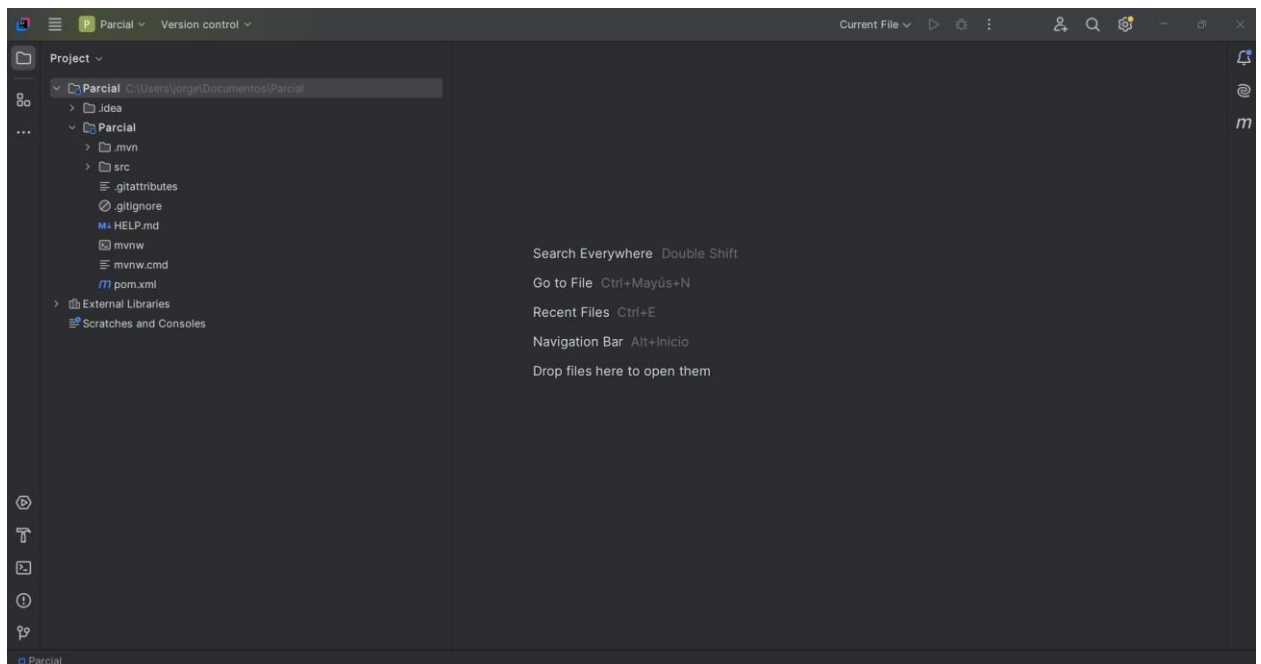
Ing. William Alexander Matallana Porras

2025

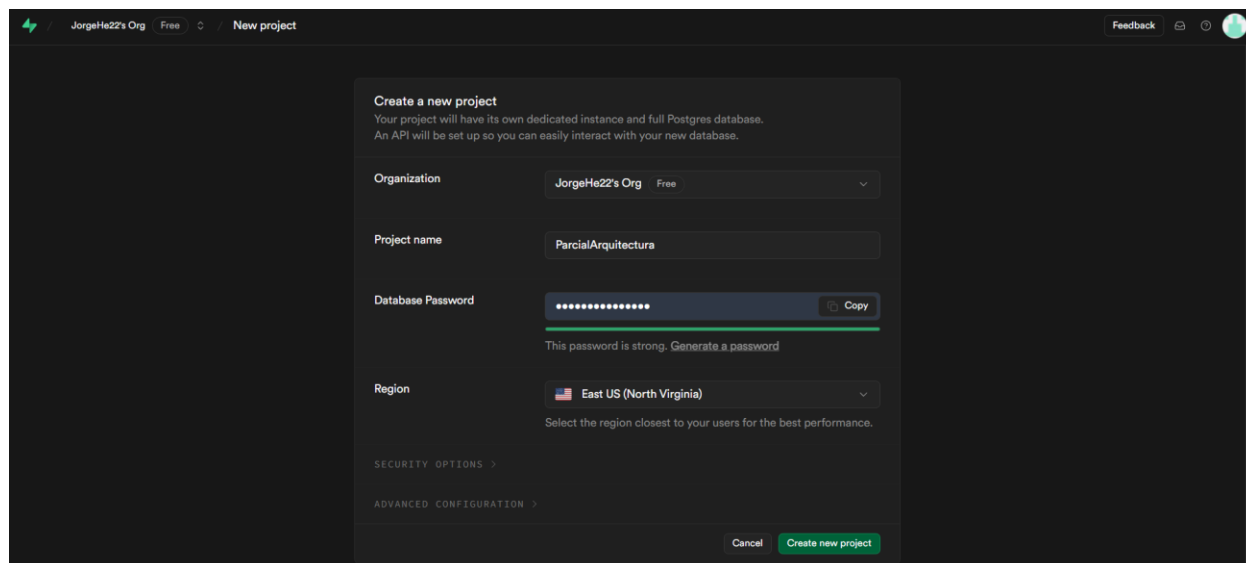
1. Primero debemos descargar el .zip en la pagina “start.spring.io” con las respectivas dependencias que necesitamos para este trabajo:



2. Seguidamente lo debemos descomprimir en una ruta que podamos ubicar para cargar el proyecto en intelliJ, una vez descomprimido vamos a intelliJ y abrimos el proyecto que descomprimimos, para esto nos dirigimos en file -> open -> (Acá ponemos la ruta donde se descomprimió y abrimos el proyecto) de manera que quede así:



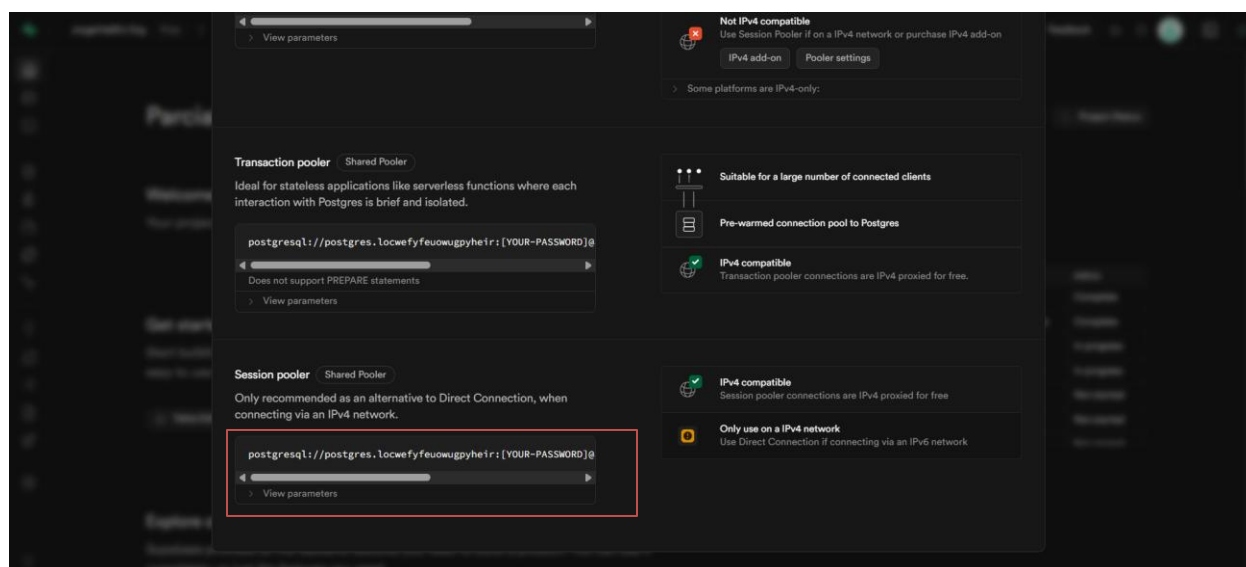
3. **Crecacion en supabase:** Primeramente, de se debe crear un nuevo proyecto en la página de supabase para realizar el trabajo, cuando se esté creando hay que tener en cuenta que se debe guardar la contraseña con la que se crea el proyecto.



The screenshot shows the 'Create a new project' form in the Supabase dashboard. The form is titled 'Create a new project' and includes the following fields and options:

- Organization:** JorgeHe22's Org (Free)
- Project name:** ParcialArquitectura
- Database Password:** A password field with a 'Copy' button. Below it, a message states: 'This password is strong. Generate a password'.
- Region:** East US (North Virginia). Below it, a message states: 'Select the region closest to your users for the best performance.'
- SECURITY OPTIONS >**
- ADVANCED CONFIGURATION >**
- Buttons:** 'Cancel' and 'Create new project'.

Una vez creado, vamos a entrar al proyecto que creamos y vamos a la parte donde dice conect y bajamos hasta el final de la ventana y copiamos el siguiente link:



The screenshot shows the 'Connect' page in the Supabase dashboard. It displays the 'Session pooler' connection string, which is highlighted with a red box. The connection string is:

```
postgresql://postgres.lcwefyfeuwugpyheir:[YOUR-PASSWORD]@
```

Below the connection string, there is a 'View parameters' link. To the right of the connection string, there are several informational boxes:

- Not IPv4 compatible:** Use Session Pooler if on a IPv4 network or purchase IPv4 add-on. Includes links for 'IPv4 add-on' and 'Pooler settings'.
- Suitable for a large number of connected clients:** Pre-warmed connection pool to Postgres.
- IPv4 compatible:** Transaction pooler connections are IPv4 proxied for free.
- IPv4 compatible:** Session pooler connections are IPv4 proxied for free.
- Only use on a IPv4 network:** Use Direct Connection if connecting via an IPv6 network.

4. **Conectar al proyecto:** Para conectar al proyecto se debe crear un archivo .env en el cual vamos a requerir la contraseña con la que creamos el proyecto en supabase y el link que copiamos anteriormente para colocarlo en el archivo .env de la siguiente manera:

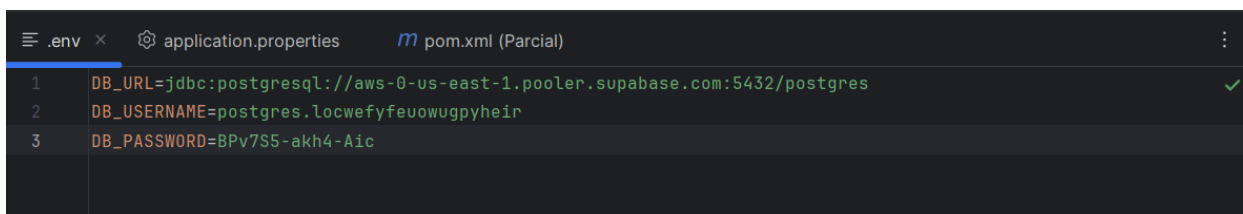
Aca el link se va dividir en 2 partes:

```
postgresql://postgres.locwefyfeowugpyheir:[YOUR-PASSWORD]@aws-0-us-east-1.pooler.supabase.com:5432/postgres
```

- **DB\_URL:** acá se coloca lo que esta subrayado de amarillo (es importante que siempre este el “jdbc:postgresql://” antes de poner la parte amarilla)

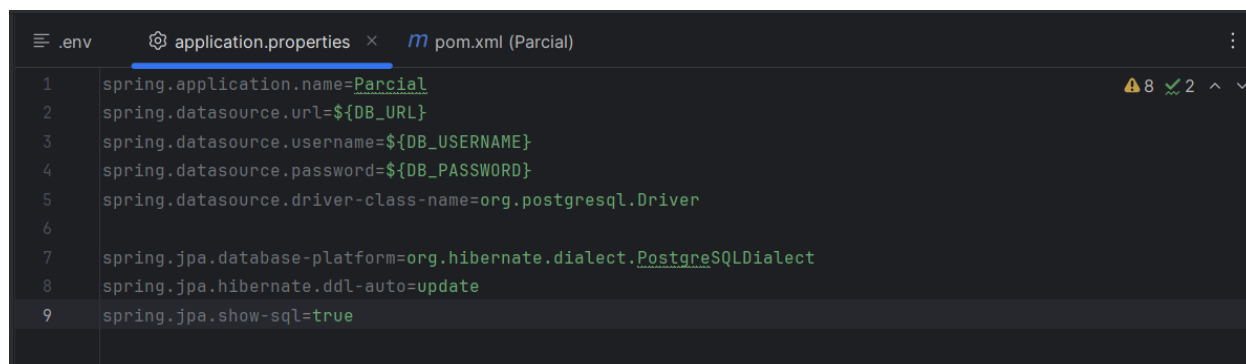
- **DB\_USERNAME:** Acá se coloca lo que este subrayado de verd

- **DB: PASSWORD:** Y acá vamos a colocar la clave que le colocamos al proyecto en supabase



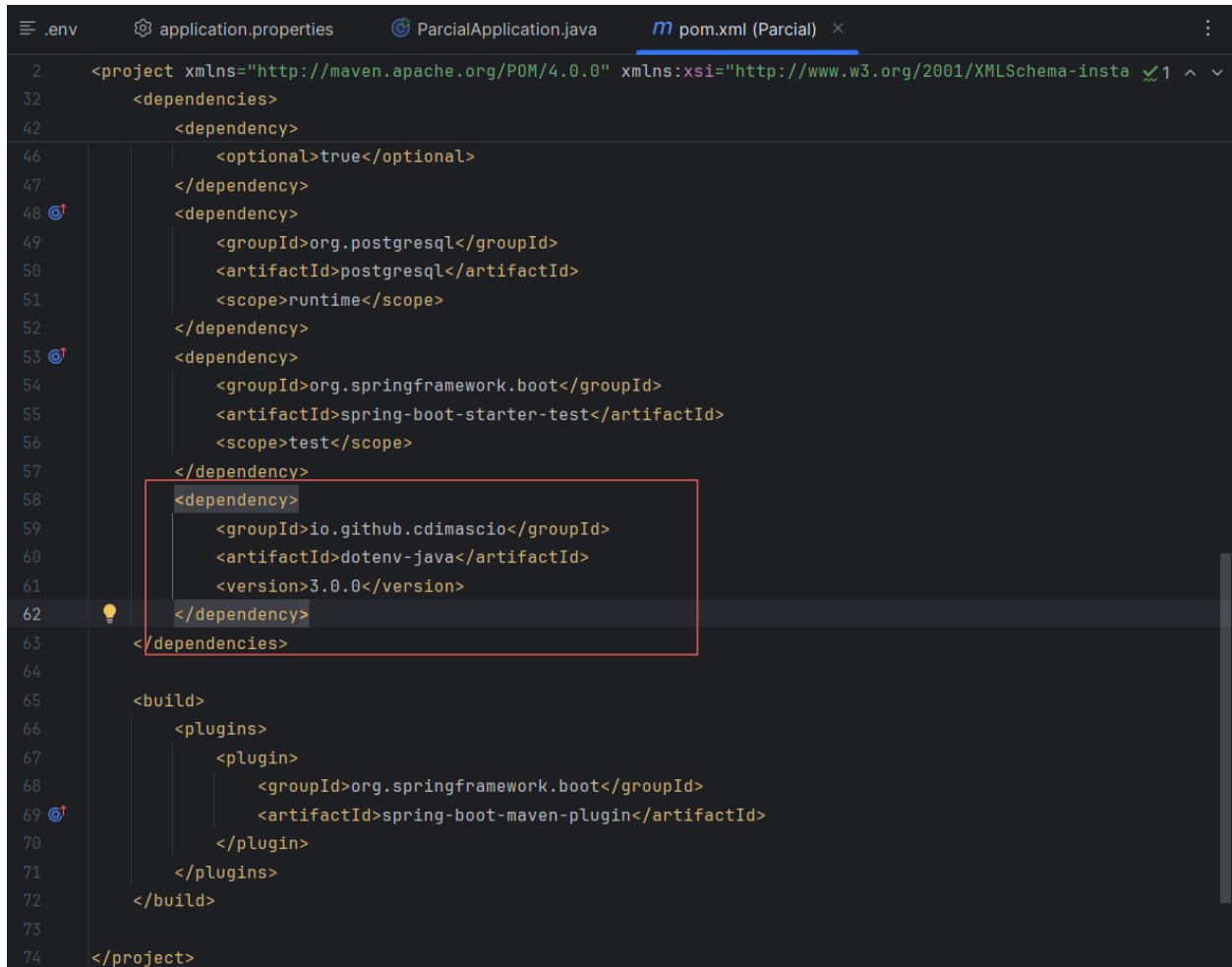
```
.env
1 DB_URL=jdbc:postgresql://aws-0-us-east-1.pooler.supabase.com:5432/postgres
2 DB_USERNAME=postgres.locwefyfeowugpyheir
3 DB_PASSWORD=BPv7S5-akh4-Aic
```

Ya después de crear el archivo “.env” y nos dirigimos al archivo application.properties y agregamos las siguientes líneas de texto:



```
.env application.properties pom.xml (Parcial)
1 spring.application.name=Parcial
2 spring.datasource.url=${DB_URL}
3 spring.datasource.username=${DB_USERNAME}
4 spring.datasource.password=${DB_PASSWORD}
5 spring.datasource.driver-class-name=org.postgresql.Driver
6
7 spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
8 spring.jpa.hibernate.ddl-auto=update
9 spring.jpa.show-sql=true
```

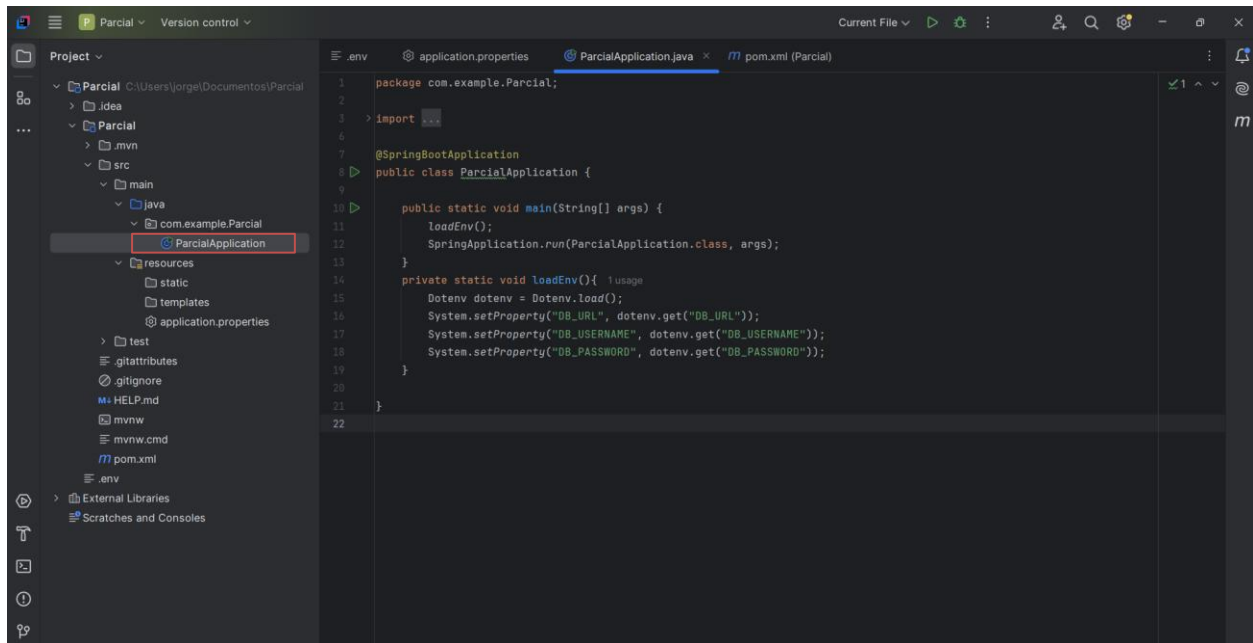
Seguidamente en el pom.xml debemos colocar la siguiente dependencia para que pueda funcionar la conexión:

A screenshot of an IDE window showing a Maven pom.xml file. The file is titled 'pom.xml (Parcial)' and has tabs for '.env', 'application.properties', 'ParcialApplication.java', and 'pom.xml (Parcial)'. The code is as follows:

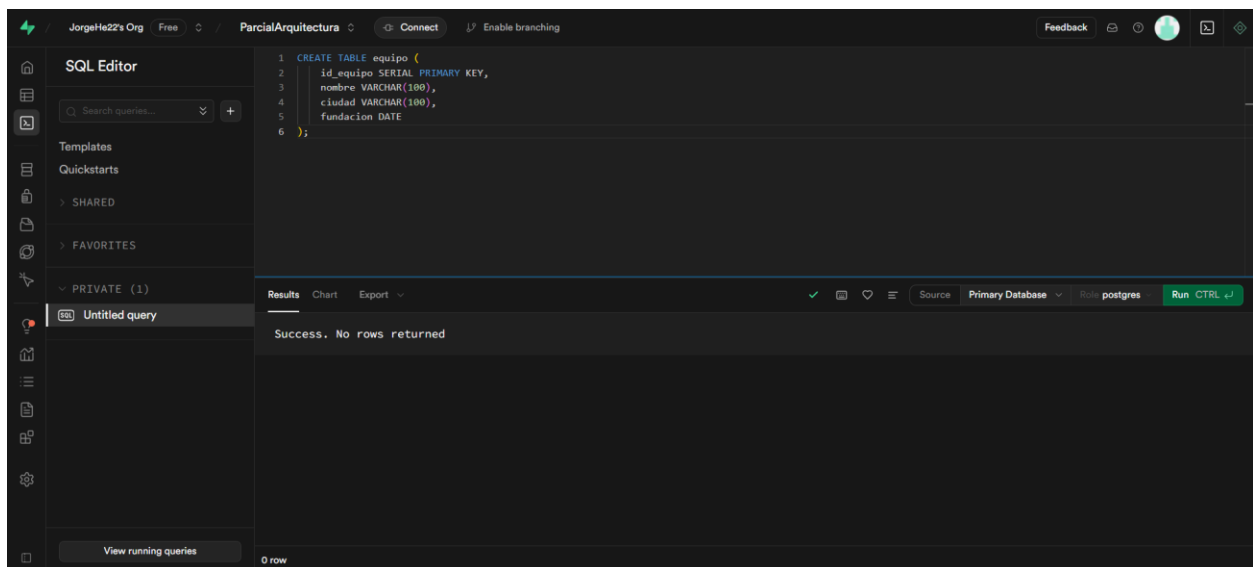
```
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
32 <dependencies>
42 <dependency>
46 <groupId>org.postgresql</groupId>
47 <artifactId>postgresql</artifactId>
48 <scope>runtime</scope>
49 </dependency>
50 <dependency>
51 <groupId>org.springframework.boot</groupId>
52 <artifactId>spring-boot-starter-test</artifactId>
53 <scope>test</scope>
54 </dependency>
55 <dependency>
56 <groupId>io.github.cdimascio</groupId>
57 <artifactId>dotenv-java</artifactId>
58 <version>3.0.0</version>
59 </dependency>
60 </dependencies>
61
62 <build>
63 <plugins>
64 <plugin>
65 <groupId>org.springframework.boot</groupId>
66 <artifactId>spring-boot-maven-plugin</artifactId>
67 </plugin>
68 </plugins>
69 </build>
70
71 </project>
```

The dependency for 'io.github.cdimascio:dotenv-java:3.0.0' is highlighted with a red box. The line numbers 2, 32, 42, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, and 74 are visible on the left side of the editor.

Después nos dirigimos a la aplicación y colocamos las siguientes líneas de código para completar la conexión:



- Una vez hecha la conexión, nos devolvemos a la pagina de supabase para la creación de las tablas que se nos pide en el documento. Aca nos dirigimos a la parte izquierda y damos clic en “SQL Editor” y creamos las tablas:



The image displays three sequential screenshots of a SQL Editor interface, likely from a database management tool like DBeaver. Each screenshot shows the 'SQL Editor' tab with a 'Team Information Table' selected in the left sidebar. The interface includes a top bar with 'JorgeHe22's Org', 'Free', 'ParcialArquitectura', 'Connect', and 'Enable branching' buttons. The main area contains a SQL query editor, a 'Results' tab, and a 'Run' button.

**First Screenshot: Creating the 'jugador' table**

```
1 CREATE TABLE jugador (  
2   id_jugador SERIAL PRIMARY KEY,  
3   nombre VARCHAR(100),  
4   posicion VARCHAR(50),  
5   dorsal INT,  
6   fecha_nac DATE,  
7   nacionalidad VARCHAR(100),  
8   id_equipo INT REFERENCES equipo(id_equipo)  
9 );
```

The 'Results' tab shows 'Success. No rows returned' and '0 row'.

**Second Screenshot: Creating the 'entrenador' table**

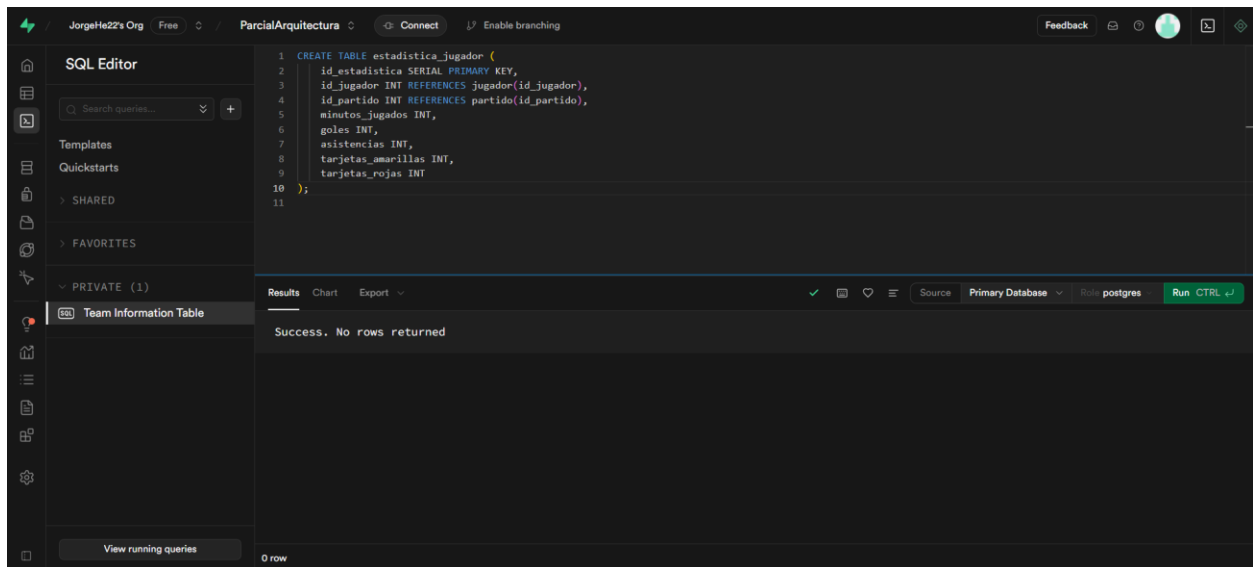
```
1 CREATE TABLE entrenador (  
2   id_entrenador SERIAL PRIMARY KEY,  
3   nombre VARCHAR(100),  
4   especialidad VARCHAR(100),  
5   id_equipo INT REFERENCES equipo(id_equipo)  
6 );
```

The 'Results' tab shows 'Success. No rows returned' and '0 row'.

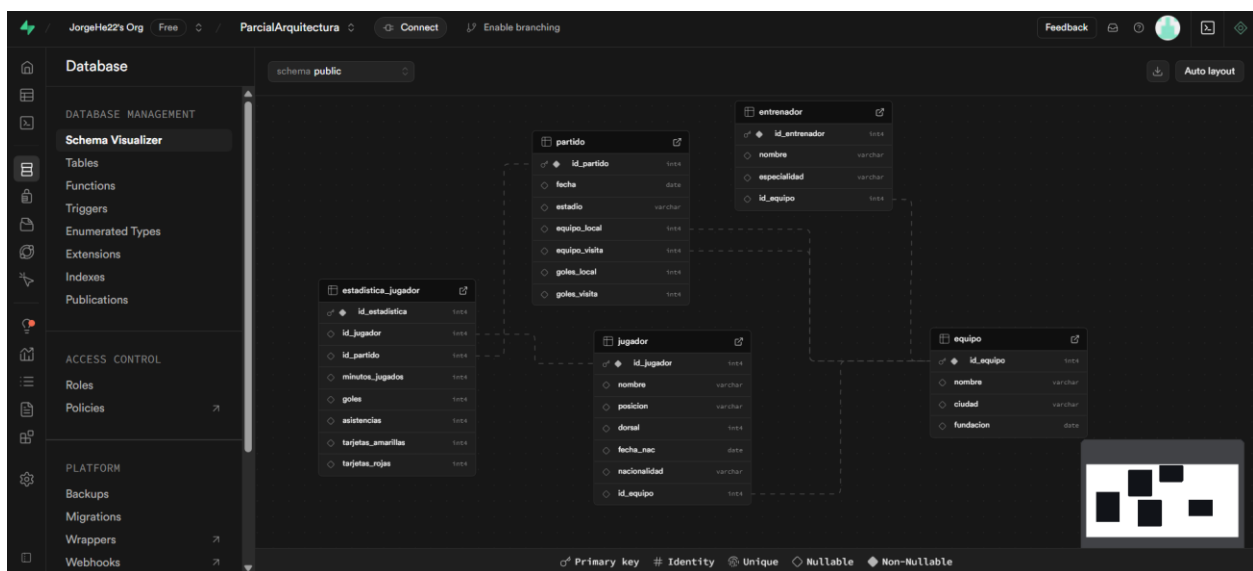
**Third Screenshot: Creating the 'partido' table**

```
1 CREATE TABLE partido (  
2   id_partido SERIAL PRIMARY KEY,  
3   fecha DATE,  
4   estadio VARCHAR(100),  
5   equipo_local INT REFERENCES equipo(id_equipo),  
6   equipo_visitante INT REFERENCES equipo(id_equipo),  
7   goles_local INT,  
8   goles_visitante INT  
9 );  
10
```

The 'Results' tab shows 'Success. No rows returned' and '0 row'.

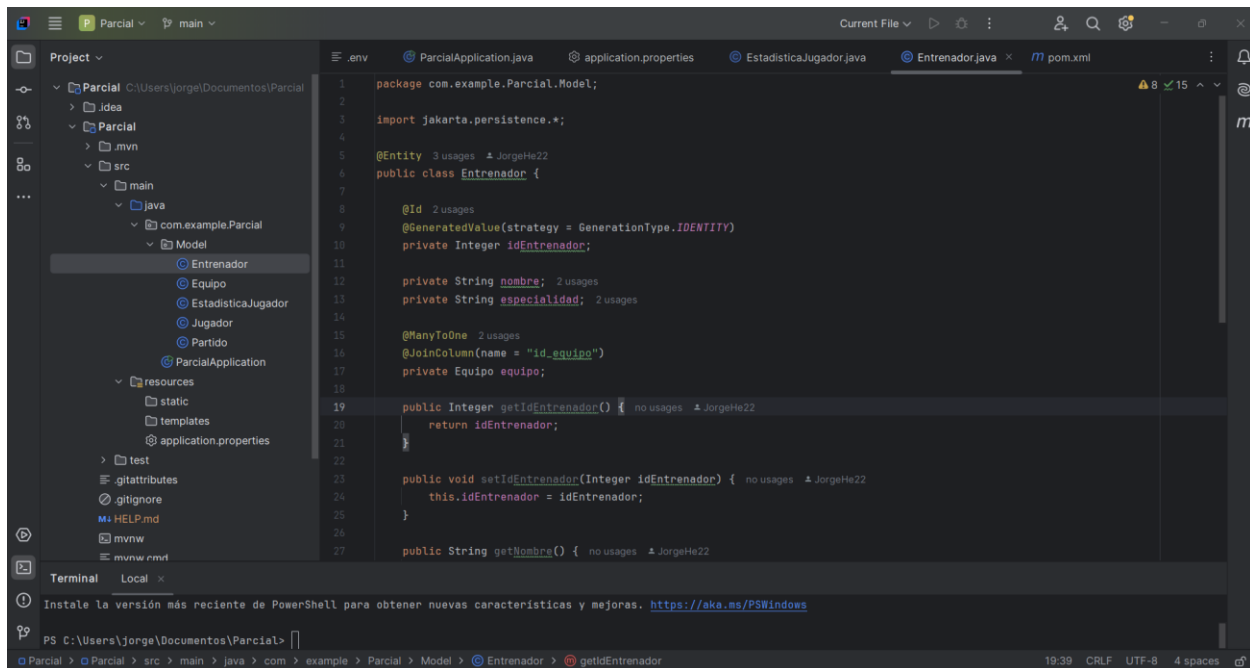


Una vez creadas las tablas, nos dirigimos al menú de la parte izquierda y damos clic en “Database”, una vez aca podremos observar el modelo de datos en supabase:

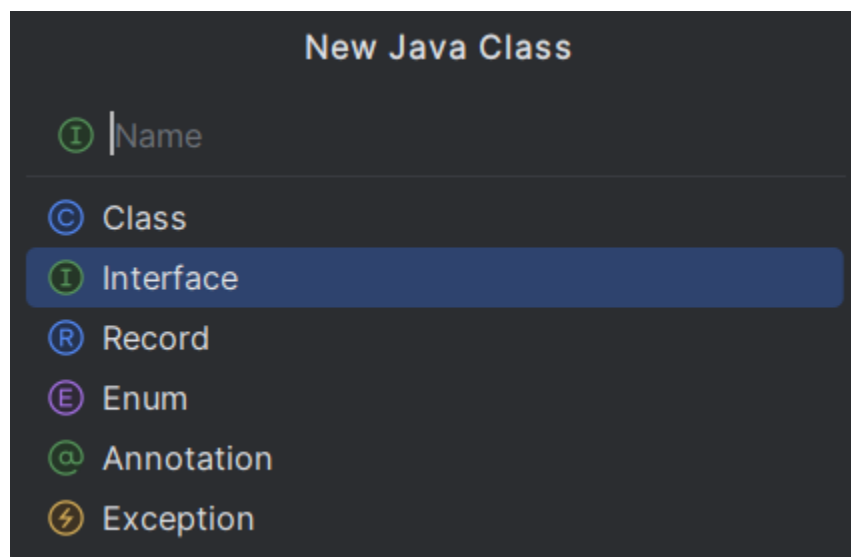


- Ahora vamos a crear las entidades del modelo en java, primero vamos a crear el paquete model con sus clases, relaciones y tipos de datos correctos, debe quedar de la siguiente manera:

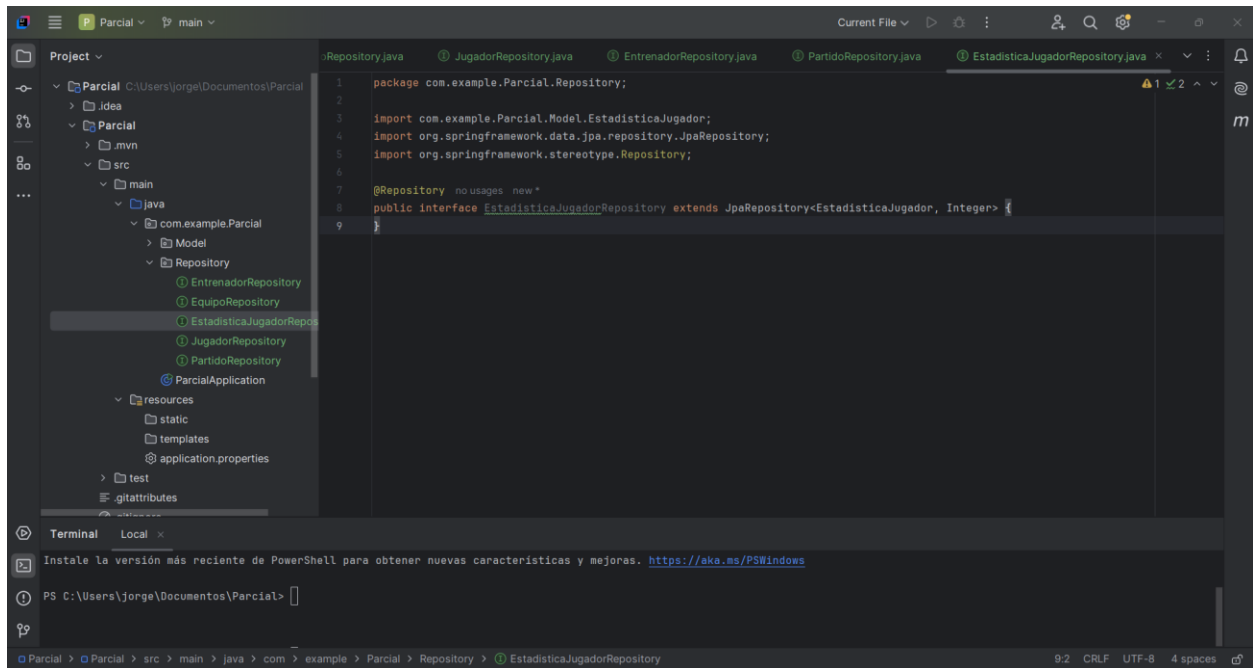




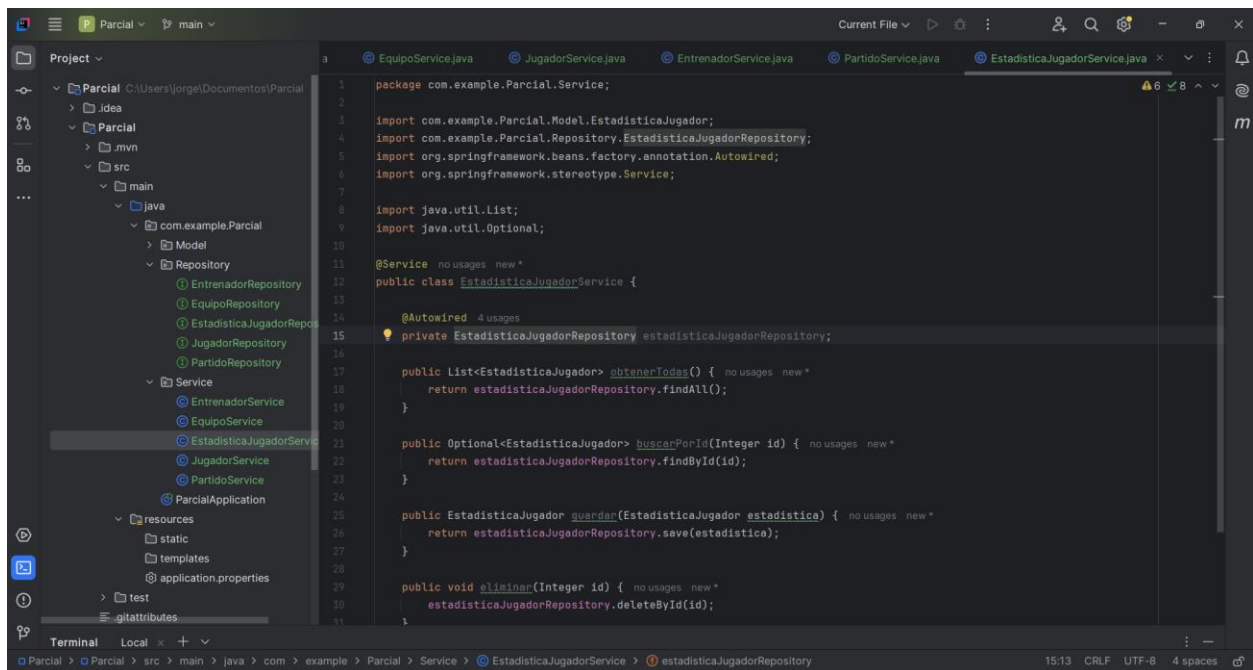
Seguidamente vamos a crear ahora el paquete “Repository” y aca respectivamente sus archivos, con la diferencia de que cuando se vayan a crear seleccionamos como archivo “Interface”



Y así de esta manera vamos a crear una interfaz para cada entidad de manera que nos quede así:



7. Ahora vamos a crear los servicios, vamos a crear el paquete “service”, en donde se manejara la lógica entre los controladores y los repositorios:



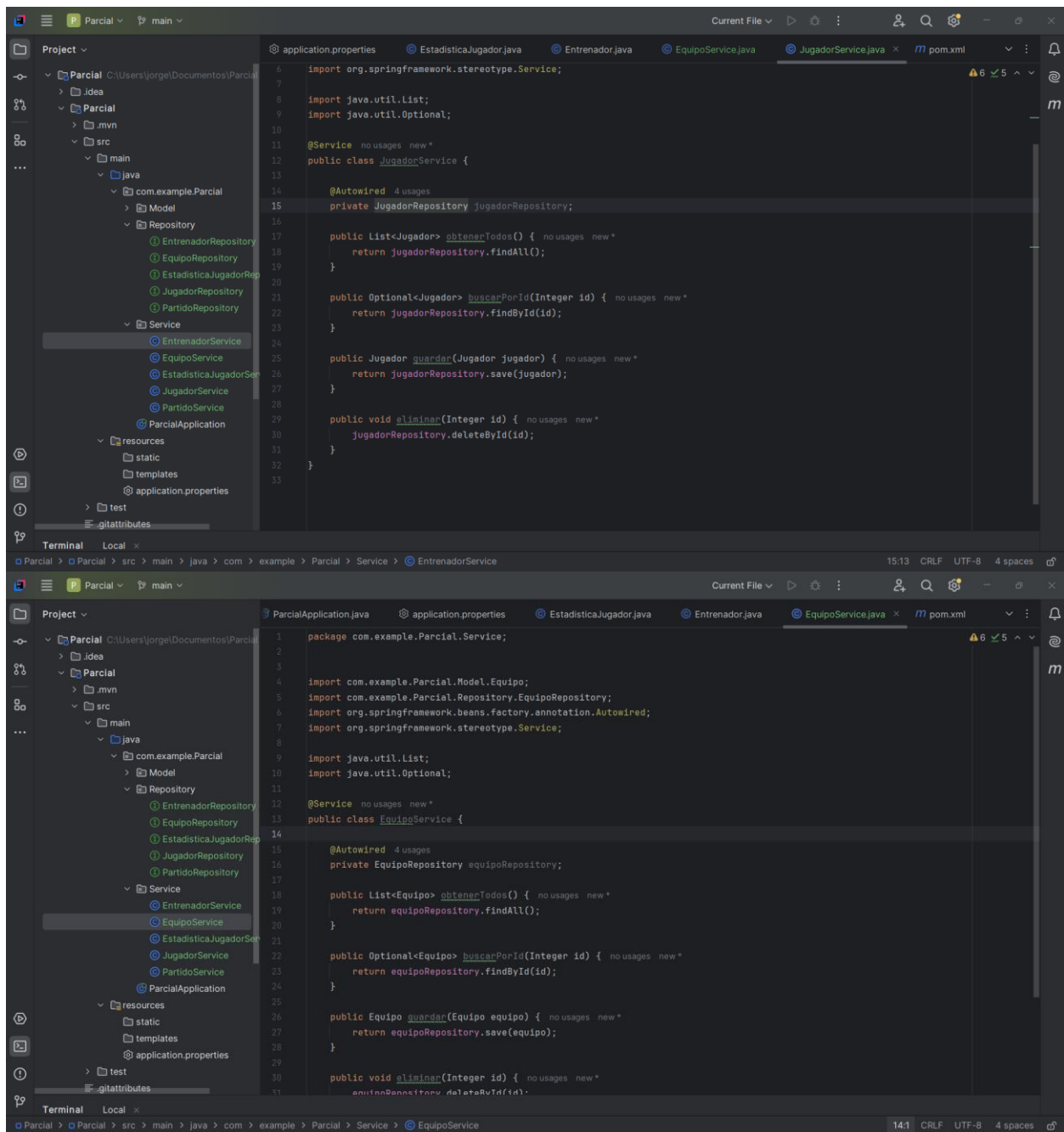
The image displays two screenshots of an IDE (IntelliJ IDEA) showing Java code for a Spring Boot application. The top screenshot shows the `PartidoService.java` file, and the bottom screenshot shows the `EntrenadorService.java` file. Both files are part of the `com.example.Parcial.Service` package and use Spring annotations like `@Service` and `@Autowired`.

**Top Screenshot: `PartidoService.java`**

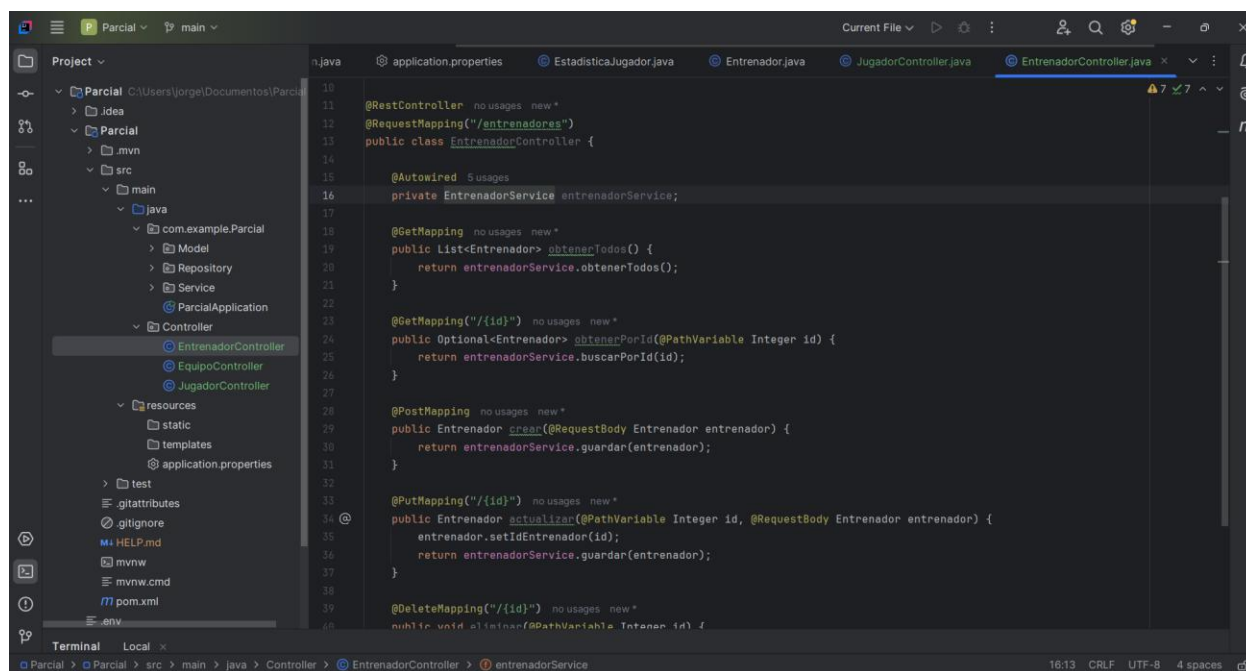
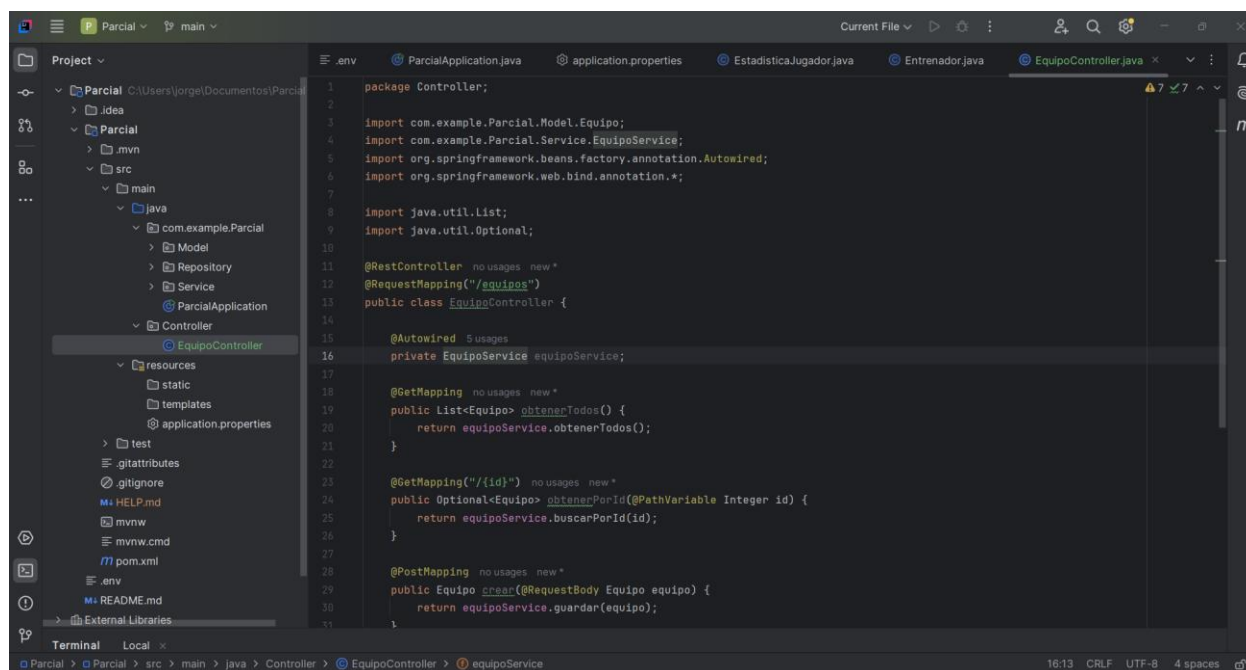
```
1 package com.example.Parcial.Service;
2
3 import com.example.Parcial.Model.Partido;
4 import com.example.Parcial.Repository.PartidoRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9 import java.util.Optional;
10
11 @Service
12 public class PartidoService {
13
14     @Autowired
15     private PartidoRepository partidoRepository;
16
17     public List<Partido> obtenerTodos() {
18         return partidoRepository.findAll();
19     }
20
21     public Optional<Partido> buscarPorId(Integer id) {
22         return partidoRepository.findById(id);
23     }
24
25     public Partido guardar(Partido partido) {
26         return partidoRepository.save(partido);
27     }
28
29     public void eliminar(Integer id) {
30         partidoRepository.deleteById(id);
31     }
32 }
```

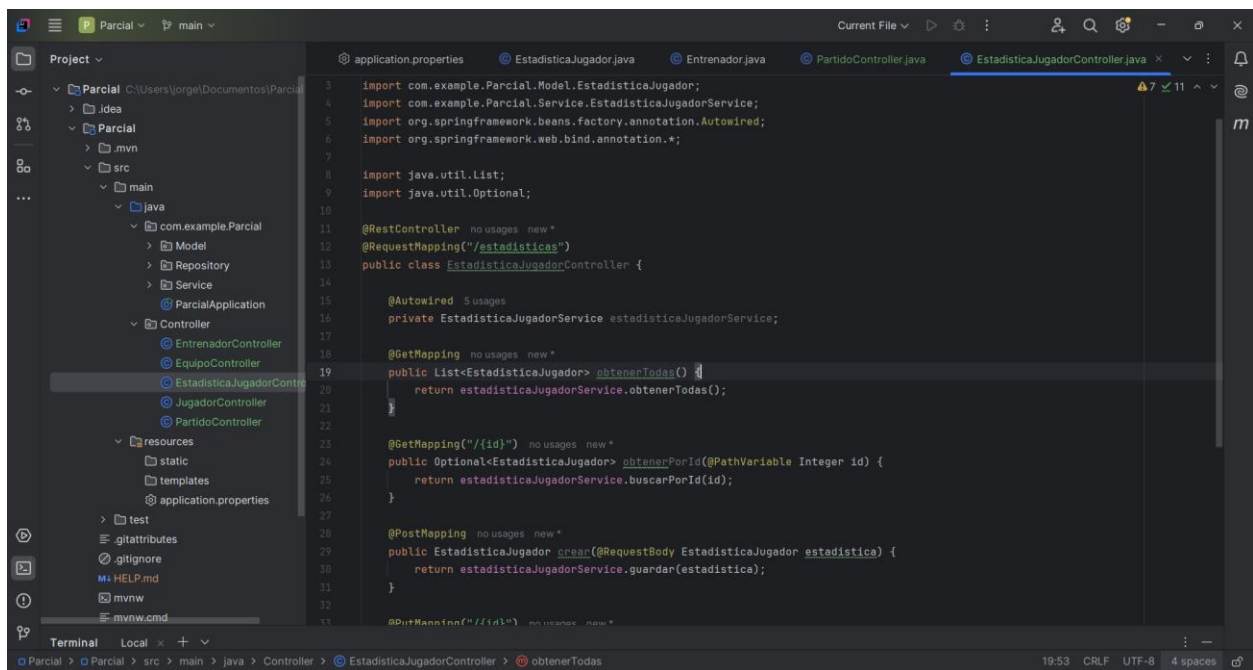
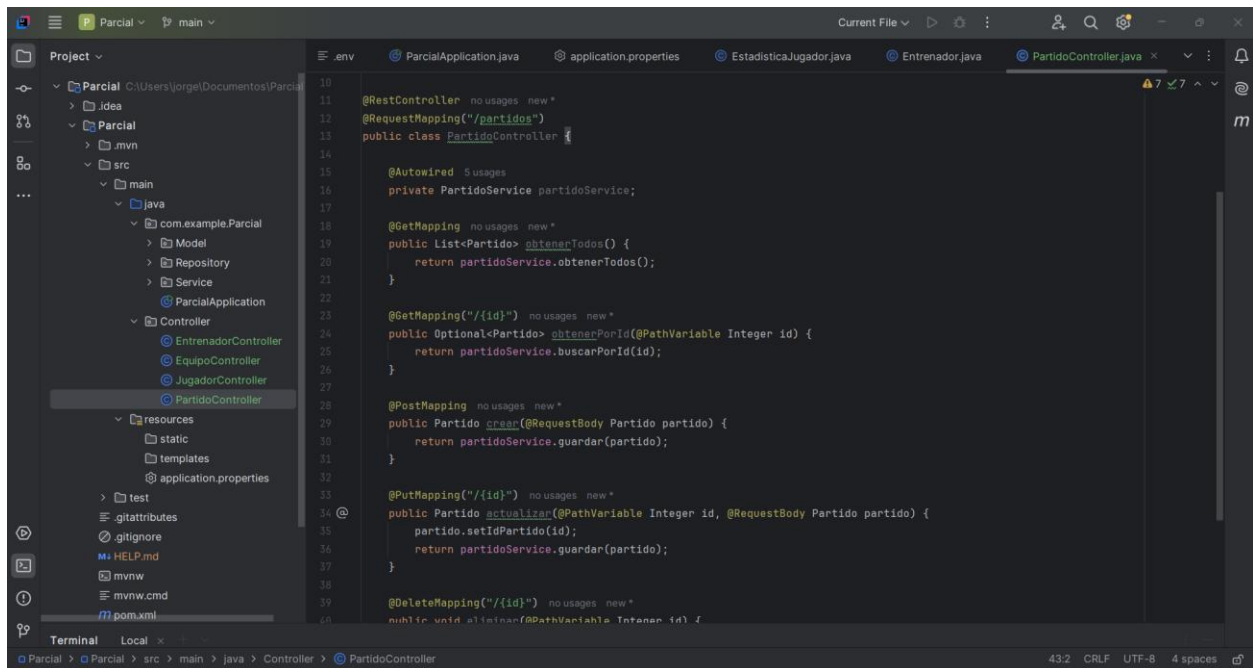
**Bottom Screenshot: `EntrenadorService.java`**

```
1 package com.example.Parcial.Service;
2
3 import com.example.Parcial.Model.Entrenador;
4 import com.example.Parcial.Repository.EntrenadorRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9 import java.util.Optional;
10
11 @Service
12 public class EntrenadorService {
13
14     @Autowired
15     private EntrenadorRepository entrenadorRepository;
16
17     public List<Entrenador> obtenerTodos() {
18         return entrenadorRepository.findAll();
19     }
20
21     public Optional<Entrenador> buscarPorId(Integer id) {
22         return entrenadorRepository.findById(id);
23     }
24
25     public Entrenador guardar(Entrenador entrenador) {
26         return entrenadorRepository.save(entrenador);
27     }
28
29     public void eliminar(Integer id) {
30         entrenadorRepository.deleteById(id);
31     }
32 }
```



8. Ahora vamos a crear los controladores (CRUD), para esto creamos un paquete con el nombre “Controller”, aca definiremos los endpoints REST para exponer las operaciones básicas (GET, POST, PUT, DELETE).





9. Ahora vamos a ingresar los registros para cada tabla, en este caso sern 100 por tabla, para generar estos registros se hizo uso de chatGPT:



SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Team Information Table

View running queries

```
1 INSERT INTO equipo (nombre, ciudad, fundacion) VALUES
2 ('Real Madrid', 'Madrid', '1902-03-06'),
3 ('FC Barcelona', 'Barcelona', '1899-11-29'),
4 ('Manchester United', 'Manchester', '1878-01-05'),
5 ('Liverpool', 'Liverpool', '1892-06-03'),
6 ('Chelsea', 'Londres', '1905-03-10'),
7 ('Arsenal', 'Londres', '1886-12-01'),
8 ('Juventus', 'Turin', '1897-11-01'),
9 ('AC Milan', 'Milán', '1899-12-16'),
10 ('Inter de Milán', 'Milán', '1908-03-09'),
11 ('Bayern Múnich', 'Múnich', '1900-02-27'),
12 ('Borussia Dortmund', 'Dortmund', '1909-12-19'),
13 ('PSG', 'Paris', '1970-08-12'),
14 ('Ajax', 'Ámsterdam', '1900-03-18'),
15 ('Feyenoord', 'Róterdam', '1908-07-19'),
16 ('Porto', 'Oporto', '1893-09-28'),
17 ('Benfica', 'Lisboa', '1904-02-28'),
18 ('Sporting CP', 'Lisboa', '1906-07-01'),
19 ('Celtic', 'Glasgow', '1887-11-06'),
20 ('Rangers', 'Glasgow', '1872-03-15'),
21 ('Atlético de Madrid', 'Madrid', '1903-04-26'),
22 ('Sevilla FC', 'Sevilla', '1890-01-25'),
23 ('Valencia CF', 'Valencia', '1919-03-18'),
24 ('Real Betis', 'Sevilla', '1907-09-12'),
25 ('Villarreal CF', 'Villarreal', '1923-01-10'),
26 ('Real Sociedad', 'San Sebastián', '1909-09-07'),
```

Results Chart Export

Primary Database postgres Run CTRL

Success. No rows returned

0 row

SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Team Information Table

View running queries

```
1 INSERT INTO jugador (nombre, posicion, dorsal, fecha_nac, nacionalidad, id_equipo) VALUES
2 ('Juan Pérez', 'portero', 1, '1990-03-14', 'Colombia', 1),
3 ('Luis Martínez', 'defensa', 2, '1992-06-22', 'México', 2),
4 ('Carlos López', 'mediocampista', 8, '1995-01-30', 'Argentina', 3),
5 ('Andrés Gómez', 'delantero', 9, '1998-04-12', 'Chile', 4),
6 ('Diego Torres', 'defensa', 3, '1991-11-19', 'Perú', 5),
7 ('Pedro Ramírez', 'portero', 12, '1993-07-08', 'Uruguay', 6),
8 ('Santiago Rojas', 'mediocampista', 6, '2000-09-27', 'Ecuador', 7),
9 ('Javier Ortega', 'delantero', 11, '1997-05-16', 'Paraguay', 8),
10 ('Sebastián Castro', 'defensa', 5, '1994-10-03', 'Colombia', 9),
11 ('Álvaro Jiménez', 'mediocampista', 14, '1989-12-22', 'México', 10),
12 ('Nicolás Herrera', 'delantero', 7, '1996-06-30', 'Argentina', 11),
13 ('Manuel Suárez', 'portero', 22, '1991-01-08', 'Chile', 12),
14 ('Cristian Díaz', 'defensa', 4, '1999-08-14', 'Perú', 13),
15 ('Fernando León', 'mediocampista', 16, '1993-02-18', 'Ecuador', 14),
16 ('Oscar Castillo', 'delantero', 17, '1997-01-05', 'Colombia', 15),
17 ('Mauricio Vega', 'defensa', 18, '1988-07-27', 'Uruguay', 16),
18 ('Rodrigo Reyes', 'mediocampista', 19, '1993-09-11', 'Venezuela', 17),
19 ('Héctor Salazar', 'delantero', 10, '1996-12-04', 'Paraguay', 18),
20 ('Iván Morales', 'portero', 20, '1994-05-23', 'Bolivia', 19),
21 ('Kevin Paredes', 'defensa', 21, '1990-10-01', 'Honduras', 20),
22 ('Mario Vargas', 'mediocampista', 13, '1995-04-18', 'Colombia', 21),
23 ('Esteban Ríos', 'delantero', 23, '2001-06-06', 'Chile', 22),
24 ('Daniel Soto', 'portero', 24, '1992-11-25', 'Argentina', 23),
```

Results Chart Export

Primary Database postgres Run CTRL

Success. No rows returned

0 row

SQL Editor

Search queries...

Templates

Quickstarts

SHARED

FAVORITES

PRIVATE (1)

Team Information Table

View running queries

```
1 INSERT INTO entrenador (nombre, especialidad, id_equipo) VALUES
2 ('Carlos Martínez', 'Entrenador Principal', 1),
3 ('Ana López', 'Asistente', 2),
4 ('Miguel Rodríguez', 'Preparador Físico', 3),
5 ('Laura Fernández', 'Entrenador Principal', 4),
6 ('Roberto Sánchez', 'Entrenador de Porteros', 5),
7 ('Isabel González', 'Nutricionista', 6),
8 ('Javier Pérez', 'Psicólogo Deportivo', 7),
9 ('María García', 'Entrenador Asistente', 8),
10 ('Antonio Díaz', 'Entrenador Principal', 9),
11 ('Elena Torres', 'Fisioterapeuta', 10),
12 ('David Ruiz', 'Entrenador Principal', 11),
13 ('Sofía Ramírez', 'Asistente', 12),
14 ('Pedro Álvarez', 'Preparador Físico', 13),
15 ('Carmen Moreno', 'Entrenador Principal', 14),
16 ('Juan Jiménez', 'Entrenador de Porteros', 15),
17 ('Patricia Vázquez', 'Nutricionista', 16),
18 ('Francisco Gómez', 'Psicólogo Deportivo', 17),
19 ('Lucía Martín', 'Entrenador Asistente', 18),
20 ('Pablo Hernández', 'Entrenador Principal', 19),
21 ('Raquel Ortega', 'Fisioterapeuta', 20),
22 ('Sergio Castro', 'Entrenador Principal', 21),
23 ('Alicia Romero', 'Asistente', 22),
24 ('Daniel Serrano', 'Preparador Físico', 23),
```

Results Chart Export

Primary Database postgres Run CTRL

Success. No rows returned

0 row

```

1 INSERT INTO partido (fecha, estadio, equipo_local, equipo_visitante, goles_local, goles_visitante) VALUES
2 ('2024-01-05', 'Estadio Azteca', 1, 23, 2, 1),
3 ('2024-01-06', 'Camp Nou', 34, 12, 3, 0),
4 ('2024-01-12', 'Santiago Bernabéu', 7, 45, 2, 2),
5 ('2024-01-13', 'Old Trafford', 56, 89, 1, 0),
6 ('2024-01-19', 'Allianz Arena', 22, 67, 4, 2),
7 ('2024-01-20', 'San Siro', 78, 33, 0, 0),
8 ('2024-01-26', 'Maracanã', 91, 10, 3, 1),
9 ('2024-01-27', 'Wembley', 44, 76, 1, 2),
10 ('2024-02-02', 'Signal Iduna Park', 13, 55, 2, 0),
11 ('2024-02-03', 'Anfield', 88, 29, 3, 3),
12 ('2024-02-09', 'Emirates Stadium', 3, 41, 1, 1),
13 ('2024-02-10', 'Parc des Princes', 60, 25, 4, 0),
14 ('2024-02-16', 'Giuseppe Meazza', 17, 92, 2, 1),
15 ('2024-02-17', 'Estadio da Luz', 39, 84, 0, 2),
16 ('2024-02-23', 'Etihad Stadium', 71, 50, 3, 0),
17 ('2024-02-24', 'Celtic Park', 26, 63, 1, 1),
18 ('2024-03-02', 'Johan Cruyff Arena', 5, 37, 2, 2),
19 ('2024-03-03', 'Estadio Monumental', 52, 18, 4, 1),
20 ('2024-03-09', 'Stamford Bridge', 95, 30, 0, 0),
21 ('2024-03-10', 'MetLife Stadium', 14, 73, 1, 2),
22 ('2024-03-16', 'Stade de France', 81, 46, 3, 1),
23 ('2024-03-17', 'La Bombonera', 28, 59, 2, 0),
24 ('2024-03-23', 'Allianz Stadium', 65, 97, 1, 1),

```

Results: Success. No rows returned

Aca pasa un problema y es que al usar autoincrement, se supone que se generaban automáticamente los id, pero se generaron contando desde 100, por lo cual el jugador\_id me toca ponerlo desde 100 hacia arriba.

```

1 INSERT INTO estadistica_jugador (id_jugador, id_partido, minutos_jugados, goles, asistencias, tarjetas_amarillas, tarjetas_rojas) VALUES
2 (100, 3, 90, 1, 0, 0, 0),
3 (101, 15, 87, 0, 1, 1, 0),
4 (102, 78, 90, 0, 0, 0, 0),
5 (103, 45, 76, 2, 1, 0, 0),
6 (104, 21, 90, 1, 0, 0, 0),
7 (105, 37, 65, 0, 1, 1, 0),
8 (106, 62, 90, 0, 2, 0, 0),
9 (107, 41, 90, 1, 0, 1, 0),
10 (108, 56, 79, 0, 0, 0, 0),
11 (109, 82, 90, 2, 1, 1, 0),
12 (110, 19, 67, 0, 0, 1, 0),
13 (111, 74, 90, 1, 1, 0, 0),
14 (112, 2, 46, 0, 0, 0, 1),
15 (113, 59, 90, 0, 0, 0, 0),
16 (114, 68, 90, 1, 0, 1, 0),
17 (115, 34, 72, 0, 1, 0, 0),
18 (116, 90, 90, 3, 1, 0, 0),
19 (117, 27, 90, 0, 0, 2, 0),
20 (118, 80, 84, 1, 1, 1, 0),
21 (119, 5, 90, 0, 2, 0, 0),
22 (120, 53, 90, 1, 0, 0, 0),
23 (121, 71, 67, 0, 0, 0, 0),
24 (122, 1, 90, 2, 1, 1, 0),
25 (123, 36, 79, 0, 1, 0, 0),
26 (124, 64, 90, 1, 0, 0, 0),

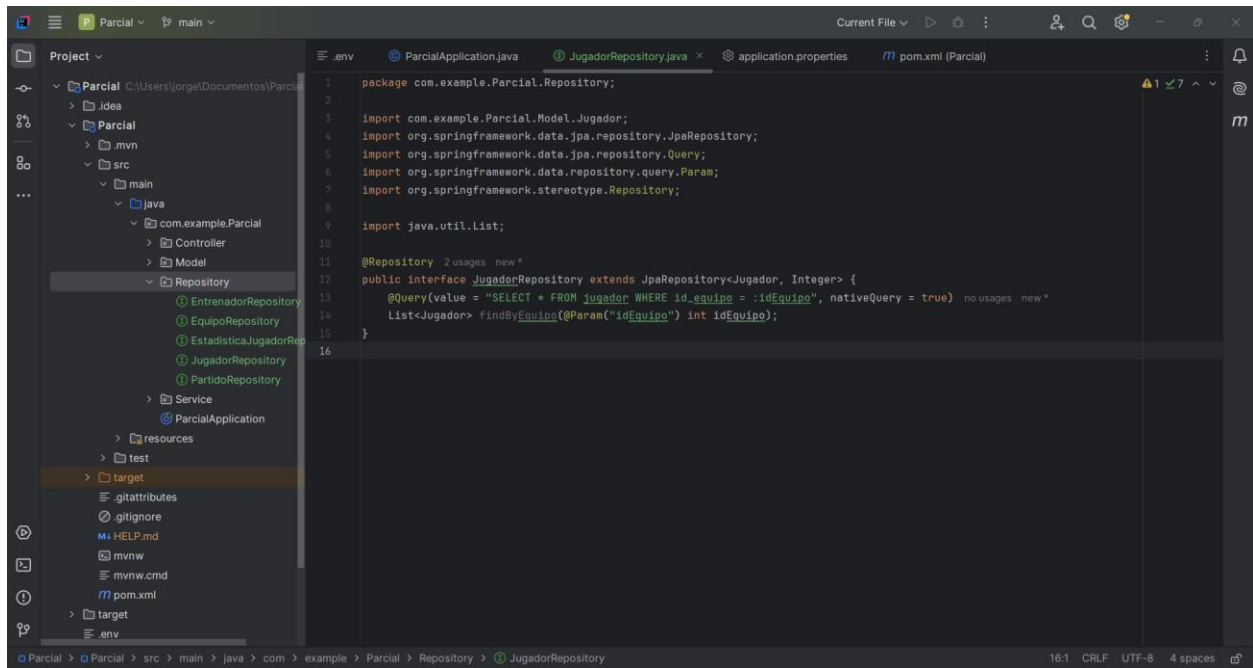
```

Results: Success. No rows returned

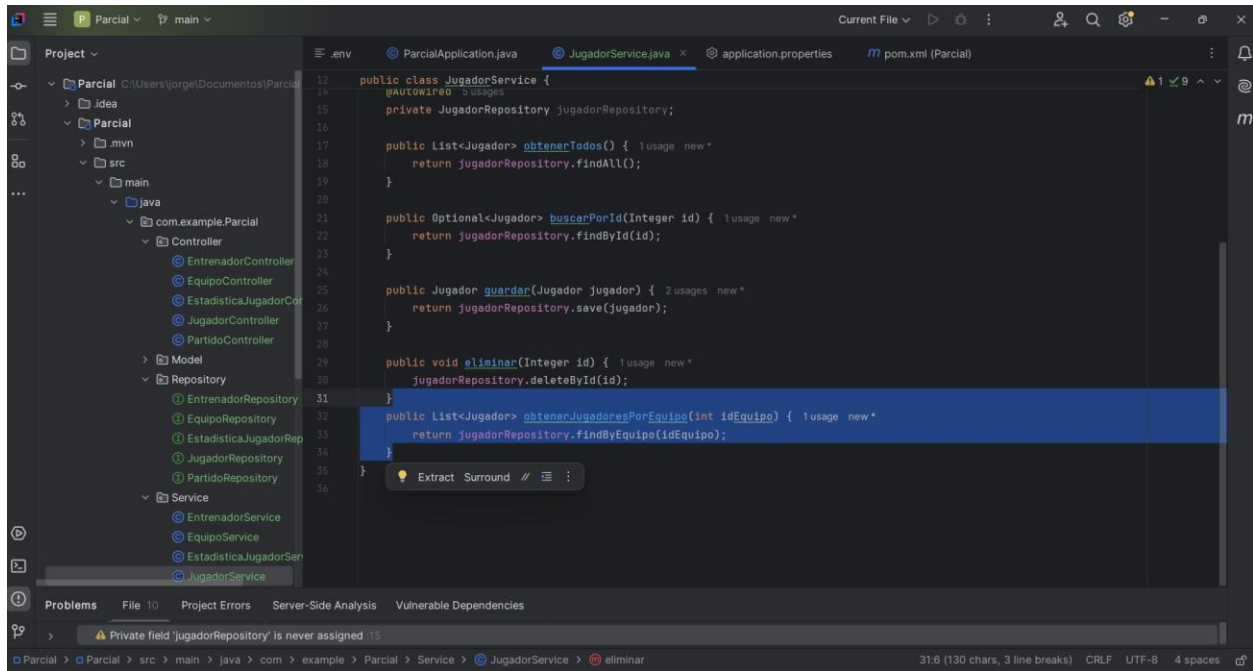
## 10. Consultas nativas: Para las consultas nativas empezamos por

1. **Obtener todos los jugadores de un equipo específico:** Aca vamos a colocar el siguiente comando en el JugadorRepository:

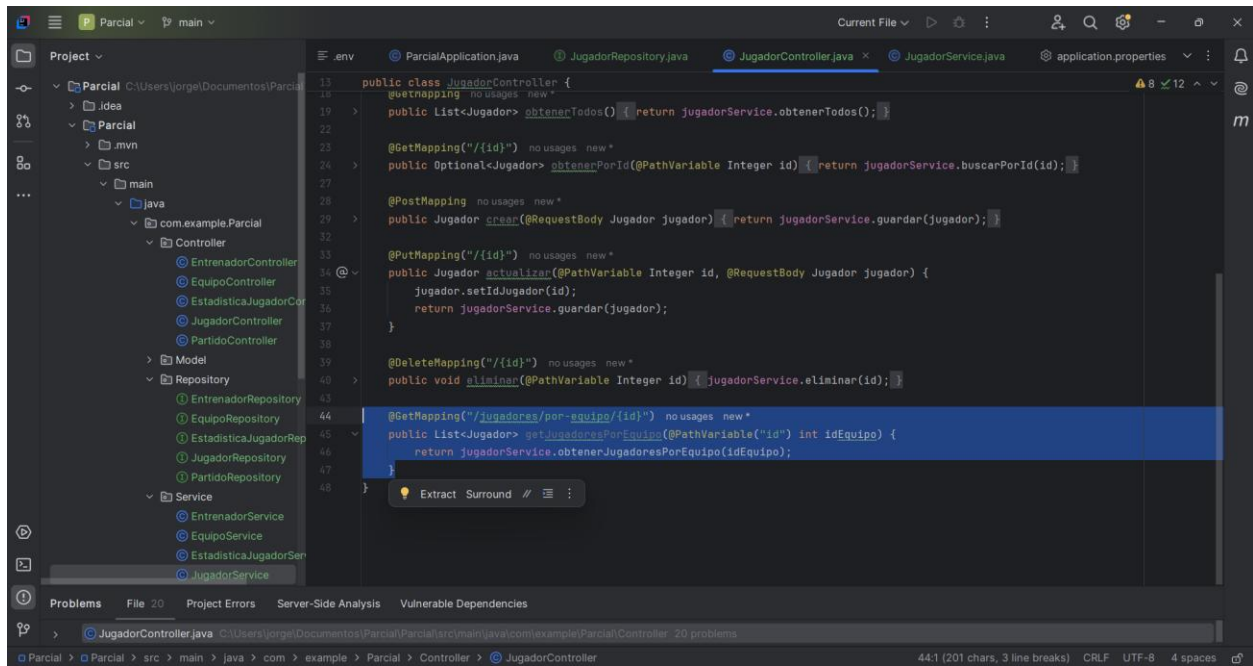




Después en “JugadorService” colocamos lo siguiente para pasar al ultimo paso:



Y seguidamente también vamos agregar el siguiente código al “JugadorController”:



```
13 public class JugadorController {
14     @GetMapping("/")
15     public List<Jugador> obtenerTodos() { return jugadorService.obtenerTodos(); }
16
17     @GetMapping("/{id}")
18     public Optional<Jugador> obtenerPorId(@PathVariable Integer id) { return jugadorService.buscarPorId(id); }
19
20     @PostMapping
21     public Jugador crear(@RequestBody Jugador jugador) { return jugadorService.guardar(jugador); }
22
23     @PutMapping("/{id}")
24     public Jugador actualizar(@PathVariable Integer id, @RequestBody Jugador jugador) {
25         jugador.setIdJugador(id);
26         return jugadorService.guardar(jugador);
27     }
28
29     @DeleteMapping("/{id}")
30     public void eliminar(@PathVariable Integer id) { jugadorService.eliminar(id); }
31
32     @GetMapping("/jugadores/por-equipo/{id}")
33     public List<Jugador> getJugadoresPorEquipo(@PathVariable("id") int idEquipo) {
34         return jugadorService.obtenerJugadoresPorEquipo(idEquipo);
35     }
36 }
```

The screenshot shows an IDE with the project structure on the left, including folders for 'Parcial', 'src', 'main', 'java', 'com.example.Parcial', 'Controller', 'Model', 'Repository', and 'Service'. The 'JugadorController.java' file is open in the editor, showing the code above. The status bar at the bottom indicates the file path and some statistics: 44:1 (201 chars, 3 line breaks) CRLF UTF-8 4 spaces.

Y probamos en el postman: