

Documentación Parcial Arquitectura

Javier Mauricio Montenegro Sierra

Ingeniería de sistemas, séptimo semestre, Corporación Universitaria

Minuto de Dios

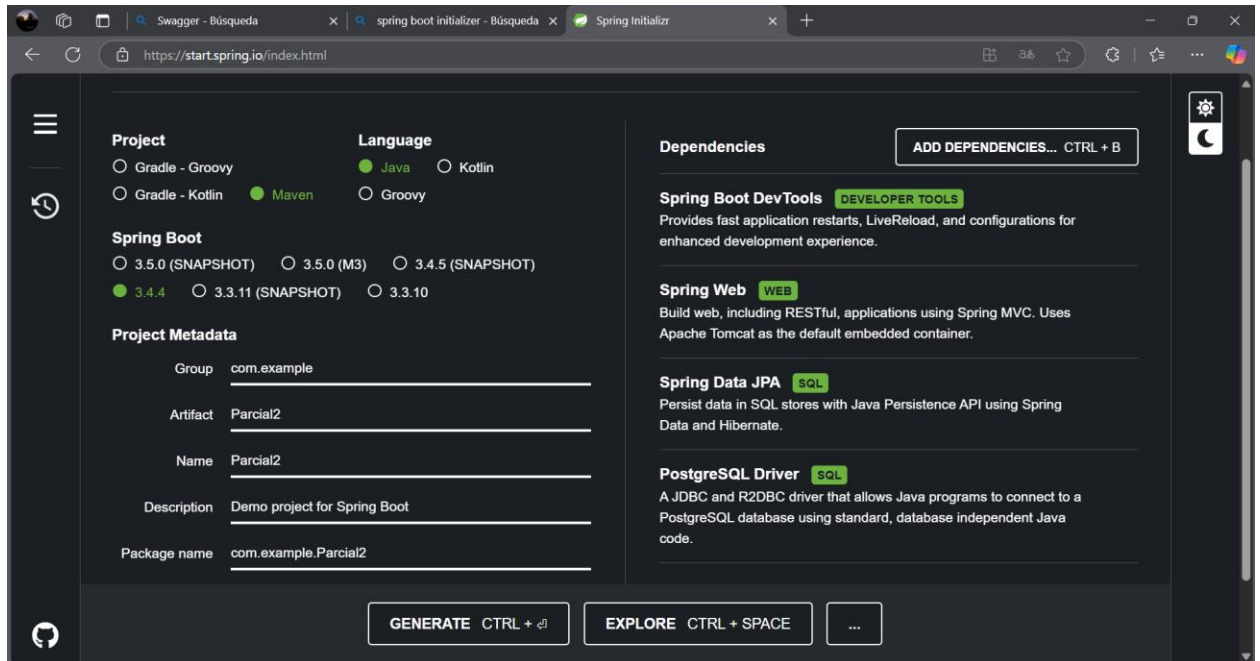
NRC 4064: Arquitectura de Software

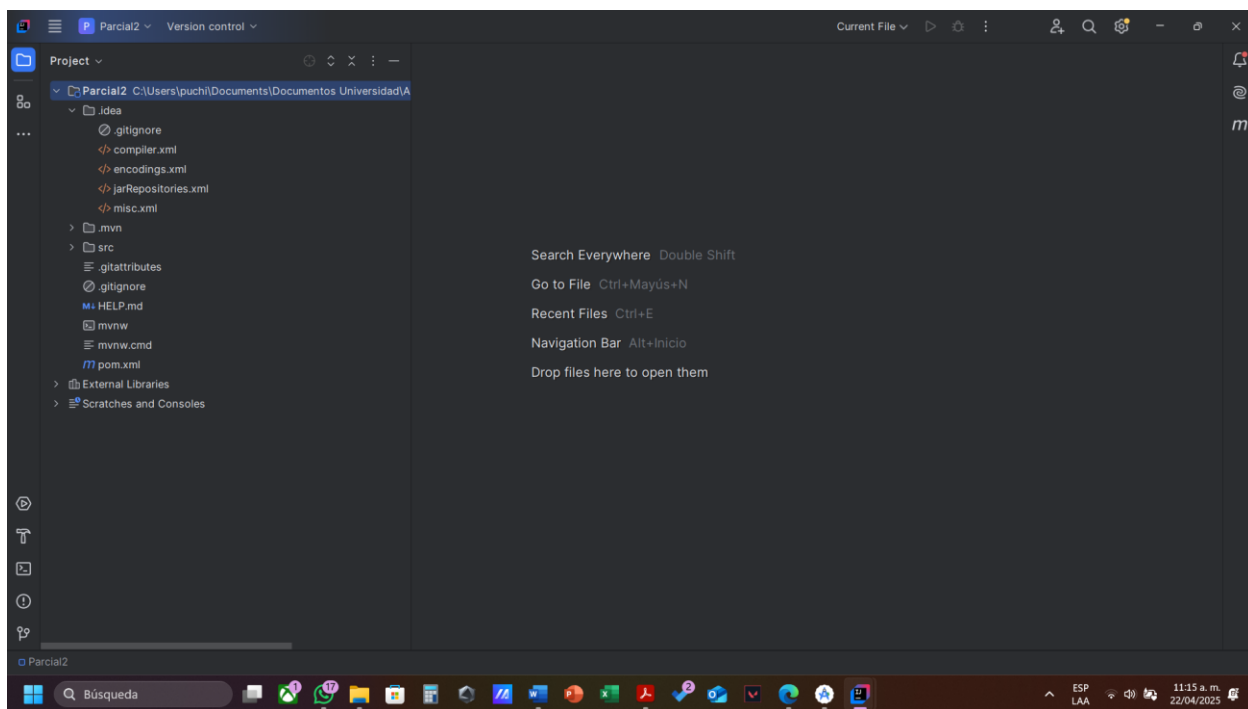
Ing. William Alexander Matallana Porras

22 de abr. de 25

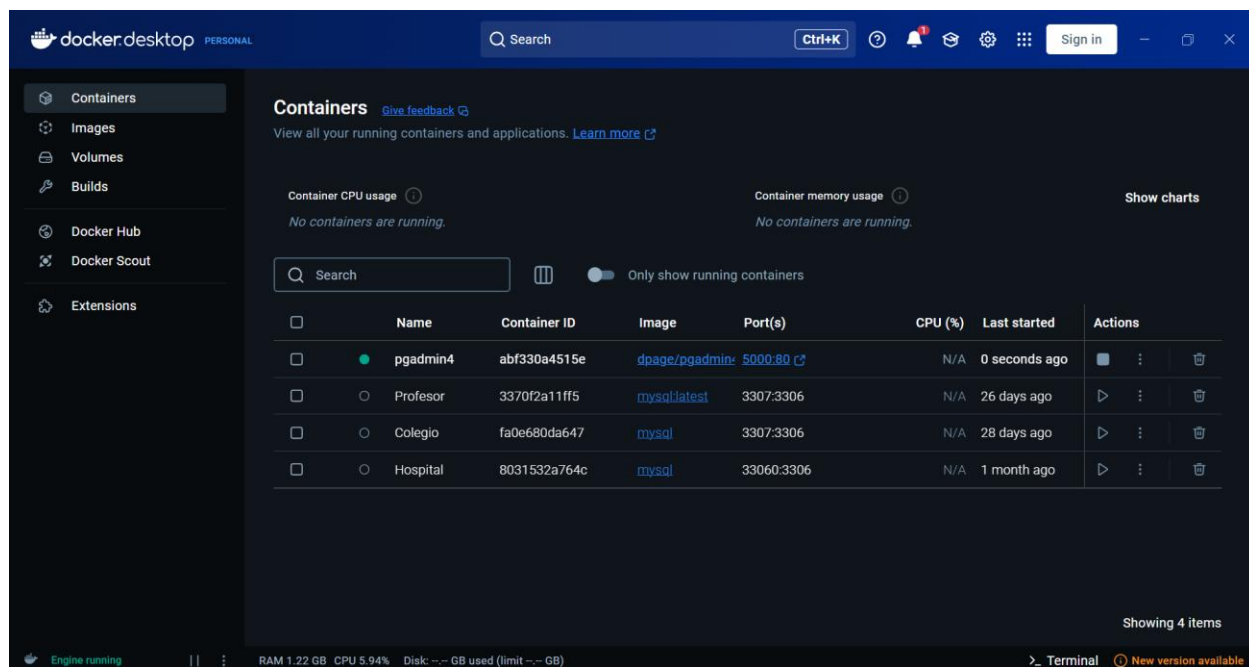
Spring boot, inicialización del proyecto y conexión

1. Vamos a spring boot y buscamos las dependencias que necesitamos y generamos el proyecto

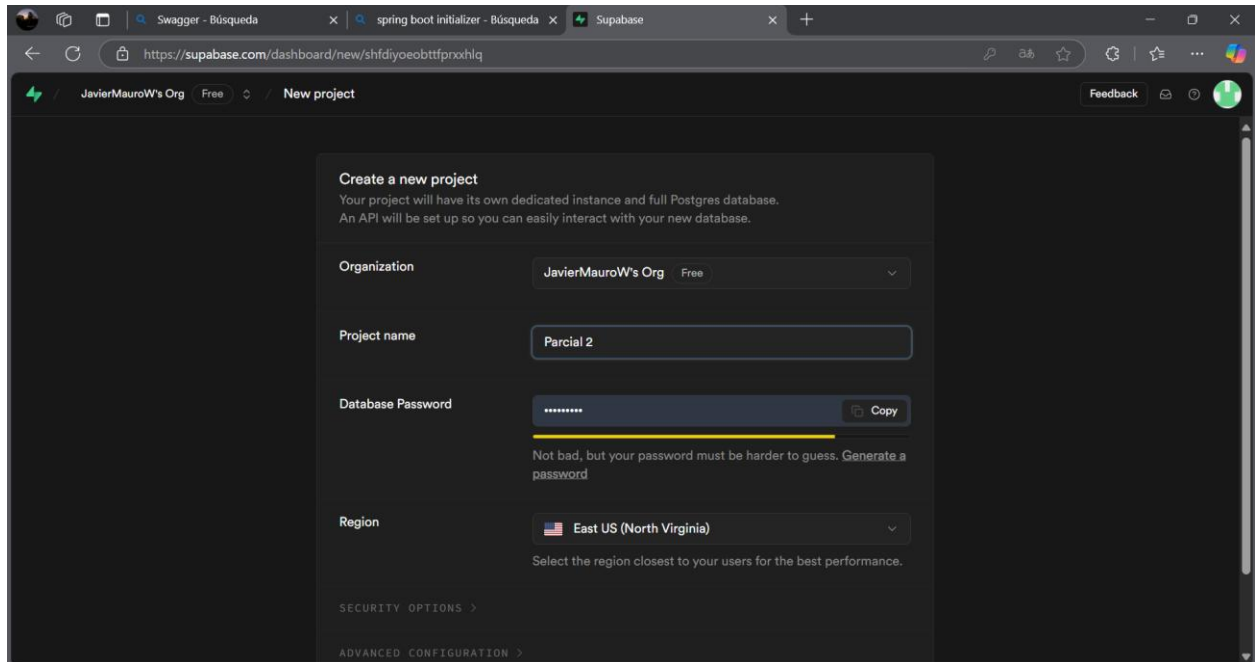




2. Para hacer la conexión primero levantamos el contenedor de Docker



3. Vamos a supabase y creamos un nuevo proyecto

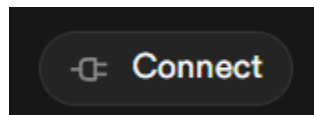


The screenshot shows the Supabase dashboard with the 'Create a new project' form. The form is titled 'Create a new project' and includes the following fields and options:

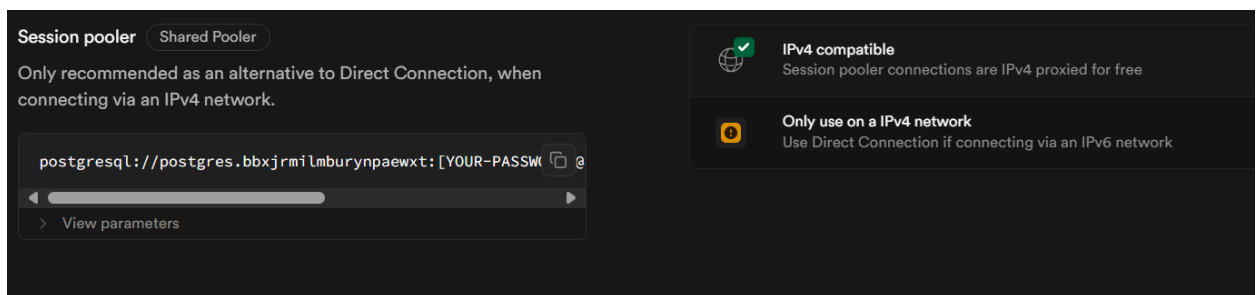
- Organization:** A dropdown menu showing 'JavierMauroW's Org' and 'Free'.
- Project name:** A text input field containing 'Parcial 2'.
- Database Password:** A password input field with a 'Copy' button. Below the field, a message reads: 'Not bad, but your password must be harder to guess. Generate a password'.
- Region:** A dropdown menu showing 'East US (North Virginia)'.

Below the main form, there are links for 'SECURITY OPTIONS >' and 'ADVANCED CONFIGURATION >'.

4. Luego de que se cree le damos en:



5. Hay que ir hasta abajo en Session pooler y se copia lo que nos da



The screenshot shows the 'Session pooler' configuration page. It includes the following information:

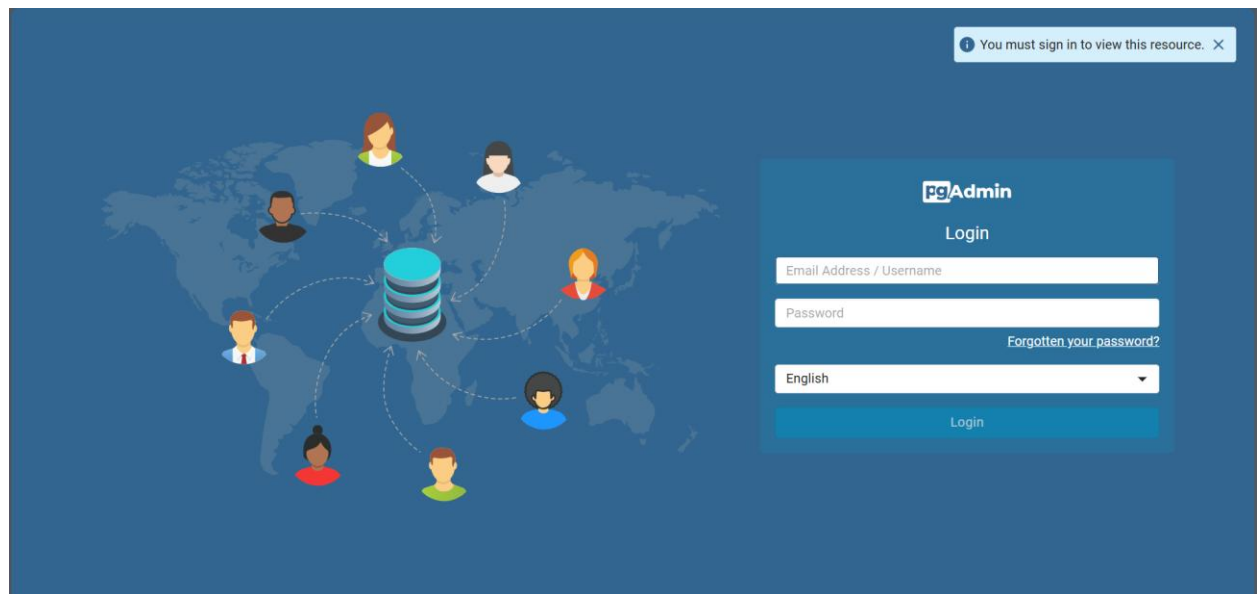
- Session pooler:** A toggle switch for 'Shared Pooler'.
- Only recommended as an alternative to Direct Connection, when connecting via an IPv4 network.**
- PostgreSQL connection string:** A text input field containing 'postgresql://postgres.bbxjrmilmburynpaewxt:[YOUR-PASSW@'.
- View parameters:** A button to view the parameters.
- IPv4 compatible:** A green checkmark icon and the text 'IPv4 compatible'. Below it, it says 'Session pooler connections are IPv4 proxied for free'.
- Only use on a IPv4 network:** A yellow warning icon and the text 'Only use on a IPv4 network'. Below it, it says 'Use Direct Connection if connecting via an IPv6 network'.

6. Volvemos al Docker y damos click sobre el puerto:

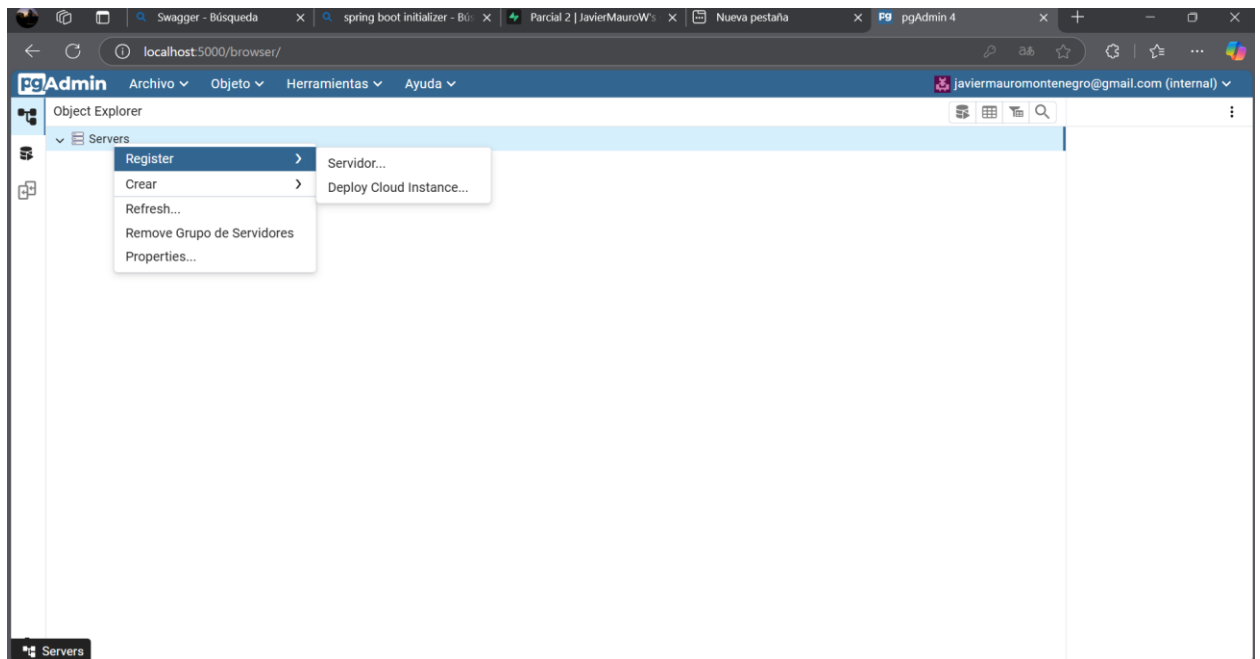
[dpage/pgadmin4](#) [5000:80](#) ↗

7. Esto nos llevara a pgadmin y nos logueamos con las credenciales que pusimos al crear el contenedor la primera vez

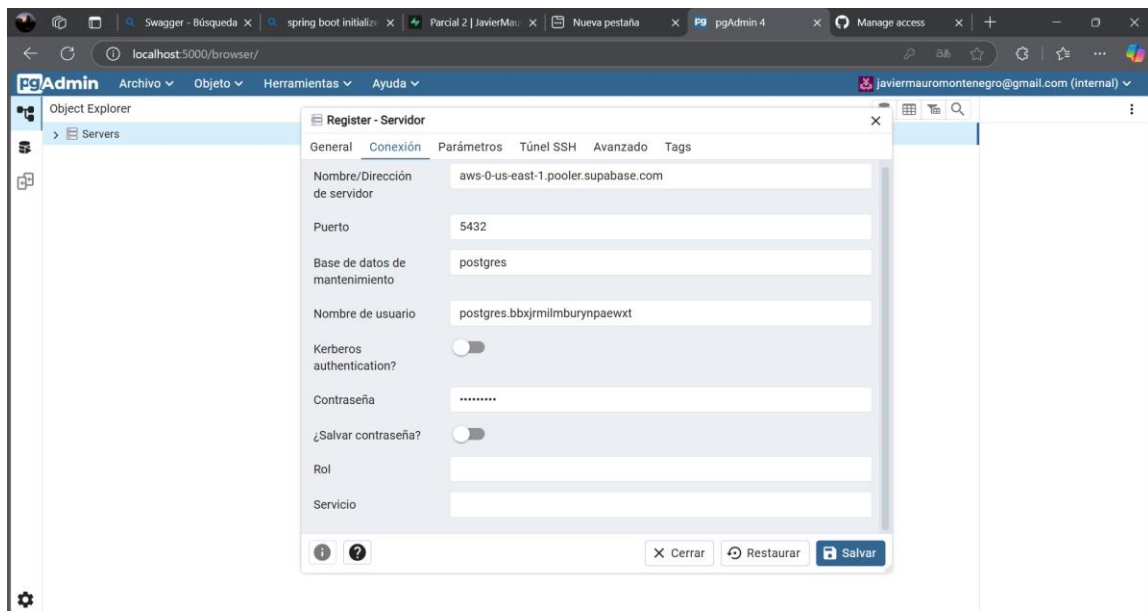
```
PS C:\Users\puchi> docker run -d --name pgadmin4 -p 5000:80 -e PGADMIN_DEFAULT_EMAIL=javiermauromontenegro@gmail.com -e PGADMIN_DEFAULT_PASSWORD=Mo.408862 dpage/pgadmin4  
abf330a4515effcf21925cc45d573b322b7d5cbb96dea4f5c0553ef225aa6a4f
```



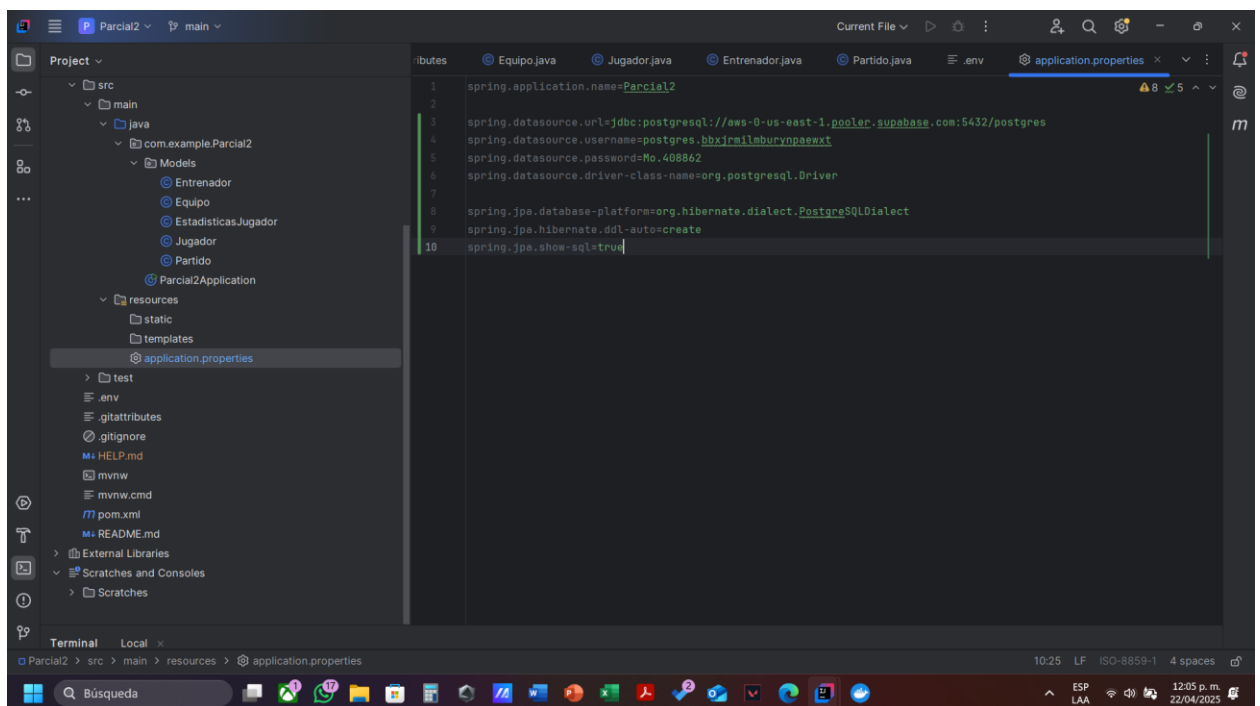
8. Posteriormente nos llevara a la interfaz principal de pgadmin, acá debemos de dar click derecho en servers y luego en registrar



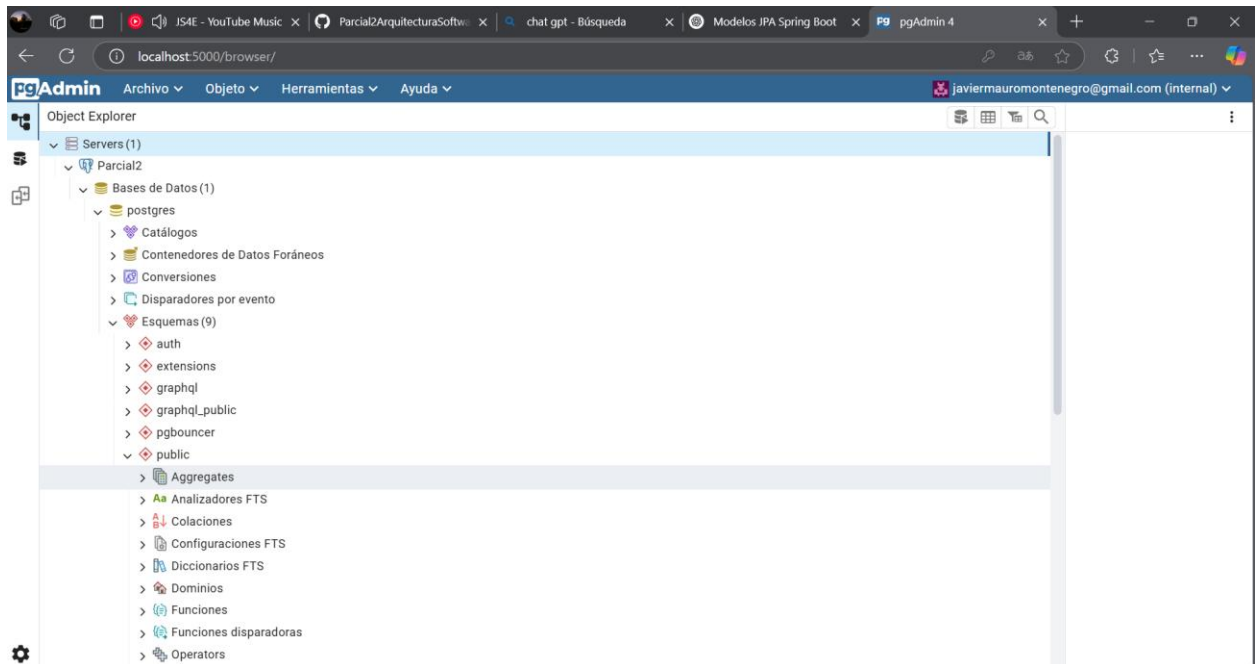
9. Le ponemos un nombre al servidor y luego nos dirigimos a conexión. En este apartado debemos de llenarlos de acuerdo con el link que copiamos en el supabase y quedaría tal que así:



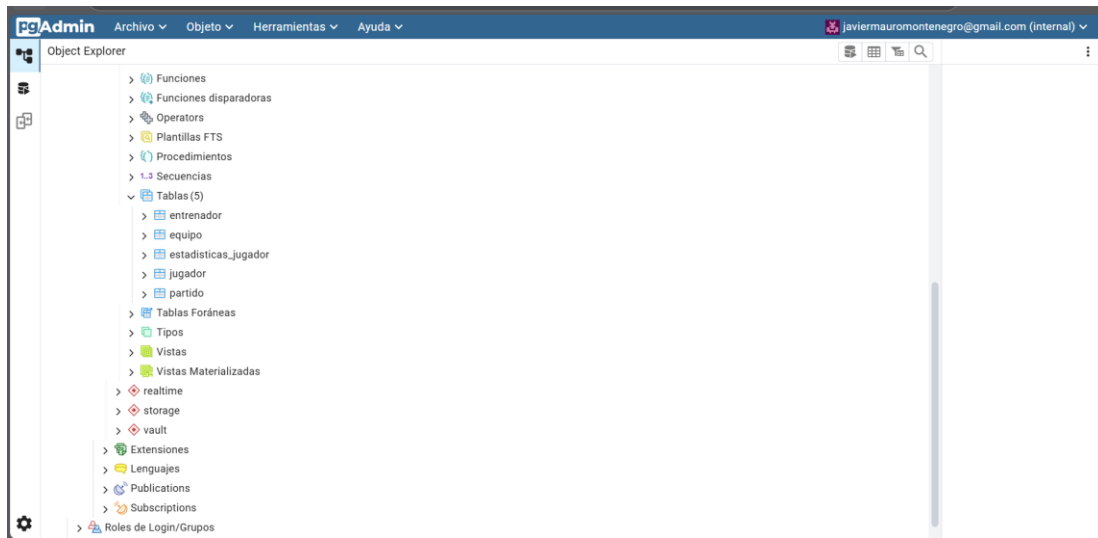
10. Ahora vamos a intelliJ y luego al archivo de application.properties y ingresamos los datos necesarios para la conexión



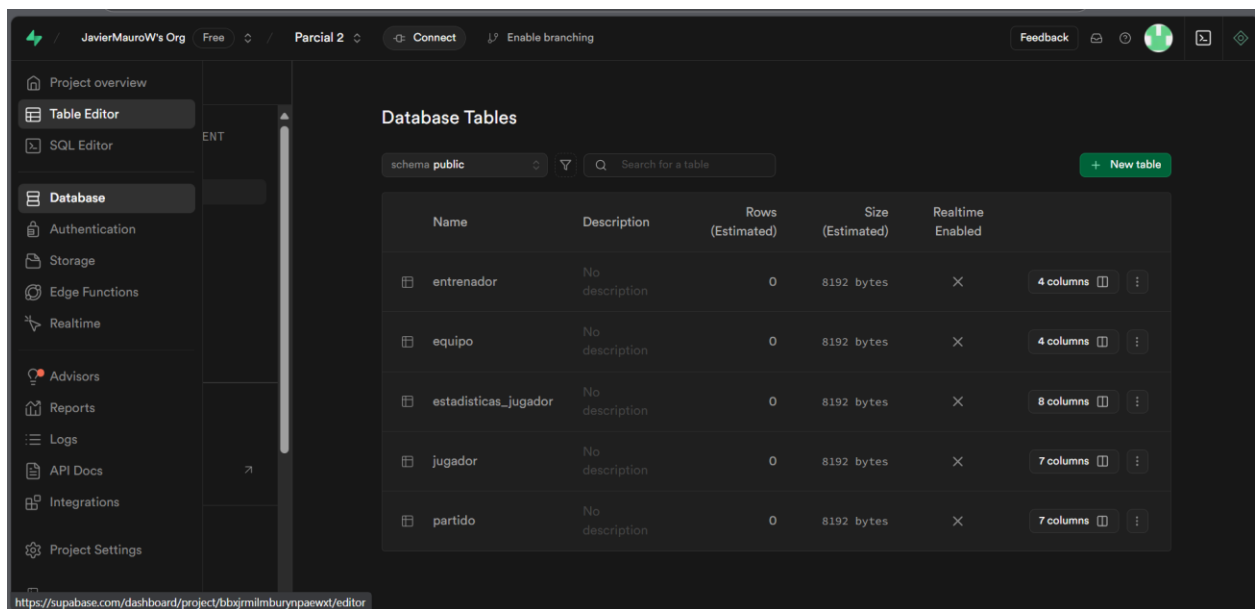
11. Ponemos a correr el proyecto y salvamos la conexión en el pgadmin al hacer esto
ya tendríamos el servidor



12. Vamos a esquema, luego public y por último a tablas y si no ha habido ningún
problema con la conexión deberían salir las tablas creadas, en este caso si se hizo
de manera correcta la conexión



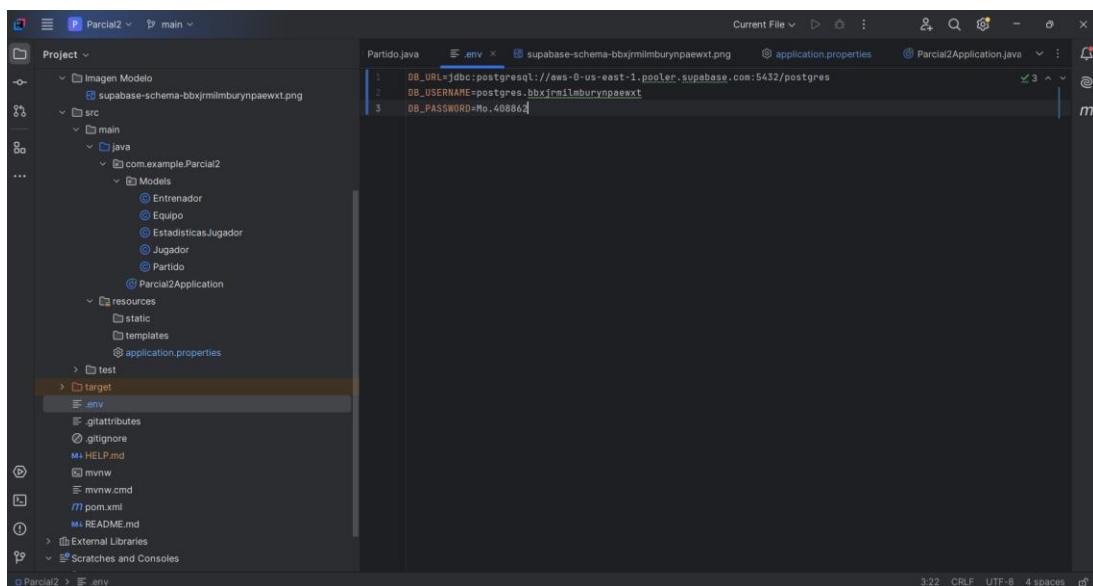
13. Si vamos a Supabase y vamos a database de igual manera deberían de salir las tablas creadas



14. En el supabase también podemos observar el modelo relacional



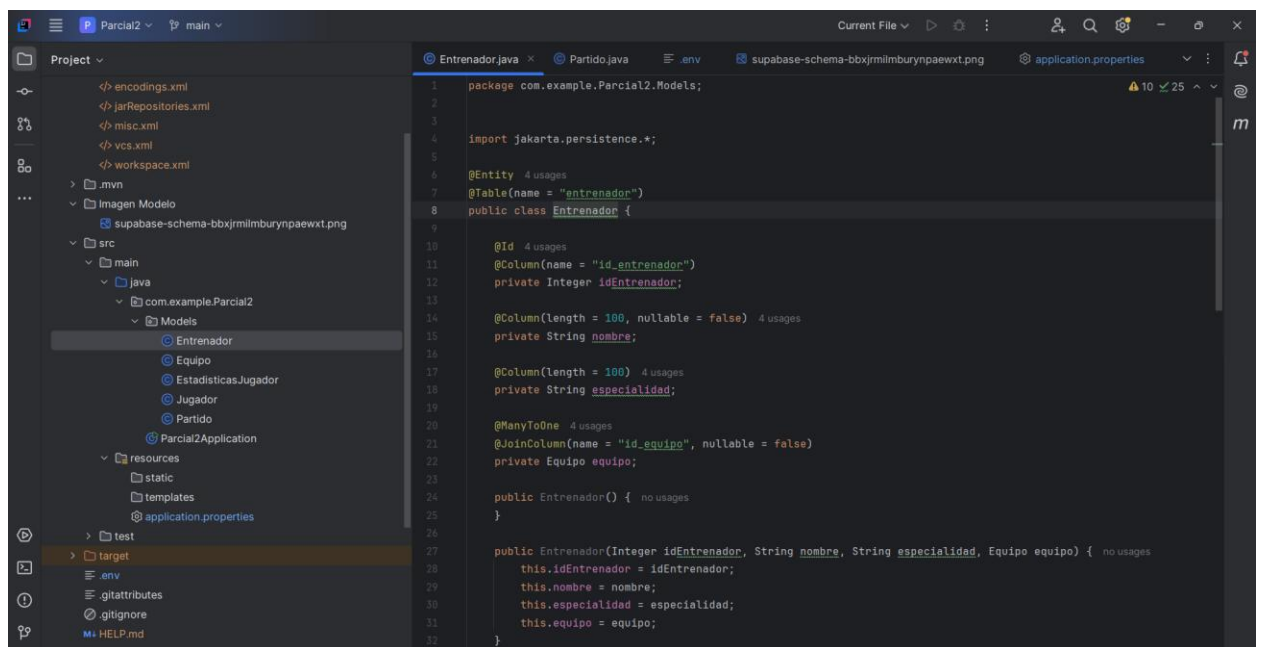
15. Por último en esta parte de conexión se creó un .env que lo que hace es configurar sin escribir datos sensibles en el código fuente. Así, si subes tu proyecto a GitHub o lo compartes, no expones credenciales o configuraciones privadas.



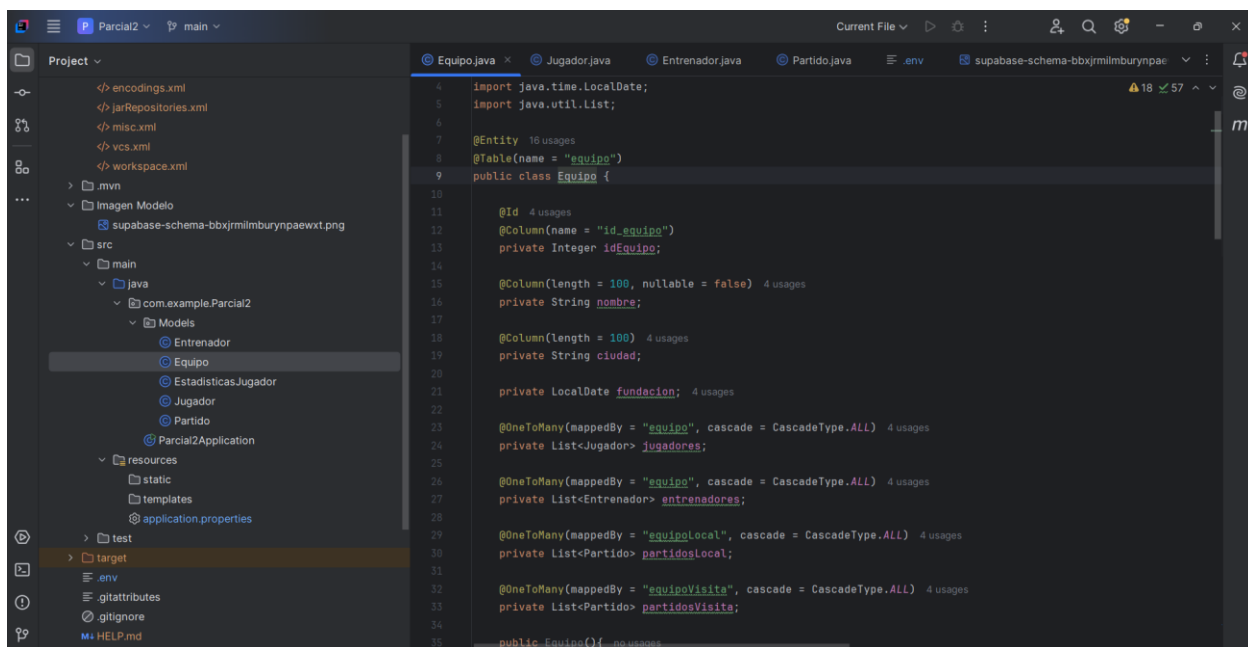
Modelos

Todas las capturas de pantalla se enfocan en mostrar las relaciones.

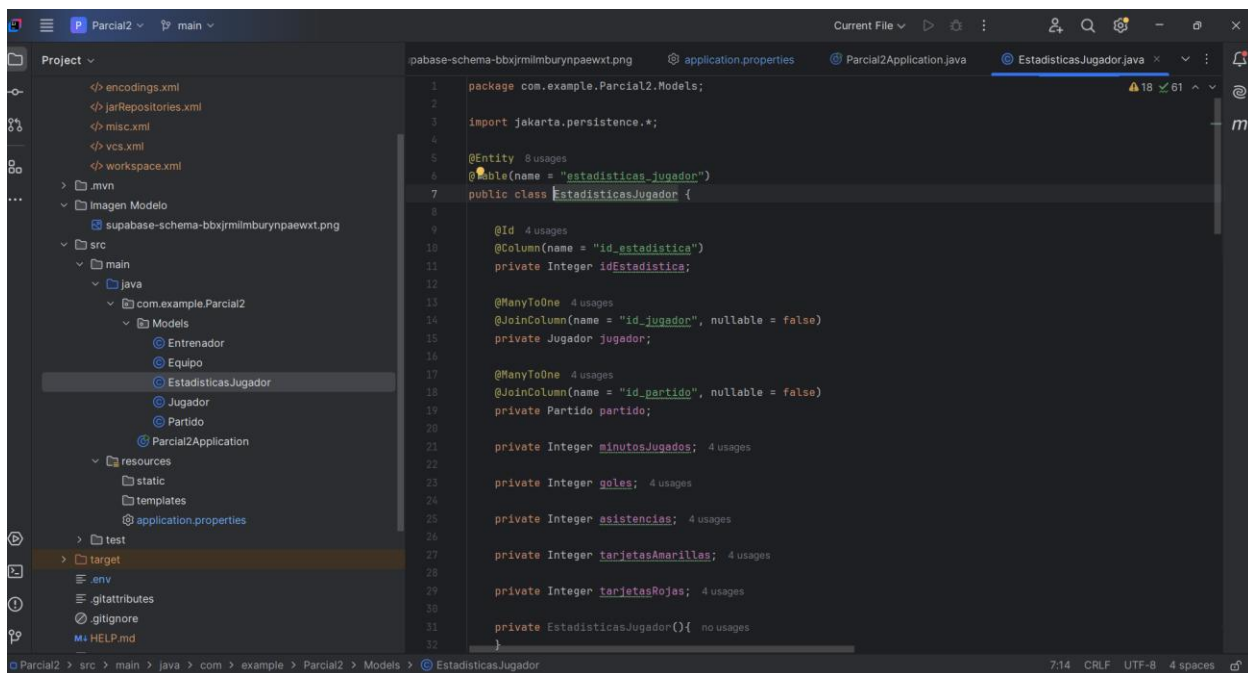
Entrenador:



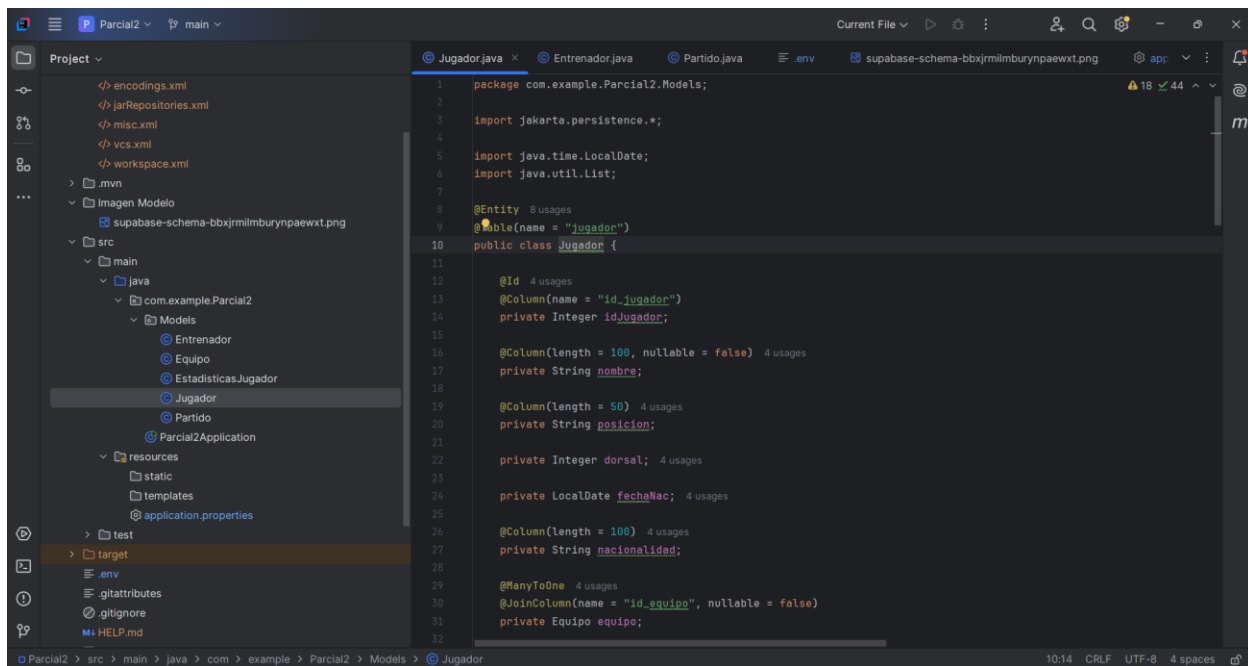
Equipo:



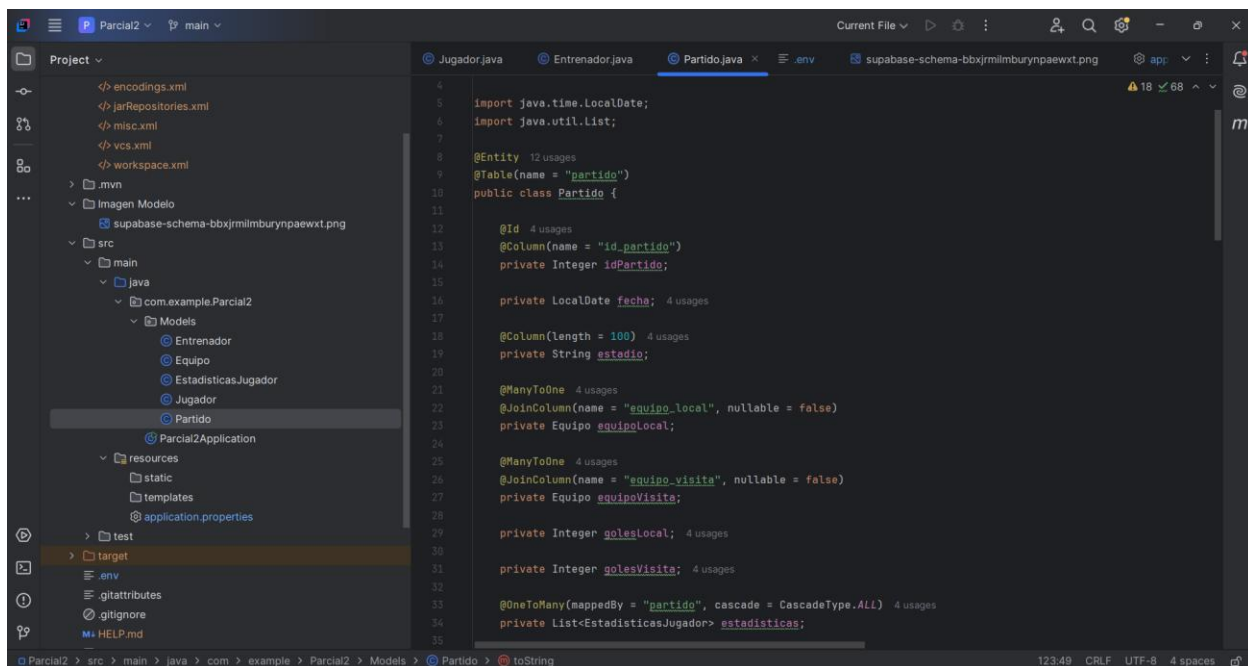
Estadísticas Jugador:



Jugador

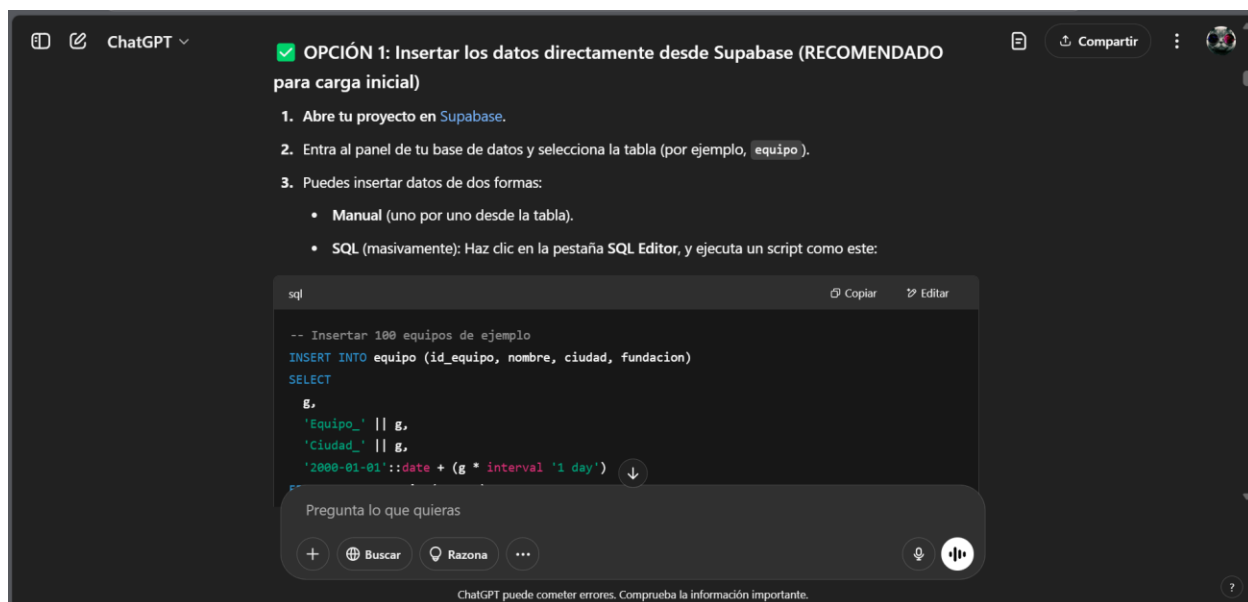


Partido:



Inserción de registros

1. Se le pide a chat gpt que nos genere 100 registros por cada tabla



2. Vamos a supabase damos en SQL editor y ahí copiamos los registros que nos de chat, luego nos dirigimos a table editor y ahí nos salen las tablas con los registros

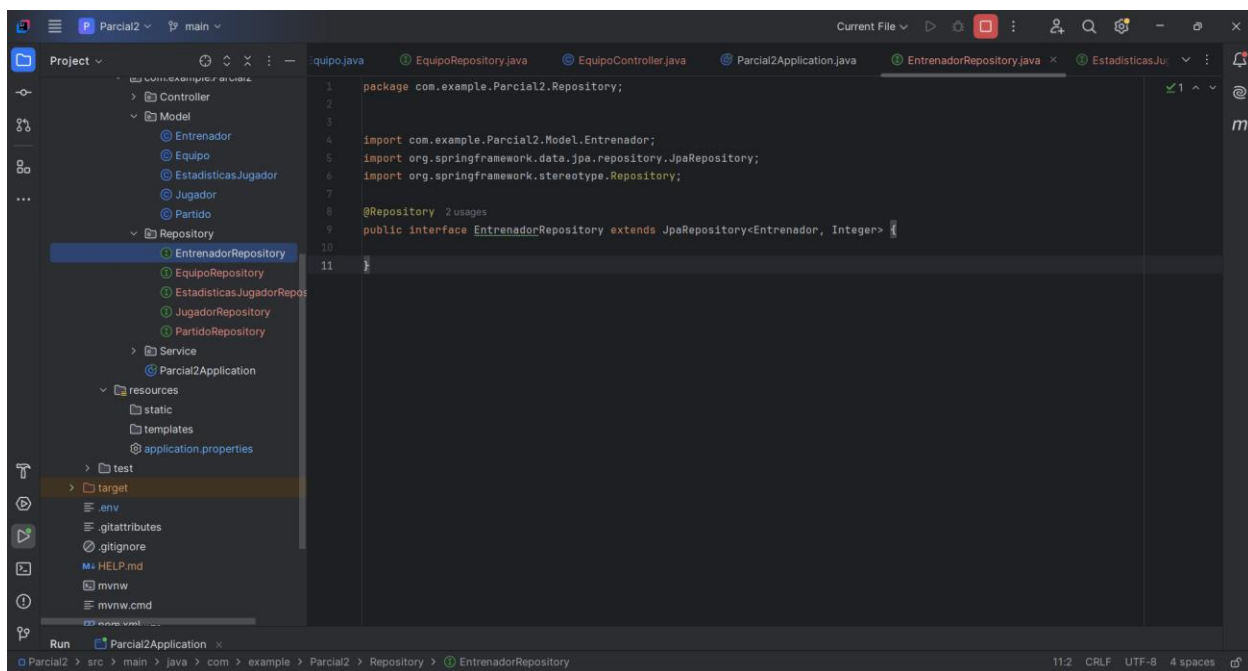
Supabase Table Editor interface showing a table named 'entrenador' with 14 rows of data.

id_entrenador	id_equipo	especialidad	nombre
1	33	Asistente	Entrenador_1
2	9	Preparador Físico	Entrenador_2
3	83	Principal	Entrenador_3
4	8	Asistente	Entrenador_4
5	69	Preparador Físico	Entrenador_5
6	76	Principal	Entrenador_6
7	55	Asistente	Entrenador_7
8	29	Preparador Físico	Entrenador_8
9	56	Principal	Entrenador_9
10	81	Asistente	Entrenador_10
11	43	Preparador Físico	Entrenador_11
12	49	Principal	Entrenador_12
13	16	Asistente	Entrenador_13
14	69	Preparador Físico	Entrenador_14

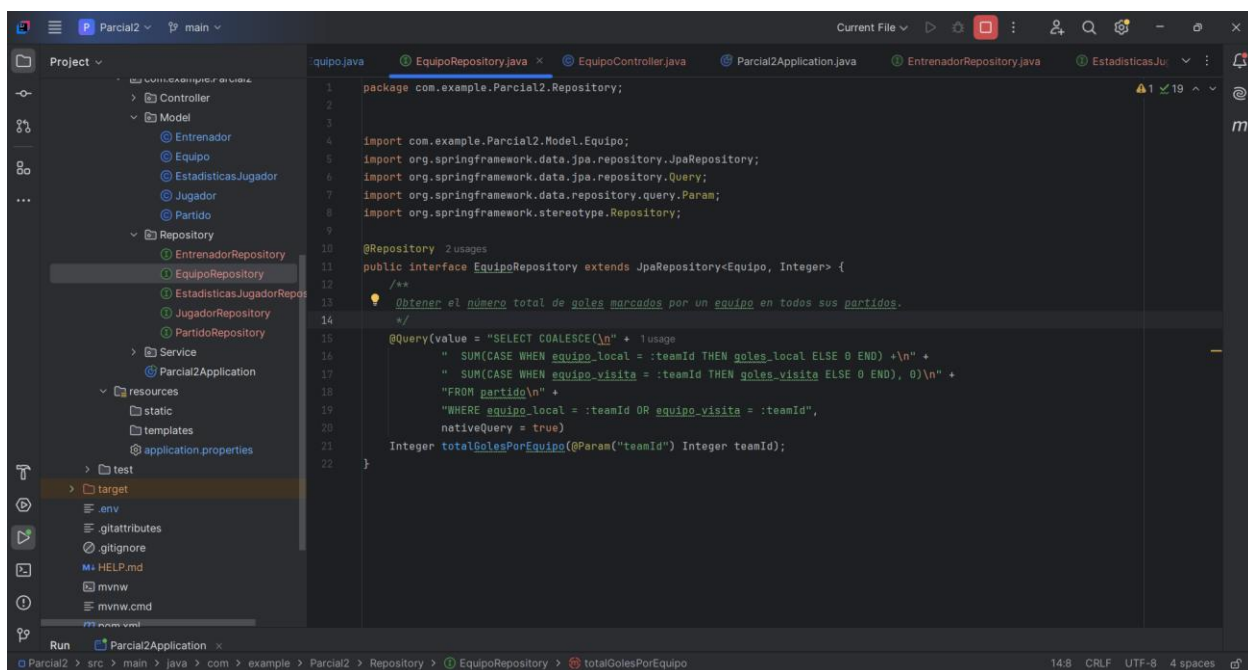
Page 1 of 1 | 100 rows | 100 records

Repositorios con consultas nativas incluidas

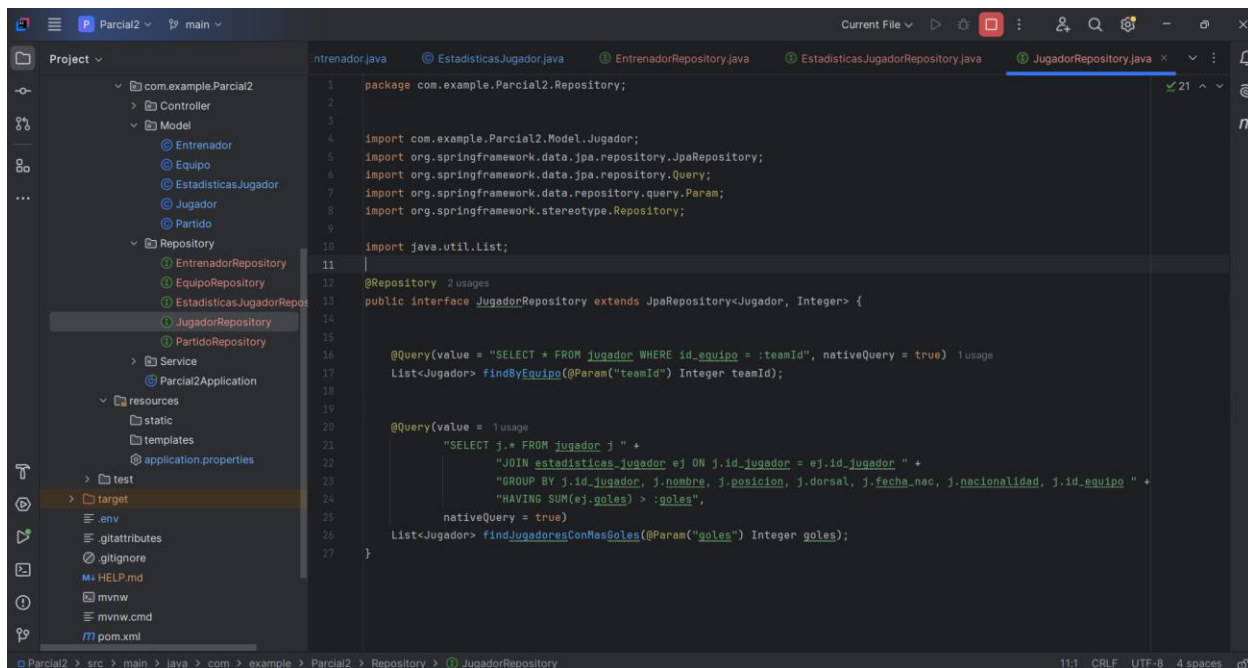
Entrenador Repositorio:



Equipo Repositorio:



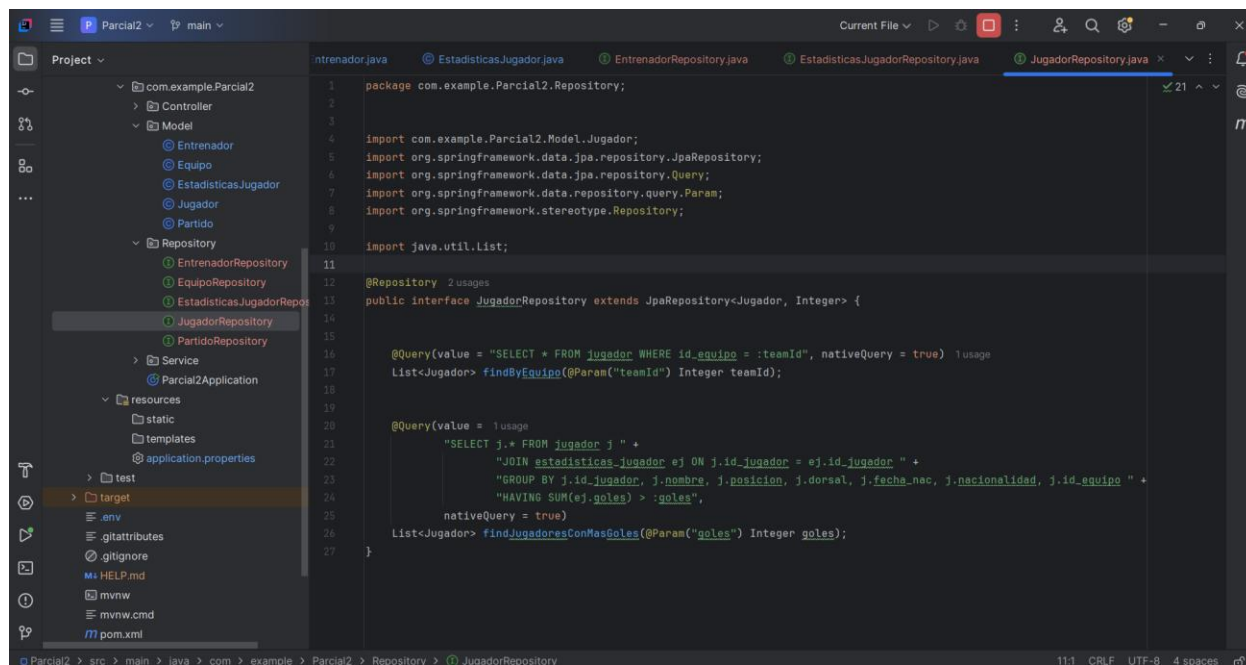
Estadísticas Jugador repositorio:



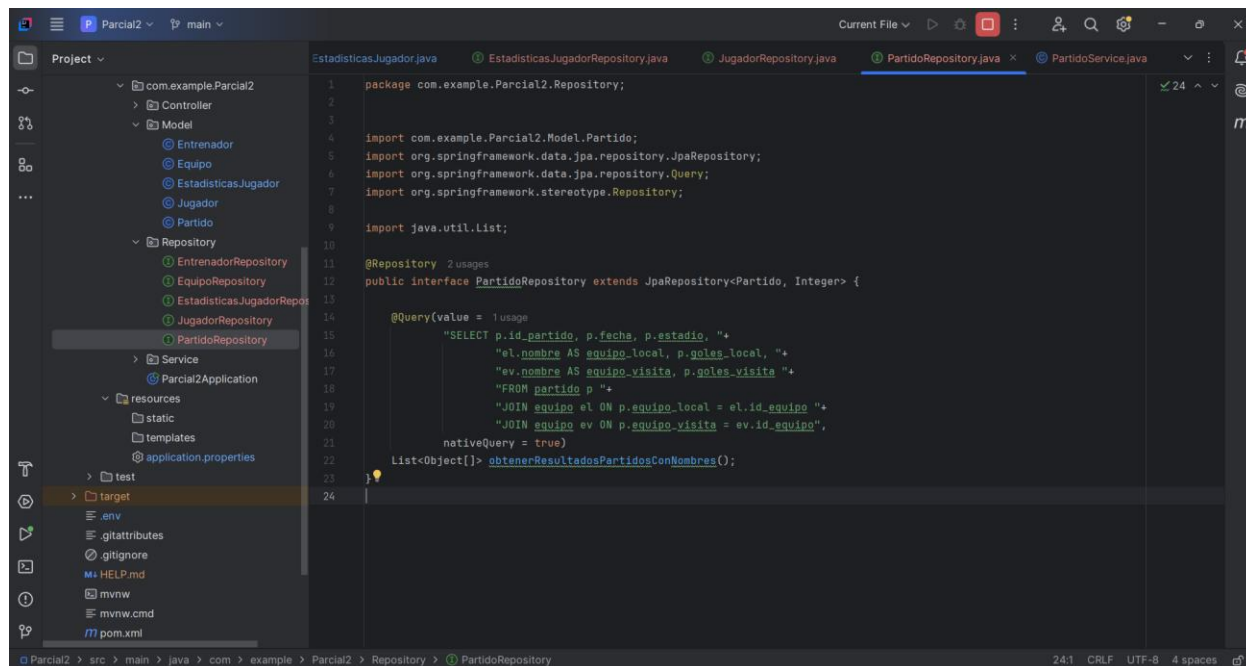
The screenshot shows an IDE with a project named 'Parcial2'. The left sidebar displays the project structure, including a 'Repository' package with several interfaces. The 'JugadorRepository' interface is selected. The main editor shows the implementation of this interface in 'JugadorRepository.java'. The code defines two methods: 'findByEquipo' and 'findJugadoresConMasGoles', both using JPA @Query annotations to interact with a database.

```
1 package com.example.Parcial2.Repository;
2
3
4 import com.example.Parcial2.Model.Jugador;
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.data.jpa.repository.Query;
7 import org.springframework.data.repository.query.Param;
8 import org.springframework.stereotype.Repository;
9
10 import java.util.List;
11
12 @Repository
13 public interface JugadorRepository extends JpaRepository<Jugador, Integer> {
14
15
16     @Query(value = "SELECT * FROM jugador WHERE id_equipo = :teamId", nativeQuery = true)
17     List<Jugador> findByEquipo(@Param("teamId") Integer teamId);
18
19
20     @Query(value = "SELECT j.* FROM jugador j " +
21             "JOIN estadisticas_jugador ej ON j.id_jugador = ej.id_jugador " +
22             "GROUP BY j.id_jugador, j.nombre, j.posicion, j.dorsal, j.fecha_nac, j.nacionalidad, j.id_equipo " +
23             "HAVING SUM(ej.goles) > :goles",
24             nativeQuery = true)
25     List<Jugador> findJugadoresConMasGoles(@Param("goles") Integer goles);
26
27 }
```


Jugador Repositorio:



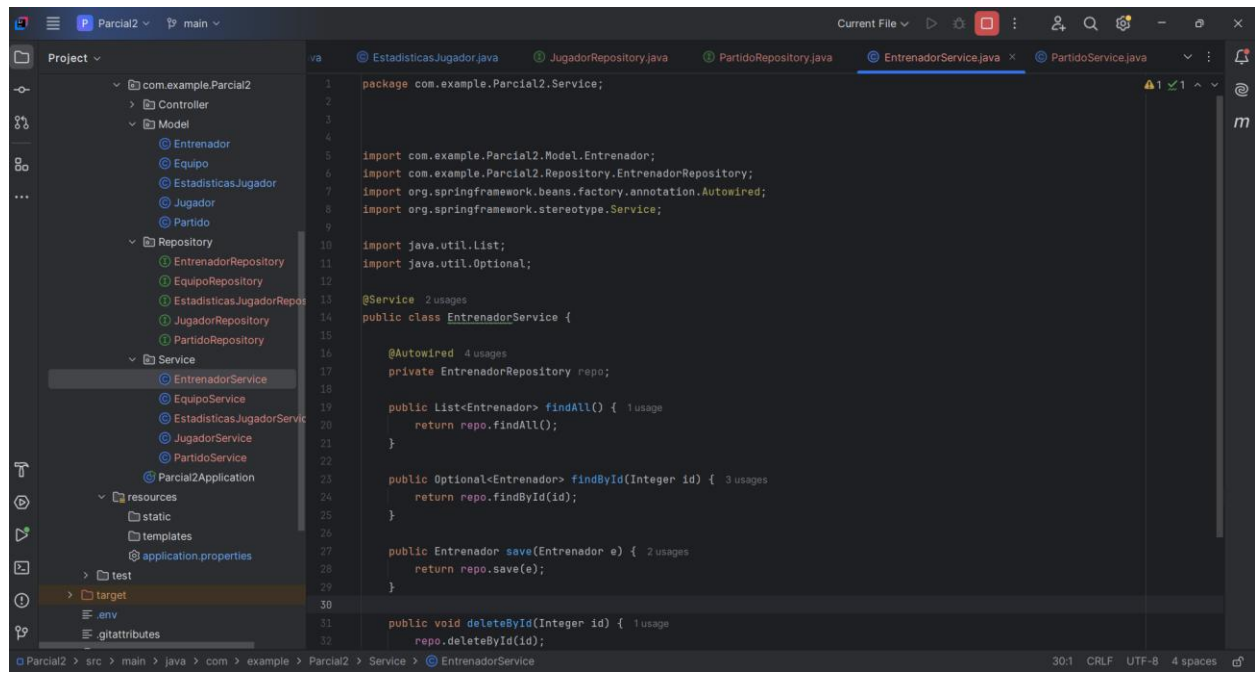
Partido Repositorio:



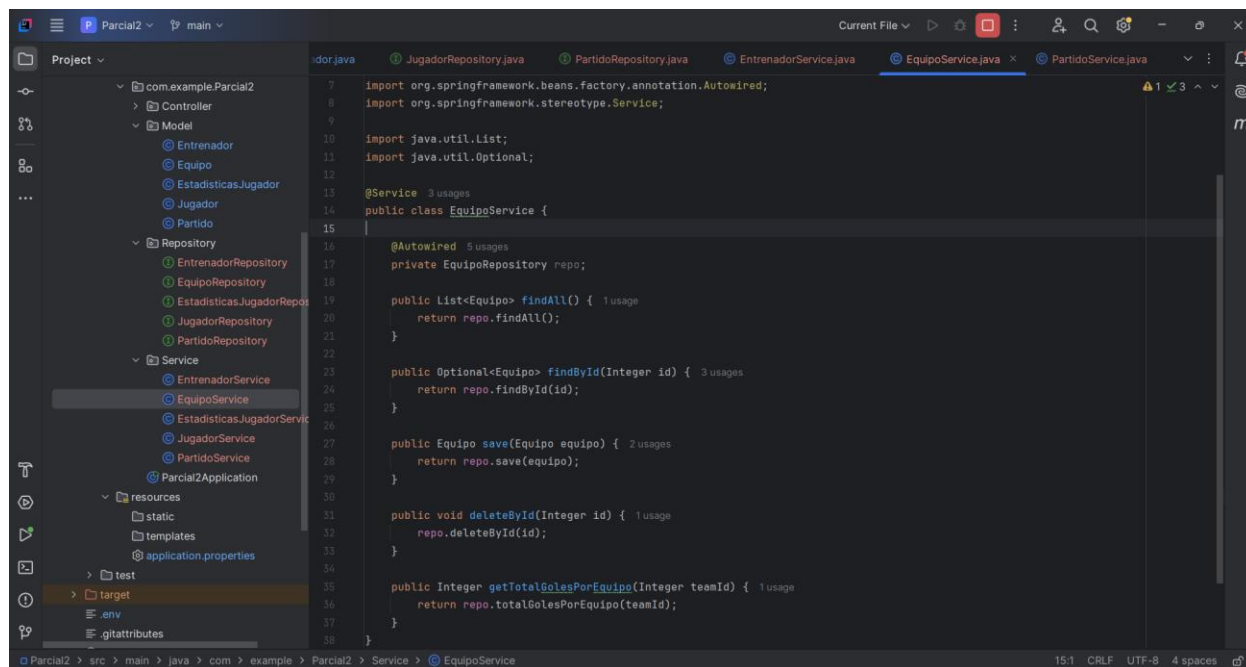
```
1 package com.example.Parcial2.Repository;
2
3
4 import com.example.Parcial2.Model.Partido;
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.data.jpa.repository.Query;
7 import org.springframework.stereotype.Repository;
8
9 import java.util.List;
10
11 @Repository
12 public interface PartidoRepository extends JpaRepository<Partido, Integer> {
13
14     @Query(value = "SELECT p.id_partido, p.fecha, p.estadio, " +
15         "el.nombre AS equipo_local, p.goles_local, " +
16         "ev.nombre AS equipo_visitante, p.goles_visitante " +
17         "FROM partido p " +
18         "JOIN equipo el ON p.equipo_local = el.id_equipo " +
19         "JOIN equipo ev ON p.equipo_visitante = ev.id_equipo",
20         nativeQuery = true)
21     List<Object> obtenerResultadosPartidosConNombres();
22 }
23
24
```

Servicios

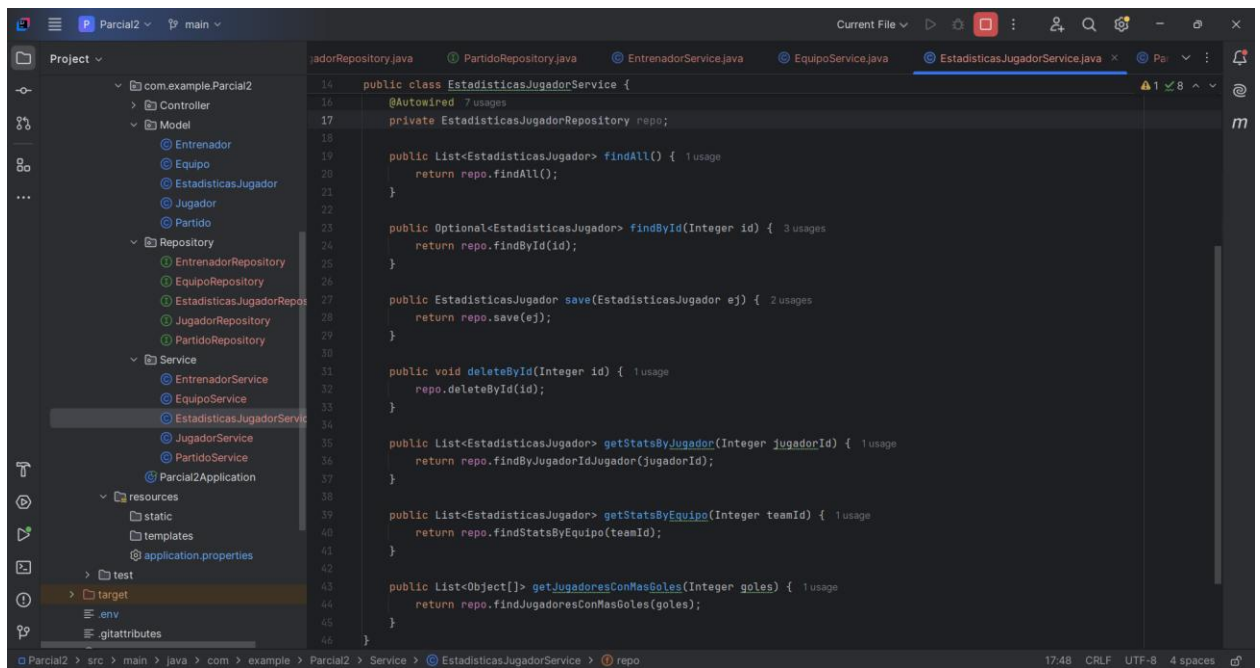
Entrenador Servicio:



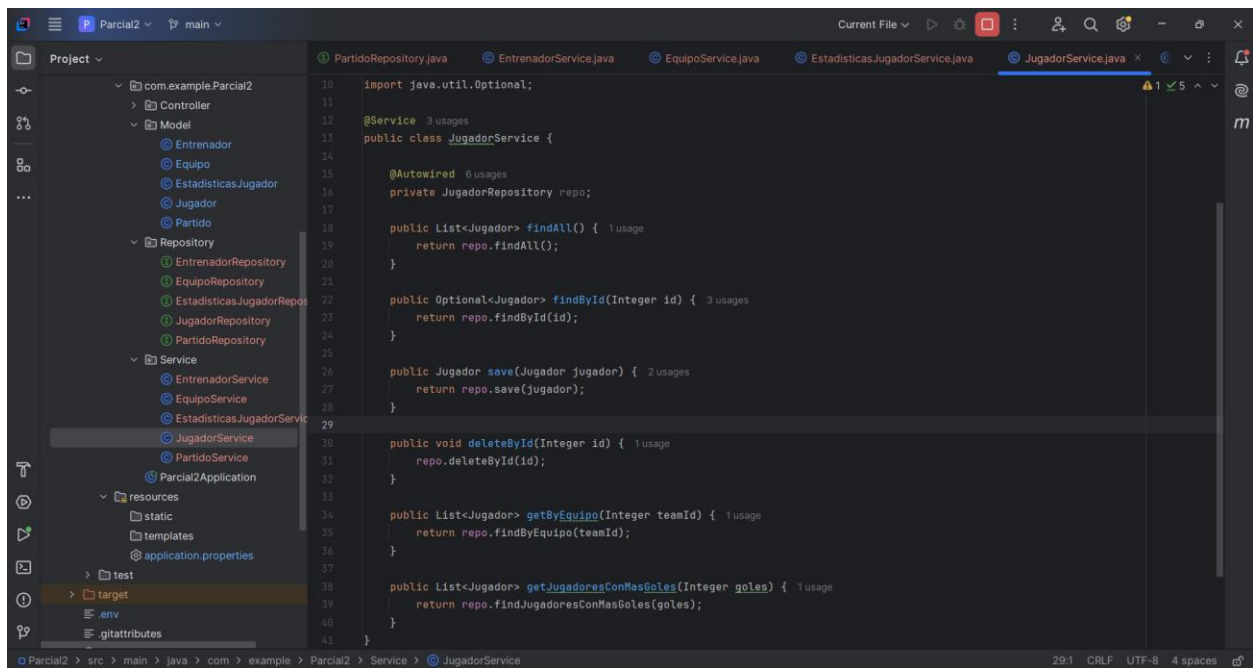
Equipo servicio:



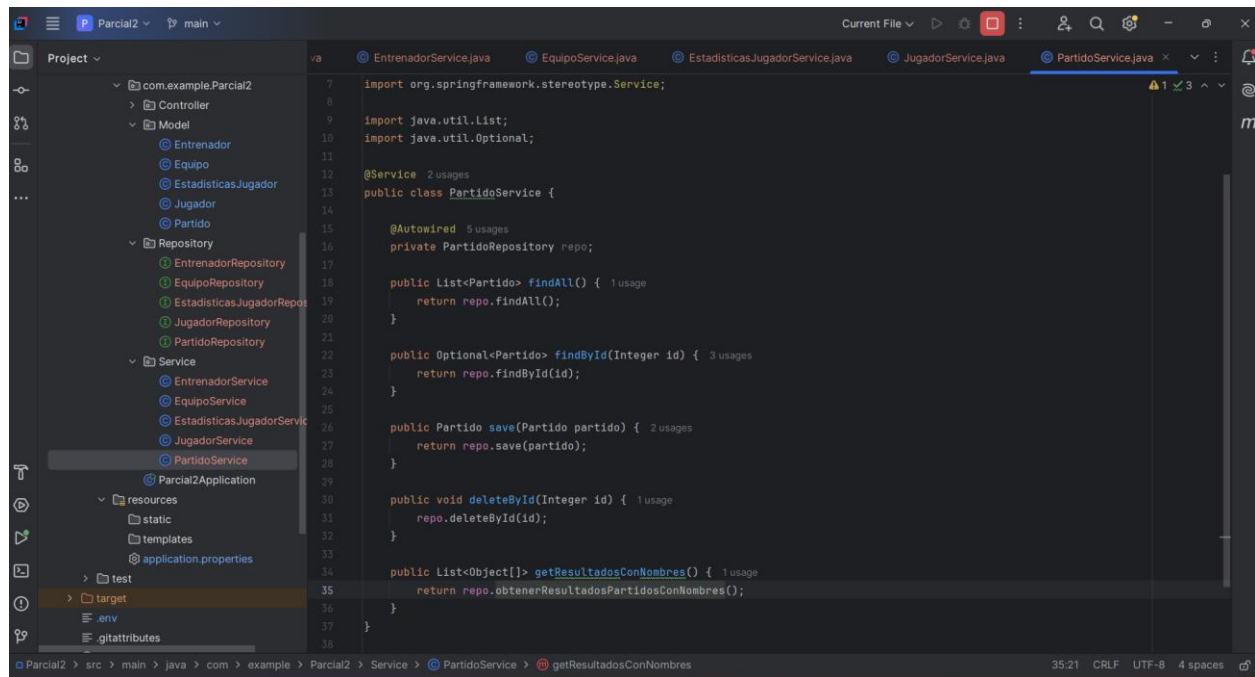
Estadísticas Jugador Servicio:



Jugador Service

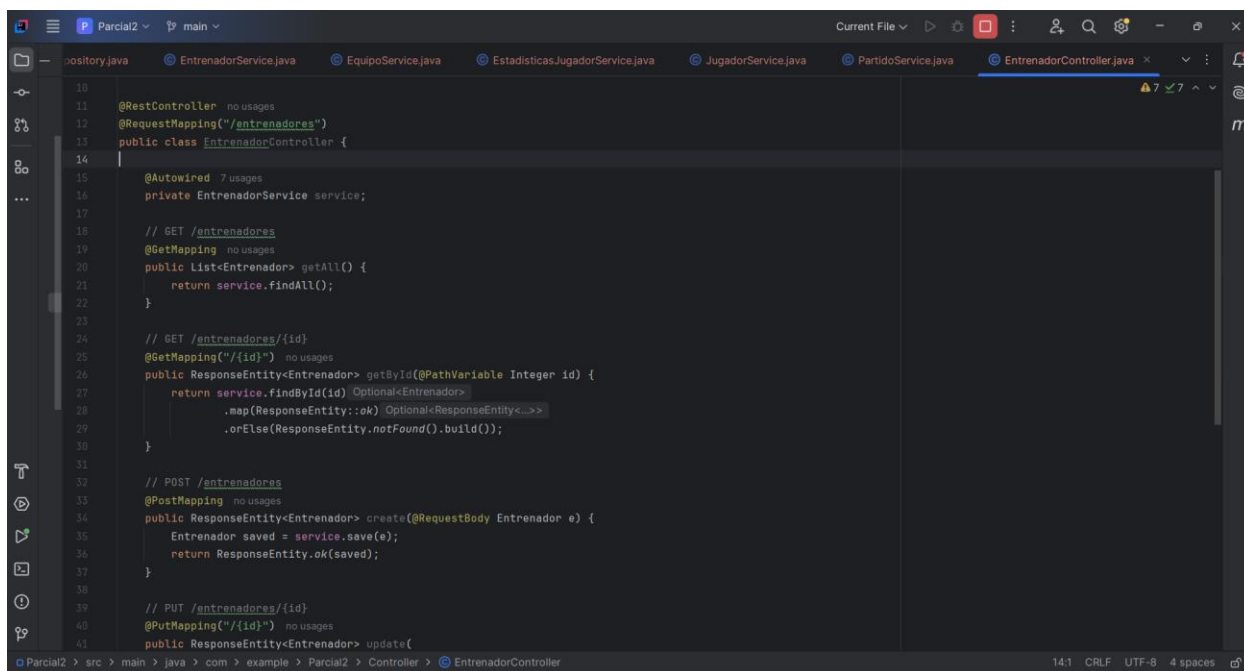


Partido Servicio:



Controladores

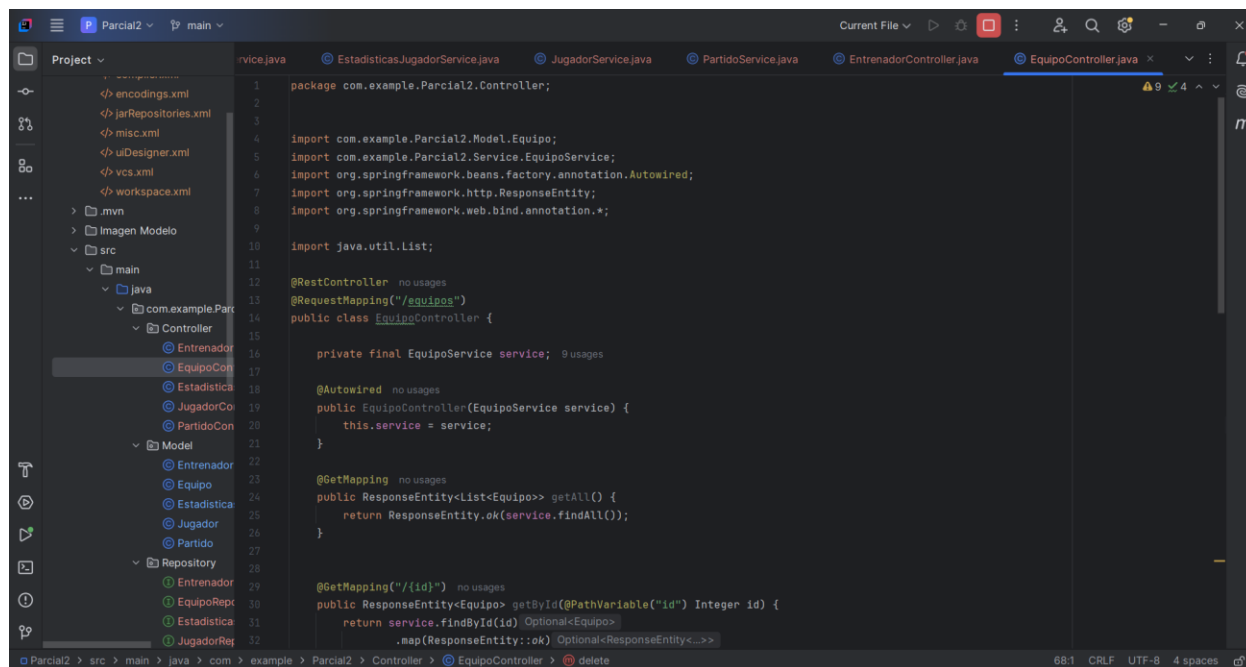
Entrenador controlador:



```
10
11 @RestController no usages
12 @RequestMapping("/entrenadores")
13 public class EntrenadorController {
14
15     @Autowired 7 usages
16     private EntrenadorService service;
17
18     // GET /entrenadores
19     @GetMapping no usages
20     public List<Entrenador> getAll() {
21         return service.findAll();
22     }
23
24     // GET /entrenadores/{id}
25     @GetMapping("/{id}") no usages
26     public ResponseEntity<Entrenador> getById(@PathVariable Integer id) {
27         return service.findById(id).Optional<Entrenador>
28             .map(ResponseEntity::ok).Optional<ResponseEntity<-->>
29             .orElse(ResponseEntity.notFound().build());
30     }
31
32     // POST /entrenadores
33     @PostMapping no usages
34     public ResponseEntity<Entrenador> create(@RequestBody Entrenador e) {
35         Entrenador saved = service.save(e);
36         return ResponseEntity.ok(saved);
37     }
38
39     // PUT /entrenadores/{id}
40     @PutMapping("/{id}") no usages
41     public ResponseEntity<Entrenador> update(
```

Parcial2 > src > main > java > com > example > Parcial2 > Controller > EntrenadorController 14:1 CRLF UTF-8 4 spaces

Equipo controlador:

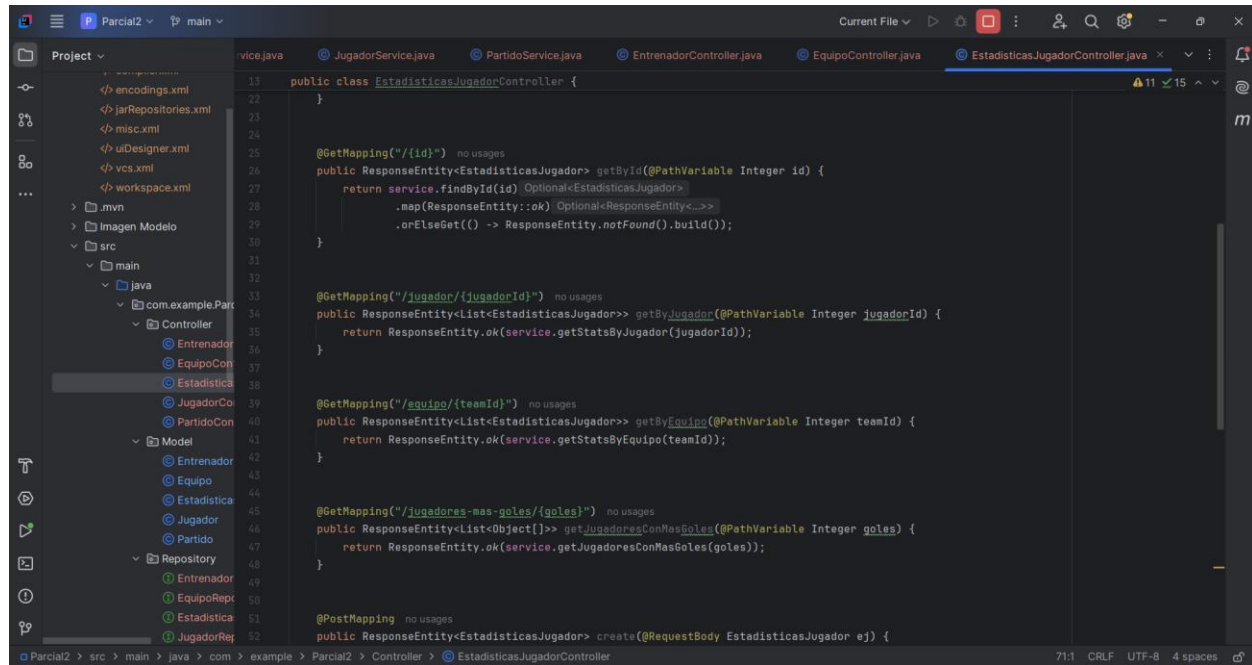


The screenshot shows an IDE with a project named 'Parcial2'. The left sidebar displays the project structure, with the 'EquipoController' class selected under the 'Controller' package. The main editor window shows the code for 'EquipoController.java'. The code is as follows:

```
1 package com.example.Parcial2.Controller;
2
3
4 import com.example.Parcial2.Model.Equipo;
5 import com.example.Parcial2.Service.EquipoService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/equipos")
14 public class EquipoController {
15
16     private final EquipoService service;
17
18     @Autowired
19     public EquipoController(EquipoService service) {
20         this.service = service;
21     }
22
23     @GetMapping
24     public ResponseEntity<List<Equipo>> getAll() {
25         return ResponseEntity.ok(service.findAll());
26     }
27
28     @GetMapping("/{id}")
29     public ResponseEntity<Equipo> getById(@PathVariable("id") Integer id) {
30         return service.findById(id).map(ResponseEntity::ok).orElse(ResponseEntity.noContent());
31     }
32 }
```

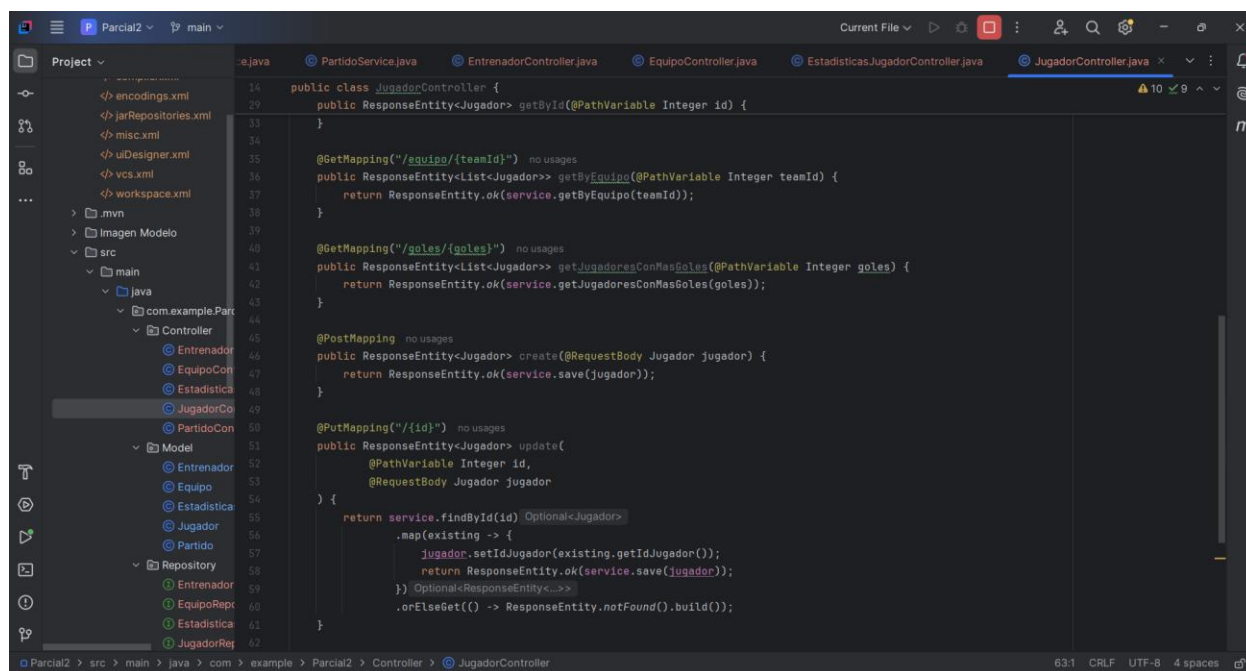
The status bar at the bottom indicates the file is 68 lines long, uses CRLF line endings, is in UTF-8 encoding, and has 4 spaces for indentation.

Estadísticas jugador controlador:



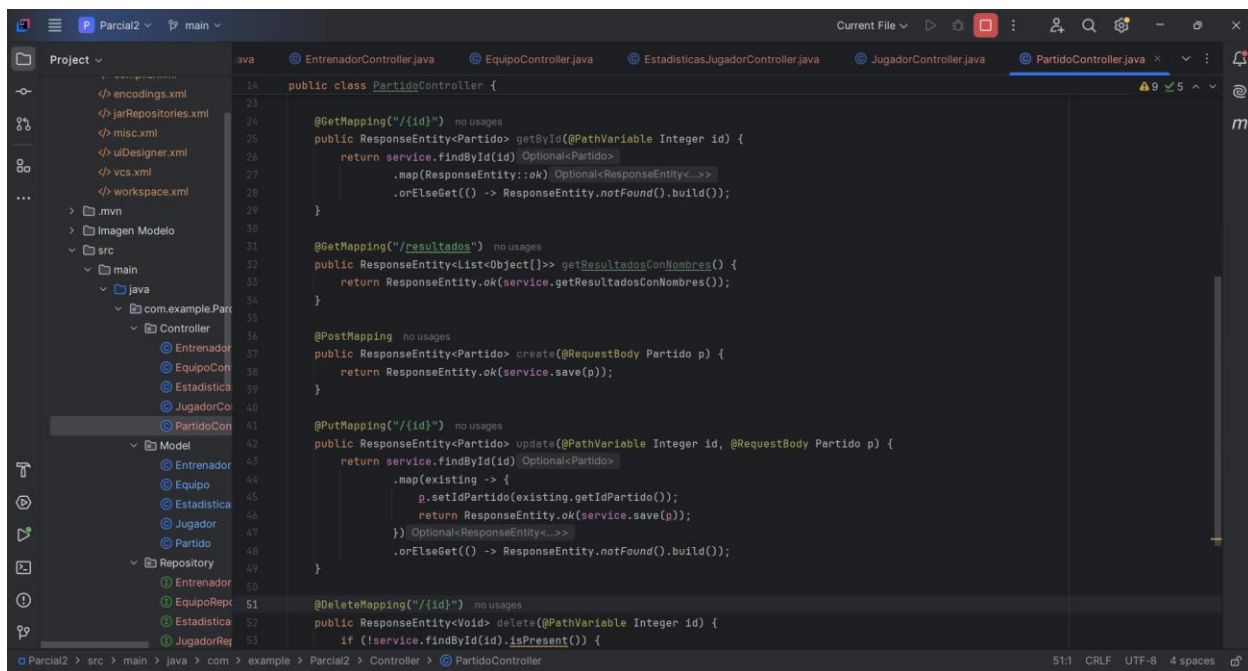
```
13 public class EstadisticasJugadorController {
14
15 }
16
17 @GetMapping("/{id}") no usages
18 public ResponseEntity<EstadisticasJugador> getById(@PathVariable Integer id) {
19     return service.findById(id).Optional<EstadisticasJugador>
20         .map(ResponseEntity::ok) Optional<ResponseEntity<_>>
21         .orElseGet(() -> ResponseEntity.notFound().build());
22 }
23
24 @GetMapping("/{jugador/{jugadorId}}") no usages
25 public ResponseEntity<List<EstadisticasJugador>> getByJugador(@PathVariable Integer jugadorId) {
26     return ResponseEntity.ok(service.getStatsByJugador(jugadorId));
27 }
28
29 @GetMapping("/{equipo/{teamId}}") no usages
30 public ResponseEntity<List<EstadisticasJugador>> getByEquipo(@PathVariable Integer teamId) {
31     return ResponseEntity.ok(service.getStatsByEquipo(teamId));
32 }
33
34 @GetMapping("/{jugadores-mas-goles/{goles}}") no usages
35 public ResponseEntity<List<Object>> getJugadoresConMasGoles(@PathVariable Integer goles) {
36     return ResponseEntity.ok(service.getJugadoresConMasGoles(goles));
37 }
38
39 @PostMapping no usages
40 public ResponseEntity<EstadisticasJugador> create(@RequestBody EstadisticasJugador ej) {
41
42 }
```

Jugador controlador:



```
14 public class JugadorController {
15     public ResponseEntity<Jugador> getById(@PathVariable Integer id) {
16     }
17
18     @GetMapping("/{equipo}/{teamId}") no usages
19     public ResponseEntity<List<Jugador>> getByEquipo(@PathVariable Integer teamId) {
20         return ResponseEntity.ok(service.getByEquipo(teamId));
21     }
22
23     @GetMapping("/{goles}/{goles}") no usages
24     public ResponseEntity<List<Jugador>> getJugadoresConMasGoles(@PathVariable Integer goles) {
25         return ResponseEntity.ok(service.getJugadoresConMasGoles(goles));
26     }
27
28     @PostMapping no usages
29     public ResponseEntity<Jugador> create(@RequestBody Jugador jugador) {
30         return ResponseEntity.ok(service.save(jugador));
31     }
32
33     @PutMapping("/{id}") no usages
34     public ResponseEntity<Jugador> update(
35         @PathVariable Integer id,
36         @RequestBody Jugador jugador
37     ) {
38         return service.findById(id).Optional<Jugador>
39             .map(existing -> {
40                 jugador.setIdJugador(existing.getIdJugador());
41                 return ResponseEntity.ok(service.save(jugador));
42             }) Optional<ResponseEntity<...>>
43             .orElseGet(() -> ResponseEntity.notFound().build());
44     }
45 }
```

Partido Controlador:



```
14 public class PartidoController {
15
16     @GetMapping("/{id}") no usages
17     public ResponseEntity<Partido> getById(@PathVariable Integer id) {
18         return service.findById(id) Optional<Partido>
19             .map(ResponseEntity::ok) Optional<ResponseEntity<...>>
20             .orElseGet(() -> ResponseEntity.notFound().build());
21     }
22
23     @GetMapping("/resultados") no usages
24     public ResponseEntity<List<Object[]>> getResultadosConNombres() {
25         return ResponseEntity.ok(service.getResultadosConNombres());
26     }
27
28     @PostMapping no usages
29     public ResponseEntity<Partido> create(@RequestBody Partido p) {
30         return ResponseEntity.ok(service.save(p));
31     }
32
33     @PutMapping("/{id}") no usages
34     public ResponseEntity<Partido> update(@PathVariable Integer id, @RequestBody Partido p) {
35         return service.findById(id) Optional<Partido>
36             .map(existing -> {
37                 p.setIdPartido(existing.getIdPartido());
38                 return ResponseEntity.ok(service.save(p));
39             }) Optional<ResponseEntity<...>>
40             .orElseGet(() -> ResponseEntity.notFound().build());
41     }
42
43     @DeleteMapping("/{id}") no usages
44     public ResponseEntity<Void> delete(@PathVariable Integer id) {
45         if (!service.findById(id).isPresent()) {
46
47         }
48     }
49 }
50
51
52
53
```

Consultas nativas

Obtener todos los jugadores de un equipo específico

```
import com.example.Parcial2.Model.Jugador;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository 2 usages
public interface JugadorRepository extends JpaRepository<Jugador, Integer> {

    @Query(value = "SELECT * FROM jugador WHERE id_equipo = :teamId", nativeQuery = true) 1 usage
    List<Jugador> findByEquipo(@Param("teamId") Integer teamId);
}
```

```
1 package com.example.Parcial2.Controller;
2
3
4 import com.example.Parcial2.Model.Jugador;
5 import com.example.Parcial2.Service.JugadorService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController no usages
13 @RequestMapping("/jugadores")
14 public class JugadorController {
15
```

The screenshot displays the Postman API client interface. On the left, the 'My Workspace' sidebar shows a collection named 'Parcial2 API' with a sub-collection 'Jugadores'. The 'Jugadores' sub-collection is expanded, showing several endpoints, with 'GET Jugadores por Equipo' selected. The main panel shows a GET request to 'http://localhost:8080/jugadores/equipo/68'. The request is successful, returning a 200 OK status. The response body is displayed in JSON format, showing a list of players for the team with ID 68. The JSON structure includes player details like name, position, and date of birth, along with team statistics.

Request: GET `http://localhost:8080/jugadores/equipo/68`

Response: 200 OK (1.02 s, 138.2 KB)

Body (JSON):

```
1 [
2   {
3     "idJugador": 76,
4     "nombre": "Jugador_76",
5     "posicion": "Portero",
6     "dorsal": 17,
7     "fechaNac": "1996-03-30",
8     "nacionalidad": "Pais_76",
9     "estadisticas": [
10      {
11        "idEstadistica": 85,
12        "partido": {
13          "idPartido": 18,
14          "fecha": "2025-05-10",
15          "estadio": "Estadio_18",
16          "..."
17        }
18      }
19    ]
20  }
21 ]
```

Obtener los jugadores que han marcado más de X goles

```

List<EstadisticasJugador> findStatsByEquipo(@Param("teamId") Integer teamId);

@Query(value = "SELECT j.id_jugador, j.nombre, SUM(ej.goles) AS total_goles " +
    "FROM jugador j " +
    "JOIN estadisticas_jugador ej ON j.id_jugador = ej.id_jugador " +
    "GROUP BY j.id_jugador, j.nombre " +
    "HAVING SUM(ej.goles) > :goles",
    nativeQuery = true)
List<Object[]> findJugadoresConMasGoles(@Param("goles") Integer goles);
}

```

```

10
11 @RestController no usages
12 @RequestMapping("/estadisticas")
13 public class EstadisticasJugadorController {
14
15     @Autowired 10 usages
16     private EstadisticasJugadorService service;
17
18
19     @GetMapping no usages
20     public ResponseEntity<List<EstadisticasJugador>> getAll() {
21         return ResponseEntity.ok(service.findAll());
22     }
23
24
25     @GetMapping("/{id}") no usages
26     public ResponseEntity<EstadisticasJugador> getById(@PathVariable Integer id) {
27         return service.findById(id).Optional<EstadisticasJugador>
28             .map(ResponseEntity::ok).Optional<ResponseEntity<...>>
29             .orElseGet(() -> ResponseEntity.notFound().build());
30     }
31

```

The screenshot displays the Postman API client interface. The left sidebar shows the 'My Workspace' with a collection named 'Parcial2 API' containing several endpoints. The main panel shows a REST client request for the endpoint 'Jugadores con + Goles (resumen)' with a GET method and the URL 'http://localhost:8080/estadisticas/jugadores-mas-goles/5'. The response is a 200 OK status with a 494 ms response time and 223 B of data. The response body is a JSON array of two objects, each representing a player's statistics.

Request Details:

- Method: GET
- URL: http://localhost:8080/estadisticas/jugadores-mas-goles/5
- Params: None
- Headers: None
- Body: None

Response Details:

- Status: 200 OK
- Time: 494 ms
- Size: 223 B
- Body: JSON

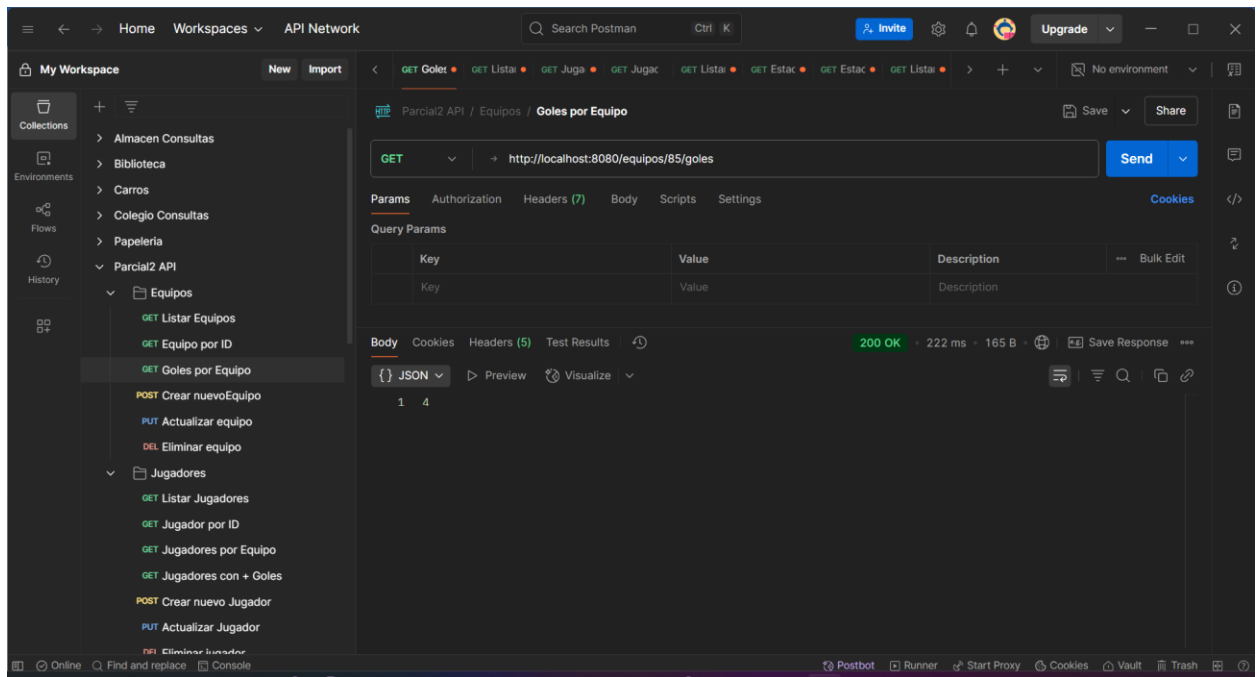
Response Body (JSON):

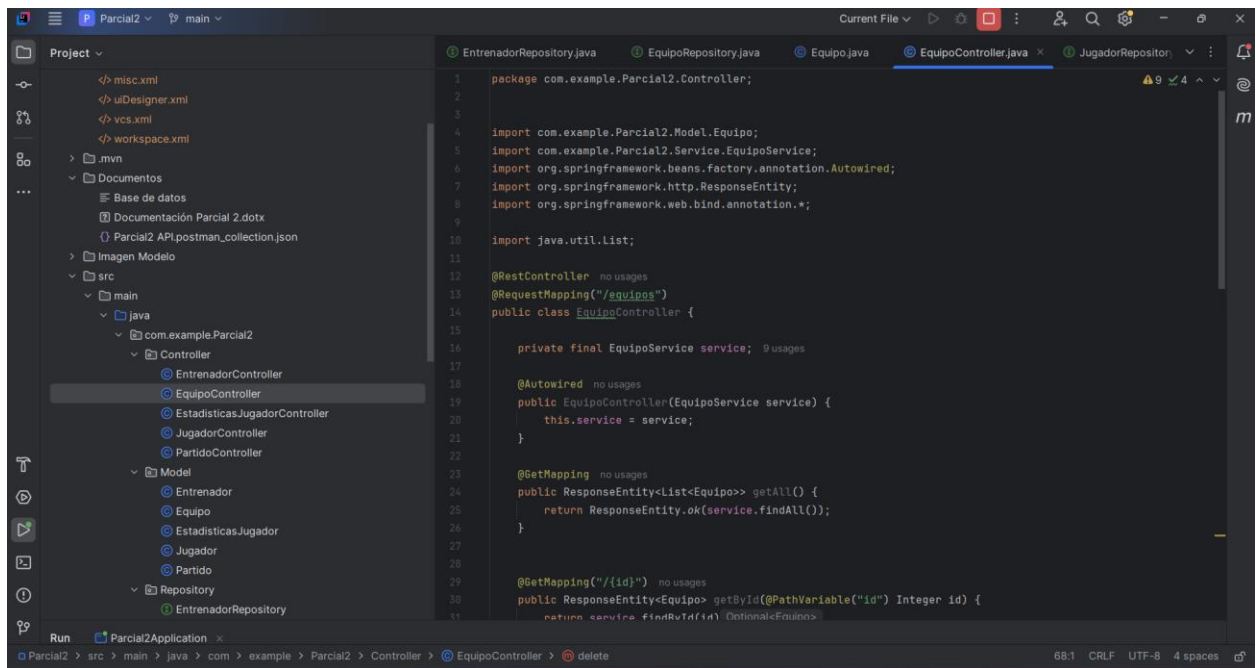
```
[{"id": 6, "jugador": "Jugador_6", "goles": 6}, {"id": 19, "jugador": "Jugador_19", "goles": 6}]
```

Obtener el número total de goles marcados por un equipo en todos sus partidos

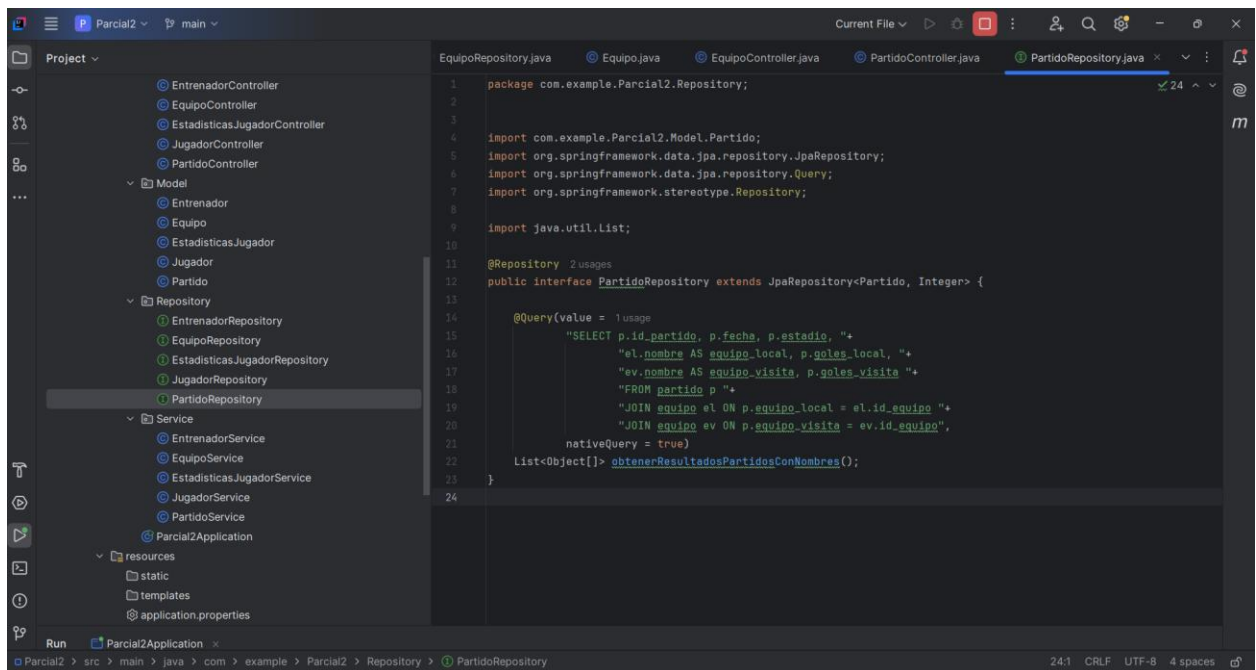
```
@Repository 2 usages
public interface EquipoRepository extends JpaRepository<Equipo, Integer> {

    @Query(value = "SELECT COALESCE(\n" + 1 usage
        " SUM(CASE WHEN equipo_local = :teamId THEN goles_local ELSE 0 END) +\n" +
        " SUM(CASE WHEN equipo_visita = :teamId THEN goles_visita ELSE 0 END), 0)\n" +
        "FROM partido\n" +
        "WHERE equipo_local = :teamId OR equipo_visita = :teamId",
        nativeQuery = true)
    Integer totalGolesPorEquipo(@Param("teamId") Integer teamId);
```





Obtener los resultados de todos los partidos indicando los nombres de los equipos



The screenshot displays the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows a 'My Workspace' collection with various API endpoints categorized under 'Estadísticas', 'Partidos', and 'Entrenador'. The main panel shows a selected endpoint: 'GET http://localhost:8080/partidos/resultados'. The request is a GET method with no parameters. The response is a 200 OK status, indicating a successful request. The response body is displayed in JSON format, showing a list of two match results.

Request:

```
GET http://localhost:8080/partidos/resultados
```

Response: 200 OK - 333 ms - 5.83 KB

JSON Body:

```
[
  {
    "id": 1,
    "date": "2025-04-23",
    "stadium": "Estadio_1",
    "team": "Equipo_12",
    "goals": 0,
    "team_name": "Equipo_6",
    "goals_scored": 3
  },
  {
    "id": 2,
    "date": "2025-04-24",
    "stadium": "Estadio_2",
    "team": "Equipo_23",
    "goals": 0
  }
]
```