

## QUIZ2LINEA3

---

RAUL ALEXANDER SUESCUN PEREZ

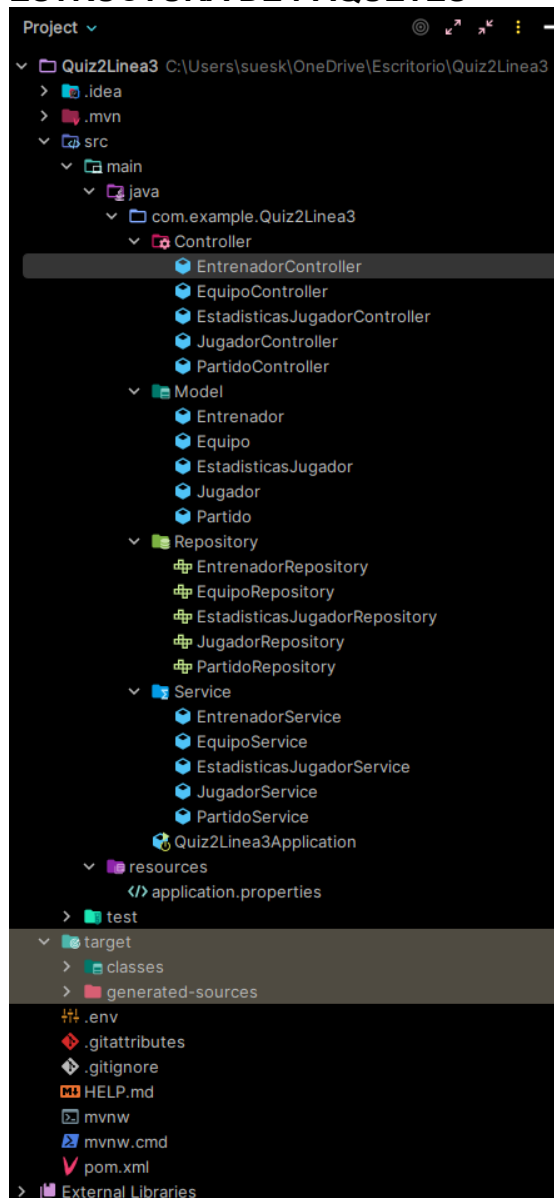
LÍNEA DE PROFUNDIZACIÓN 3

FECHA: 14 DE OCTUBRE DE 2025

DESARROLLADO CON SPRING BOOT Y MYSQL

El proyecto Quiz2Linea3 fue desarrollado utilizando Spring Boot con Java 21 y MySQL como base de datos relacional. El objetivo es implementar un sistema de gestión para equipos de fútbol, incluyendo entidades relacionadas como Equipo, Jugador, Entrenador, Partido y EstadísticasJugador, cumpliendo con las relaciones definidas en el modelo relacional.

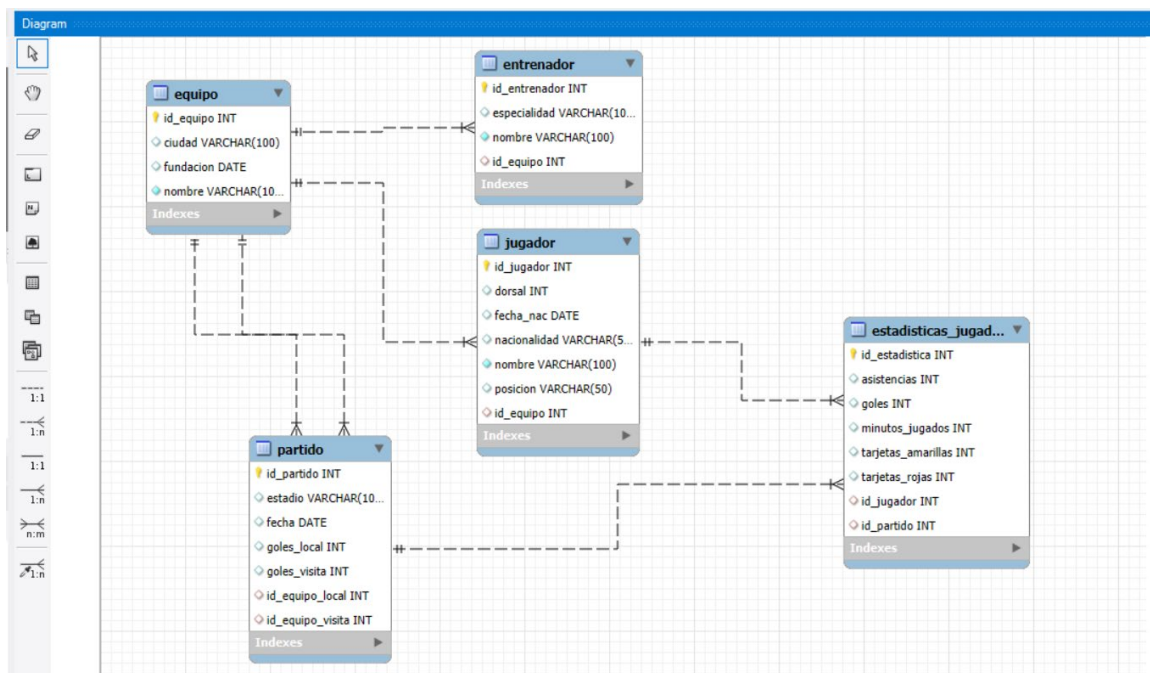
### ESTRUCTURA DE PAQUETES



## MODELO RELACIONAL

Las principales tablas del sistema son las siguientes:

- Equipo: id\_equipo (PK), nombre, ciudad, fundacion
- Jugador: id\_jugador (PK), nombre, posicion, dorsal, fecha\_nac, nacionalidad, id\_equipo (FK)
- Entrenador: id\_entrenador (PK), nombre, especialidad, id\_equipo (FK)
- Partido: id\_partido (PK), fecha, estadio, goles\_local, goles\_visita, id\_equipo\_local (FK), id\_equipo\_visita (FK)
- EstadisticasJugador: id\_estadistica (PK), minutos\_jugados, goles, asistencias, tarjetas\_amarillas, tarjetas\_rojas, id\_jugador (FK), id\_partido (FK)
- Relaciones principales:
  - Equipo (1) ---< Jugador (N)
  - Equipo (1) ---< Entrenador (N)
  - Jugador (1) ---< EstadisticasJugador (N)
  - Partido (1) ---< EstadisticasJugador (N)



## CONFIGURACIÓN DEL PROYECTO

application.properties:

spring.application.name=Quiz2Linea3

spring.datasource.url=\${DB\_URL}

spring.datasource.username=\${DB\_USERNAME}

spring.datasource.password=\${DB\_PASSWORD}

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

.env:

DB\_URL=jdbc:mysql://localhost:3315/quiz2linea3

DB\_USERNAME=root

DB\_PASSWORD=1234

## COLECCIÓN DE CONSULTAS, METODOS, PARÁMETROS Y RESULTADOS

### 1. EQUIPO

- GET /api/equipo/listar - Lista todos los registros

```
// GET - LISTAR EQUIPOS
public ResponseEntity<?> listar() {
    List<com.example.Quiz2Linea3.Model.Equipo> equipos = equipoRepository.findAll();
    if (equipos.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NO_CONTENT).body("NO EXISTEN EQUIPOS REGISTRADOS");
    }
    return ResponseEntity.ok(equipos);
}
```

The screenshot shows a REST client interface with a GET request to `{{baseUri}}/equipo/listar`. The response is a 200 OK status with a response time of 765 ms and a body size of 4.47 KB. The response body is a JSON array containing one team object.

**Query Params**

Key	Value	Description
Key	Value	Description

**Body**

```
{
  "idEquipo": 1,
  "nombre": "Real Bogotá",
  "ciudad": "Bogotá",
  "fundacion": "1990-03-15",
  "jugadores": [
    {
      "idJugador": 1,
      "nombre": "David Torres",
      "posicion": "Delantero",
      "dorsal": 9,
      "fechaNac": "1997-05-12",
      "nacionalidad": "Colombiana",
      "estadisticas": [
        {
          "idEstadistica": 1,
          "minutosJugados": 90,
          "goles": 2,
          "asistencias": 1,
          "tarjetasAmarillas": 0,

```

- GET /api/equipo/buscar/{id} - Busca un registro por ID

```
// GET - BUSCAR EQUIPO POR ID
public ResponseEntity<?> buscarPorId(Integer id) { 1 usage
    try {
        Optional<com.example.Quiz2Linea3.Model.Equipo> equipo = equipoRepository.findById(id);

        if (equipo.isPresent()) {
            return ResponseEntity.status(HttpStatus.OK).body(equipo.get());
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO SE ENCONTRÓ EQUIPO CON ID: " + id);
        }
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("ERROR AL BUSCAR EQUIPO CON ID: " + id + ". DETALLE: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUrl}} /equipo/buscar/1
- Status:** 200 OK
- Time:** 92 ms
- Size:** 1.75 KB
- Body:** JSON

The JSON response body is as follows:

```
{
  "idEquipo": 1,
  "nombre": "Real Bogotá",
  "ciudad": "Bogotá",
  "fundacion": "1990-03-15",
  "jugadores": [
    {
      "idJugador": 1,
      "nombre": "David Torres",
      "posicion": "Delantero",
      "dorsal": 9,
      "fechaNac": "1997-05-12",
      "nacionalidad": "Colombiana",
      "estadisticas": [
        {
          "idEstadistica": 1,
          "minutosJugados": 90,
          "goles": 2,
          "asistencias": 1,
          "tarjetasAmarillas": 0,
          "tarjetasRojas": 0
        }
      ]
    }
  ]
}
```

- POST /api/equipo/guardar - Crea un nuevo registro

```
// POST - GUARDAR EQUIPO
public ResponseEntity<?> guardar(com.example.Quiz2Linea3.Model.Equipo equipo) { 1 usage
    try {
        if (equipo.getNombre() == null || equipo.getNombre().isBlank()) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("EL NOMBRE DEL EQUIPO ES OBLIGATORIO");
        }

        com.example.Quiz2Linea3.Model.Equipo nuevo = equipoRepository.save(equipo);
        return ResponseEntity.status(HttpStatus.CREATED).body("EQUIPO CREADO CON ID: " + nuevo.getIdEquipo());
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body("ERROR AL GUARDAR EQUIPO: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** {{baseUrl}} /equipo/guardar
- Body:** A JSON object: 

```
{ 1: { 2: "nombre": "Millonarios FC", 3: "ciudad": "Bogotá", 4: "fundacion": "1946-06-18" 5: } 6: }
```
- Response:** 201 Created, 393 ms, 273 B. The response body is: 

```
1 EQUIPO CREADO CON ID: 6
```

Verificado al listar:

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUrl}} /equipo/listar
- Response:** 200 OK, 250 ms, 1.9 KB. The response body is a JSON array: 

```
120 entrenadores : L 121: { 122: "idEntrenador": 5, 123: "nombre": "Luis López", 124: "especialidad": "Motivacional" 125: } 126: ], 127: { 128: "idEquipo": 6, 129: "nombre": "Millonarios FC", 130: "ciudad": "Bogotá", 131: "fundacion": "1946-06-18", 132: "jugadores": [], 133: "entrenadores": [] 134: } 135: ]
```

- PUT /api/equipo/actualizar/{id} - Actualiza un registro existente

```
// PUT - ACTUALIZAR EQUIPO
public ResponseEntity<?> actualizar(Integer id, com.example.Quiz2Linea3.Model.Equipo equipoActualizado) { 1 usage
    Optional<com.example.Quiz2Linea3.Model.Equipo> existente = equipoRepository.findById(id);
    if (existente.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("NO SE ENCONTRÓ EQUIPO CON ID: " + id);
    }

    com.example.Quiz2Linea3.Model.Equipo equipo = existente.get();
    equipo.setNombre(equipoActualizado.getNombre());
    equipo.setCiudad(equipoActualizado.getCiudad());
    equipo.setFundacion(equipoActualizado.getFundacion());

    equipoRepository.save(equipo);
    return ResponseEntity.ok(body: "EQUIPO CON ID: " + id + " ACTUALIZADO CORRECTAMENTE");
}
```

PUT {{baseUri}} /equipo/actualizar/1

Params Authorization Headers (10) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "nombre": "Tiburones Dorados",
3   "ciudad": "Santa Marta",
4   "fundacion": "2005-04-18"
5 }
```

Body Cookies Headers (8) Test Results

200 OK • 199 ms • 287 B • Save Response

{ JSON Preview Visualize

1 EQUIPO CON ID: 1 ACTUALIZADO CORRECTAMENTE

## Verificado al listar

GET {{baseUri}} /equipo/listar

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (8) Test Results

200 OK • 89 ms • 4.59 KB • Save Response

{ JSON Preview Visualize

```
1 [
2   {
3     "idEquipo": 1,
4     "nombre": "Tiburones Dorados",
5     "ciudad": "Santa Marta",
6     "fundacion": "2005-04-18",
```

- DELETE /api/equipo/eliminar/{id} - Elimina un registro

```
// DELETE - ELIMINAR EQUIPO
public ResponseEntity<?> eliminarPorId(Integer id) { 1 usage
    if (!equipoRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("NO EXISTE EQUIPO CON ID: " + id);
    }
    equipoRepository.deleteById(id);
    return ResponseEntity.ok(body: "EQUIPO CON ID: " + id + " ELIMINADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface for a project named "Quiz2Linea3 CRUD + Nativa". The selected endpoint is "Eliminar equipo" with the URL template "{(baseUrl)} /equipo/eliminar/6". The method is set to "DELETE". The "Send" button is visible. Below the URL bar, there are tabs for "Params", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". The "Query Params" section is empty. The "Body" tab is active, showing a "Raw" view with the response text: "1 EQUIPO CON ID: 6 ELIMINADO CORRECTAMENTE". The status bar at the bottom indicates a "200 OK" response with a time of 612 ms and a size of 293 B.

## Verificado al listar

The screenshot shows a REST client interface for the same project. The selected endpoint is "/equipo/listar" with the method set to "GET". The "Send" button is visible. Below the URL bar, there are tabs for "Params", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". The "Query Params" section is empty. The "Body" tab is active, showing a "JSON" view of the response. The JSON data is as follows:

```
{
  "idEquipo": 5,
  "nombre": "Llaneros FC",
  "ciudad": "Villavicencio",
  "fundacion": "1995-09-18",
  "jugadores": [],
  "entrenadores": [
    {
      "idEntrenador": 5,
      "nombre": "Luis López",
      "especialidad": "Motivacional"
    }
  ]
}
```

The status bar at the bottom indicates a "200 OK" response with a time of 78 ms and a size of 4.48 KB.

- CONSULTA NATIVA /api/equipo/contar – Cuenta el número de equipos

```
// CONSULTA NATIVA - CONTAR EQUIPOS
public ResponseEntity<?> contarEquipos() { 1 usage
    Integer total = equipoRepository.contarEquipos();
    return ResponseEntity.ok(body: "TOTAL DE EQUIPOS REGISTRADOS: " + total);
}
```

```
package com.example.Quiz2Linea3.Repository;

import org.springframework.data.jpa.repository.*;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository 2 usages
public interface EquipoRepository extends JpaRepository<com.example.Quiz2Linea3.Model.Equipo, Integer> {

    // CONSULTA NATIVA 1: Buscar equipos por ciudad
    @Query(value = "SELECT * FROM Equipo WHERE ciudad = :ciudad", nativeQuery = true) 1 usage
    List<com.example.Quiz2Linea3.Model.Equipo> buscarPorCiudad(@Param("ciudad") String ciudad);

    // CONSULTA NATIVA 2: Contar equipos registrados
    @Query(value = "SELECT COUNT(*) FROM Equipo", nativeQuery = true) 1 usage
    Integer contarEquipos();
}
```

The screenshot shows a REST client interface with a GET request to `{{baseUri}} /equipo/contar`. The request is sent, and the response is displayed in the 'Body' tab. The response status is `200 OK` with a response time of `28 ms` and a body size of `284 B`. The response body is `TOTAL DE EQUIPOS REGISTRADOS: 5`.

GET `{{baseUri}} /equipo/contar` Send

Params Authorization Headers (7) Body Scripts Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (8) Test Results 200 OK • 28 ms • 284 B Save Response

Raw Preview Visualize

1 TOTAL DE EQUIPOS REGISTRADOS: 5

Runner Start Proxy Cookies Vault Trash



## 2. JUGADOR

- GET /api/jugador/listar - Lista todos los registros

```
// GET - LISTAR JUGADORES
public ResponseEntity<?> listar() {
    List<com.example.Quiz2Linea3.Model.Jugador> jugadores = jugadorRepository.findAll();
    if (jugadores.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NO_CONTENT).body("NO EXISTEN JUGADORES REGISTRADOS");
    }
    return ResponseEntity.ok(jugadores);
}
```

The screenshot shows a REST client interface with the following details:

- Request:** Method: GET, URL: {{baseUrl}} /jugador/listar. Status: 200 OK, 66 ms, 3.56 KB.
- Params:** Tab selected. Query Params table is empty.
- Body:** Tab selected. Content type: JSON. The response is a JSON array with one object representing a player.

**JSON Response:**

```
[
  {
    "idJugador": 1,
    "nombre": "David Torres",
    "posicion": "Delantero",
    "dorsal": 9,
    "fechaNac": "1997-05-12",
    "nacionalidad": "Colombiana",
    "estadisticas": [
      {
        "idEstadistica": 1,
        "minutosJugados": 90,
        "goles": 2,
        "asistencias": 1,
        "tarjetasAmarillas": 0,
        "tarjetasRojas": 0
      },
      {
        "idEstadistica": 6,
        "minutosJugados": 90,

```

- GET /api/jugador/buscar/{id} - Busca un registro por ID

```
// GET - BUSCAR JUGADOR POR ID
public ResponseEntity<?> buscarPorId(Integer id) { 1 usage
    try {
        Optional<com.example.Quiz2Linea3.Model.Jugador> jugador = jugadorRepository.findById(id);

        if (jugador.isPresent()) {
            return ResponseEntity.status(HttpStatus.OK).body(jugador.get());
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO SE ENCONTRÓ JUGADOR CON ID: " + id);
        }
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("ERROR AL BUSCAR JUGADOR CON ID: " + id + ". DETALLE: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUri}} /jugador/buscar/5
- Status:** 200 OK
- Time:** 18 ms
- Size:** 928 B
- Body:** JSON

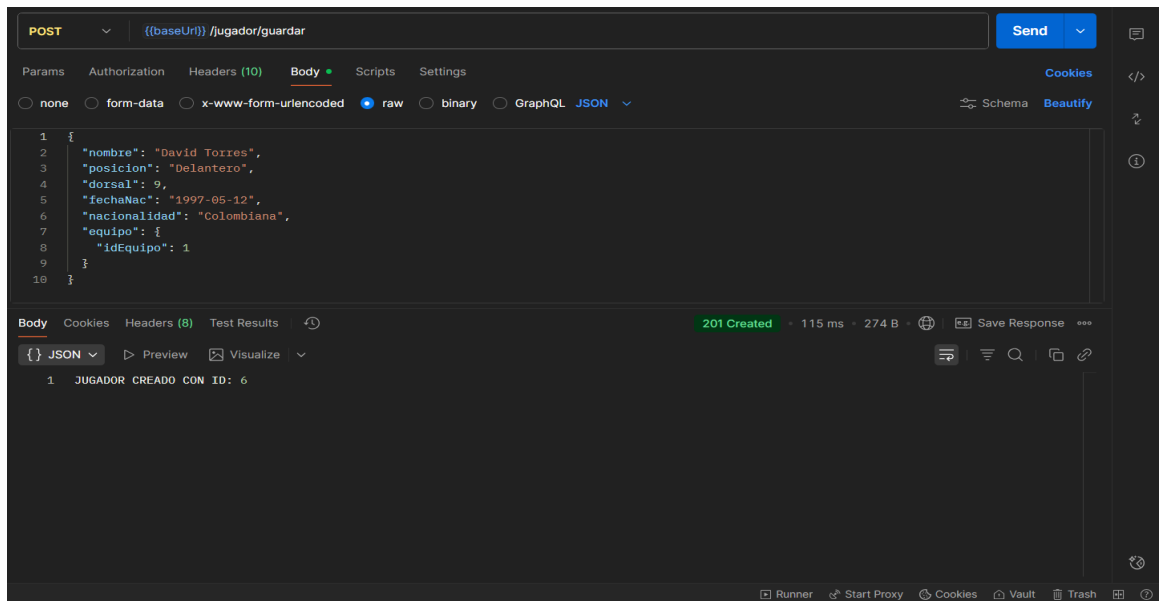
The JSON response is as follows:

```
{
  "idJugador": 5,
  "nombre": "Luis Rojas",
  "posicion": "Delantero",
  "dorsal": 10,
  "fechaNac": "1996-03-18",
  "nacionalidad": "Colombiana",
  "estadisticas": [
    {
      "idEstadistica": 5,
      "minutosJugados": 88,
      "goles": 0,
      "asistencias": 2,
      "tarjetasAmarillas": 0,
      "tarjetasRojas": 0
    },
    {
      "idEstadistica": 10,

```

- POST /api/jugador/guardar - Crea un nuevo registro

```
// POST - GUARDAR JUGADOR
public ResponseEntity<?> guardar(com.example.Quiz2Linea3.Model.Jugador jugador) { 1 usage
    try {
        if (jugador.getNombre() == null || jugador.getNombre().isBlank()) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("EL NOMBRE DEL JUGADOR ES OBLIGATORIO");
        }
        com.example.Quiz2Linea3.Model.Jugador nuevo = jugadorRepository.save(jugador);
        return ResponseEntity.status(HttpStatus.CREATED).body("JUGADOR CREADO CON ID: " + nuevo.getIdJugador());
    } catch (Exception e) {
        return ResponseEntity.internalServerError().body("ERROR AL GUARDAR JUGADOR: " + e.getMessage());
    }
}
```



- PUT /api/jugador/actualizar/{id} - Actualiza un registro existente

```
// PUT - ACTUALIZAR JUGADOR
public ResponseEntity<?> actualizar(Integer id, Jugador datos) { 1 usage
    Optional<Jugador> existente = jugadorRepository.findById(id);
    if (existente.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE JUGADOR CON ID: " + id);
    }

    Jugador jugador = existente.get();
    jugador.setNombre(datos.getNombre());
    jugador.setPosicion(datos.getPosicion());
    jugador.setDorsal(datos.getDorsal());
    jugador.setFechaNac(datos.getFechaNac());
    jugador.setNacionalidad(datos.getNacionalidad());
    jugador.setEquipo(datos.getEquipo());

    jugadorRepository.save(jugador);
    return ResponseEntity.ok().body("JUGADOR CON ID: " + id + " ACTUALIZADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** {{baseUri}} /jugador/actualizar/1
- Body (Request):** A JSON object representing a player: 

```
{  "nombre": "David Torres",  "posicion": "Extremo Derecho",  "dorsal": 7,  "fechaNac": "1997-05-12",  "nacionalidad": "Colombiana",  "equipo": {    "idEquipo": 1  }}
```
- Status:** 200 OK
- Response Body:** JUGADOR CON ID: 1 ACTUALIZADO CORRECTAMENTE

- DELETE /api/jugador/eliminar/{id} - Elimina un registro

```
// DELETE - ELIMINAR JUGADOR
public ResponseEntity<?> eliminarPorId(Integer id) { 1 usage
    if (!jugadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE JUGADOR CON ID: " + id);
    }
    jugadorRepository.deleteById(id);
    return ResponseEntity.ok(body: "JUGADOR CON ID: " + id + " ELIMINADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** {{baseUrl}} /jugador/eliminar/6
- Params:** Authorization, Headers (7), Body, Scripts, Settings
- Query Params:** A table with 4 columns: Key, Value, Description, and Bulk Edit. The first row contains 'Key', 'Value', 'Description', and 'Bulk Edit'.
- Body:** Cookies, Headers (8), Test Results, and a refresh icon.
- Status:** 200 OK, 360 ms, 294 B
- Response Body:** 1 JUGADOR CON ID: 6 ELIMINADO CORRECTAMENTE

The interface also includes a 'Send' button, a 'Raw' view selector, and a 'Save Response' option.

- CONSULTA NATIVA api/jugador/contar/equipo/{idEquipo} – Cuenta jugadores en un equipo

```
// CONSULTA NATIVA 2 - Contar jugadores por equipo
public ResponseEntity<?> contarPorEquipo(Integer idEquipo) { 1 usage
    Integer total = jugadorRepository.contarPorEquipo(idEquipo);
    return ResponseEntity.ok(body: "TOTAL DE JUGADORES EN EL EQUIPO " + idEquipo + ": " + total);
}
```

```
package com.example.Quiz2Linea3.Repository;

import com.example.Quiz2Linea3.Model.Jugador;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.util.List;

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

    // CONSULTA NATIVA 1: Buscar jugadores por posición
    @Query(value = "SELECT * FROM Jugador WHERE posicion = ?1", nativeQuery = true) 1 usage
    List<Jugador> buscarPorPosicion(String posicion);

    // CONSULTA NATIVA 2: Contar jugadores por equipo
    @Query(value = "SELECT COUNT(*) FROM Jugador WHERE id_equipo = ?1", nativeQuery = true) 1 usage
    Integer contarPorEquipo(Integer idEquipo);
}
```

GET {{baseUri}} /jugador/contar/equipo/3 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results 200 OK 26 ms 289 B Save Response

Raw Preview Visualize

```
1 TOTAL DE JUGADORES EN EL EQUIPO 3: 1
```

Runner Start Proxy Cookies Vault Trash

### 3. ENTRENADOR

- GET /api/entrenador/listar - Lista todos los registros

```
// GET - LISTAR ENTRENADORES
public ResponseEntity<?> listar() {
    List<Entrenador> lista = entrenadorRepository.findAll();
    if (lista.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NO_CONTENT)
            .body("NO EXISTEN ENTRENADORES REGISTRADOS");
    }
    return ResponseEntity.ok(lista);
}
```

The screenshot shows a REST client interface with a GET request to `{[baseUrl]} /entrenador/listar`. The response is a 200 OK status with a response time of 19 ms and a body size of 601 B. The response body is displayed in JSON format, showing a list of four trainers.

Key	Value	Description
Key	Value	Description

```
1 [
2   {
3     "idEntrenador": 1,
4     "nombre": "Carlos Pérez",
5     "especialidad": "Táctico"
6   },
7   {
8     "idEntrenador": 2,
9     "nombre": "Juan Gómez",
10    "especialidad": "Físico"
11  },
12  {
13    "idEntrenador": 3,
14    "nombre": "Mario Ruiz",
15    "especialidad": "Ofensivo"
16  },
17  {
18    "idEntrenador": 4,
19    "nombre": "Andrés Díaz",
20    "especialidad": "Defensivo"
21  }
22 ]
```

- GET /api/entrenador/buscar/{id} - Busca un registro por ID

```
// GET - BUSCAR POR ID
public ResponseEntity<?> buscarPorId(Integer id) { 1 usage
    Optional<Entrenador> entrenador = entrenadorRepository.findById(id);
    if (entrenador.isPresent()) {
        return ResponseEntity.ok(entrenador.get());
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO SE ENCONTRÓ ENTRENADOR CON ID: " + id);
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUri}}/entrenador/buscar/1
- Params:** Authorization, Headers (7), Body, Scripts, Settings
- Query Params:** A table with columns Key, Value, and Description. The first row shows 'Key' and 'Value'.
- Body:** JSON view showing the response body: 

```
{
  "idEntrenador": 1,
  "nombre": "Carlos Pérez",
  "especialidad": "Táctico"
}
```
- Status:** 200 OK
- Response Time:** 23 ms
- Response Size:** 322 B
- Buttons:** Send, Bulk Edit, Save Response, Preview, Visualize, Runner, Start Proxy, Cookies, Vault, Trash.



- POST /api/entrenador/guardar - Crea un nuevo registro

```
// POST - GUARDAR ENTRENADOR
public ResponseEntity<?> guardar(Entrenador entrenador) { 1 usage
    try {
        if (entrenador.getNombre() == null || entrenador.getNombre().isBlank()) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("EL NOMBRE DEL ENTRENADOR ES OBLIGATORIO");
        }
        Entrenador nuevo = entrenadorRepository.save(entrenador);
        return ResponseEntity.status(HttpStatus.CREATED)
            .body("ENTRENADOR REGISTRADO CON ID: " + nuevo.getIdEntrenador());
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("ERROR AL GUARDAR ENTRENADOR: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** {{baseUrl}} /entrenador/guardar
- Body:** A JSON object representing a coach: 

```
{ 1: { 2: "nombre": "Franco Escamilla", 3: "especialidad": "Táctico", 4: "equipo": { 5: "idEquipo": 1 6: } 7: }
```
- Response:** 201 Created, 28 ms, 281 B. The response body is: 

```
1 ENTRENADOR REGISTRADO CON ID: 6
```
- Interface Elements:** Tabs for Params, Authorization, Headers (10), Body, Scripts, Settings, Cookies. The Body tab is active, showing the request and response JSON. The response is also displayed in a 'Body' section at the bottom.

- PUT /api/entrenador/actualizar/{id} - Actualiza un registro existente

```
// PUT - ACTUALIZAR ENTRENADOR
public ResponseEntity<?> actualizar(Integer id, Entrenador entrenadorActualizado) { 1 usage
    Optional<Entrenador> existente = entrenadorRepository.findById(id);
    if (existente.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE ENTRENADOR CON ID: " + id);
    }

    Entrenador entrenador = existente.get();
    entrenador.setNombre(entrenadorActualizado.getNombre());
    entrenador.setEspecialidad(entrenadorActualizado.getEspecialidad());
    entrenador.setEquipo(entrenadorActualizado.getEquipo());

    entrenadorRepository.save(entrenador);
    return ResponseEntity.ok(body: "ENTRENADOR CON ID " + id + " ACTUALIZADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** {{baseUrl}}/entrenador/actualizar/1
- Body:** A JSON object representing a coach: 

```
{  "nombre": "Carlos Suarez",  "especialidad": "Defensivo",  "equipo": {    "idEquipo": 1  }}
```
- Response:** 200 OK, 38 ms, 290 B. The response body is: 

```
ENTRENADOR CON ID 1 ACTUALIZADO CORRECTAMENTE
```

The interface also includes tabs for Params, Authorization, Headers (10), Body, Scripts, and Settings. The Body tab is active, showing the JSON body and the response body. The response status is 200 OK, indicating a successful update.

- DELETE /api/entrenador/eliminar/{id} - Elimina un registro

```
// DELETE - ELIMINAR ENTRENADOR
public ResponseEntity<?> eliminarPorId(Integer id) { 1 usage
    if (!entrenadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE ENTRENADOR CON ID: " + id);
    }
    entrenadorRepository.deleteById(id);
    return ResponseEntity.ok(body: "ENTRENADOR ELIMINADO CON ÉXITO");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** {{baseUrl}}/entrenador/eliminar/6
- Send Button:** A blue button labeled "Send".
- Params Tab:** Active, showing a table for Query Params.
- Query Params Table:**

Key	Value	Description
Key	Value	Description
- Body Tab:** Active, showing the response body.
- Response Status:** 200 OK
- Response Time:** 44 ms
- Response Size:** 284 B
- Response Body:** ENTRENADOR ELIMINADO CON ÉXITO
- Raw Tab:** Active, showing the raw response text.

- CONSULTA NATIVA api/entrenador/contar – Cuenta entrenadores

```
// CONSULTA NATIVA 2 - Contar todos los entrenadores
@GetMapping("/contar")
public ResponseEntity<?> contarEntrenadores() {
    return entrenadorService.contarEntrenadores();
}
```

```
package com.example.Quiz2Linea3.Repository;

import com.example.Quiz2Linea3.Model.Entrenador;
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
public interface EntrenadorRepository extends JpaRepository<Entrenador, Integer> {

    @Query(value = "SELECT * FROM Entrenador WHERE especialidad = :especialidad", nativeQuery = true)
    List<Entrenador> buscarPorEspecialidad(@Param("especialidad") String especialidad);

    @Query(value = "SELECT COUNT(*) FROM Entrenador", nativeQuery = true)
    Integer contarEntrenadores();

    @Query(value = "SELECT COUNT(*) FROM Entrenador WHERE id_equipo = :idEquipo", nativeQuery = true)
    Integer contarPorEquipo(@Param("idEquipo") Integer idEquipo);
}
```

GET {{baseUri}}/entrenador/contar Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (8) Test Results 200 OK 12 ms 289 B Save Response

Raw Preview Visualize

1 TOTAL DE ENTRENADORES REGISTRADOS: 5

Runner Start Proxy Cookies Vault Trash

#### 4. PARTIDO

- GET /api/partido/listar - Lista todos los registros

```
// GET - LISTAR TODOS LOS PARTIDOS
public ResponseEntity<?> listar() {
    List<Partido> lista = partidoRepository.findAll();
    if (lista.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NO_CONTENT)
            .body("NO EXISTEN PARTIDOS REGISTRADOS");
    }
    return ResponseEntity.ok(lista);
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUrl}} /partido/listar
- Status:** 200 OK
- Time:** 46 ms
- Size:** 3.4 KB
- Body:** JSON

The JSON response body is as follows:

```
[
  {
    "idPartido": 1,
    "fecha": "2025-03-01",
    "estadio": "Estadio Nacional",
    "golesLocal": 2,
    "golesVisita": 1,
    "estadisticas": [
      {
        "idEstadistica": 1,
        "minutosJugados": 90,
        "goles": 2,
        "asistencias": 1,
        "tarjetasAmarillas": 0,
        "tarjetasRojas": 0
      },
      {
        "idEstadistica": 2,
        "minutosJugados": 90,
        "goles": 0
      }
    ]
  }
]
```

- GET /api/partido/buscar/{id} - Busca un registro por ID

```
// GET - BUSCAR POR ID
public ResponseEntity<?> buscarPorId(Integer id) { 1 usage
    Optional<Partido> partido = partidoRepository.findById(id);
    if (partido.isPresent()) {
        return ResponseEntity.ok(partido.get());
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO SE ENCONTRÓ PARTIDO CON ID: " + id);
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUrl}}/partido/buscar/5
- Send Button:** A blue button labeled "Send".
- Params Tab:** Shows a table for Query Params with columns Key, Value, and Description.
- Body Tab:** Shows the response body in JSON format. The status is 200 OK, with a response time of 18 ms and a size of 360 B. The JSON data is: 

```
{
  "idPartido": 5,
  "fecha": "2025-03-20",
  "estadio": "La Sabana",
  "golesLocal": 2,
  "golesVisita": 2,
  "estadisticas": []
}
```
- Bottom Bar:** Includes buttons for Runner, Start Proxy, Cookies, Vault, and Trash.

- POST /api/partido/guardar - Crea un nuevo registro

```
// POST - GUARDAR PARTIDO
public ResponseEntity<?> guardar(Partido partido) { 1 usage
    try {
        if (partido.getFecha() == null) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("LA FECHA DEL PARTIDO ES OBLIGATORIA");
        }

        Partido nuevo = partidoRepository.save(partido);
        return ResponseEntity.status(HttpStatus.CREATED)
            .body("PARTIDO CREADO CON ID: " + nuevo.getIdPartido());
    } catch (Exception e) {
        return ResponseEntity.internalServerError()
            .body("ERROR AL GUARDAR PARTIDO: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** {{baseUrl}} /partido/guardar
- Body:** A JSON object representing a match:

```
{
  "fecha": "2025-03-01",
  "estadio": "Metropolitano",
  "golesLocal": 2,
  "golesVisita": 1,
  "equipoLocal": {
    "idEquipo": 1
  },
  "equipoVisita": {
    "idEquipo": 2
  }
}
```
- Response:** 201 Created, 26 ms, 274 B. The response body is: 

```
1 PARTIDO CREADO CON ID: 6
```
- Interface Elements:** Tabs for Params, Authorization, Headers (10), Body, Scripts, Settings. A 'Send' button is present. The 'Body' tab is active, showing the JSON payload and the response body.

- PUT /api/partido/actualizar/{id} - Actualiza un registro existente

```
// PUT - ACTUALIZAR PARTIDO
public ResponseEntity<?> actualizar(Integer id, Partido datos) { 1 usage
    Optional<Partido> existente = partidoRepository.findById(id);
    if (existente.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE PARTIDO CON ID: " + id);
    }

    Partido partido = existente.get();
    partido.setFecha(datos.getFecha());
    partido.setEstadio(datos.getEstadio());
    partido.setGolesLocal(datos.getGolesLocal());
    partido.setGolesVisita(datos.getGolesVisita());
    partido.setEquipoLocal(datos.getEquipoLocal());
    partido.setEquipoVisita(datos.getEquipoVisita());

    partidoRepository.save(partido);
    return ResponseEntity.ok().body("PARTIDO CON ID: " + id + " ACTUALIZADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** {{baseUrl}}/partido/actualizar/1
- Body:** A JSON object representing a match update:

```
{
  "fecha": "2025-03-02",
  "estadio": "Arena Nacional",
  "golesLocal": 3,
  "golesVisita": 2,
  "equipoLocal": {
    "idEquipo": 1
  },
  "equipoVisita": {
    "idEquipo": 2
  }
}
```
- Response:** 200 OK, 34 ms, 288 B. The response body is: "PARTIDO CON ID: 1 ACTUALIZADO CORRECTAMENTE".



- DELETE /api/partido/eliminar/{id} - Elimina un registro

```
// DELETE - ELIMINAR PARTIDO
public ResponseEntity<?> eliminarPorId(Integer id) { 1 usage
    if (!partidoRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE PARTIDO CON ID: " + id);
    }
    partidoRepository.deleteById(id);
    return ResponseEntity.ok(body: "PARTIDO CON ID: " + id + " ELIMINADO CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** {{baseUrl}} /partido/eliminar/1
- Status:** 200 OK
- Time:** 69 ms
- Size:** 294 B
- Body:** PARTIDO CON ID: 1 ELIMINADO CORRECTAMENTE

Key	Value	Description
Key	Value	Description

- CONSULTA NATIVA api/Partido/contar/estadio/{estadio} – Cuenta partidos por estadio

```
// CONSULTA NATIVA 2 - Contar por estadio
@GetMapping("/{contar/estadio/{estadio}")
public ResponseEntity<?> contarPorEstadio(@PathVariable String estadio) {
    return partidoService.contarPorEstadio(estadio);
}
```

```
package com.example.Quiz2Linea3.Repository;

import com.example.Quiz2Linea3.Model.Partido;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.time.LocalDate;
import java.util.List;

@Repository
public interface PartidoRepository extends JpaRepository<Partido, Integer> {

    // CONSULTA NATIVA 1: Buscar partidos por fecha
    @Query(value = "SELECT * FROM Partido WHERE fecha = ?1", nativeQuery = true)
    List<Partido> buscarPorFecha(LocalDate fecha);

    // CONSULTA NATIVA 2: Contar partidos jugados en un estadio
    @Query(value = "SELECT COUNT(*) FROM Partido WHERE estadio = ?1", nativeQuery = true)
    Integer contarPorEstadio(String estadio);
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUri}}/partido/contar/estadio/Metropolitano
- Params:** Query Params table with 2 columns: Key, Value, Description.
- Body:** 200 OK, 25 ms, 311 B. The response body is: 1 TOTAL DE PARTIDOS JUGADOS EN EL ESTADIO 'Metropolitano': 2

Key	Value	Description
Key	Value	Description

## 5. ESTADISTICASJUGADOR

- GET /api/estadisticas/listar - Lista todos los registros

```
// GET - LISTAR TODAS
public ResponseEntity<?> listar() {
    List<EstadisticasJugador> lista = estadisticasJugadorRepository.findAll();
    if (lista.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NO_CONTENT)
            .body("NO EXISTEN ESTADÍSTICAS REGISTRADAS");
    }
    return ResponseEntity.ok(lista);
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUri}} /estadisticas/listar
- Status:** 200 OK
- Time:** 33 ms
- Size:** 1.81 KB
- Body:** JSON array of 3 player statistics.

The JSON response is as follows:

```
[
  {
    "idEstadistica": 3,
    "minutosJugados": 85,
    "goles": 0,
    "asistencias": 1,
    "tarjetasAmarillas": 0,
    "tarjetasRojas": 0
  },
  {
    "idEstadistica": 4,
    "minutosJugados": 90,
    "goles": 1,
    "asistencias": 0,
    "tarjetasAmarillas": 1,
    "tarjetasRojas": 0
  },
  {
    "idEstadistica": 5,
    "minutosJugados": 88,
    "goles": 0,
    "asistencias": 0,
    "tarjetasAmarillas": 0,
    "tarjetasRojas": 0
  }
]
```

- GET /api/estadisticas/buscar/{id} - Busca un registro por ID

```
// GET - BUSCAR POR ID
public ResponseEntity<?> buscarPorId(Integer id) { 1 usage
    Optional<EstadisticasJugador> estadistica = estadisticasJugadorRepository.findById(id);
    if (estadistica.isPresent()) {
        return ResponseEntity.ok(estadistica.get());
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO SE ENCONTRÓ ESTADÍSTICA CON ID: " + id);
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** {{baseUri}} /estadisticas/buscar/5
- Send Button:** A blue button labeled "Send".
- Params Tab:** Shows "Query Params" with a table:

Key	Value	Description
Key	Value	Description

- Body Tab:** Shows the response body in JSON format:

```
1 {
2   "idEstadistica": 5,
3   "minutosJugados": 88,
4   "goles": 0,
5   "asistencias": 2,
6   "tarjetasAmarillas": 0,
7   "tarjetasRojas": 0
8 }
```

At the bottom of the interface, there is a status bar with icons for Runner, Start Proxy, Cookies, Vault, Trash, and a help icon.

- POST /api/estadisticas/guardar - Crea un nuevo registro

```
// POST - GUARDAR ESTADÍSTICAS
public ResponseEntity<?> guardar(EstadisticasJugador estadistica) { 1 usage
    try {
        if (estadistica.getMinutosJugados() == null) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                .body("LOS MINUTOS JUGADOS SON OBLIGATORIOS");
        }

        EstadisticasJugador nueva = estadisticasJugadorRepository.save(estadistica);
        return ResponseEntity.status(HttpStatus.CREATED)
            .body("ESTADÍSTICA REGISTRADA CON ID: " + nueva.getIdEstadistica());
    } catch (Exception e) {
        return ResponseEntity.internalServerError()
            .body("ERROR AL GUARDAR ESTADÍSTICA: " + e.getMessage());
    }
}
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** {{baseUri}}/estadisticas/guardar
- Body Type:** raw
- Request Body (JSON):**

```
{
  "minutosJugados": 90,
  "goles": 2,
  "asistencias": 1,
  "tarjetasAmarillas": 0,
  "tarjetasRojas": 0,
  "jugador": { "idJugador": 1 },
  "partido": { "idPartido": 3 }
}
```
- Response:** 201 Created, 20 ms, 284 B
- Response Body (JSON):**

```
{
  "ESTADÍSTICA REGISTRADA CON ID: 28"
}
```

The interface also includes tabs for Params, Authorization, Headers (10), Body, Scripts, and Settings. The Body tab is currently selected, showing the raw request and the JSON response.

- PUT /api/estadisticas/actualizar/{id} - Actualiza un registro existente

```
// PUT - ACTUALIZAR
public ResponseEntity<?> actualizar(Integer id, EstadisticasJugador datos) { 1 usage
    Optional<EstadisticasJugador> existente = estadisticasJugadorRepository.findById(id);
    if (existente.isEmpty()) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE ESTADÍSTICA CON ID: " + id);
    }

    EstadisticasJugador estadistica = existente.get();
    estadistica.setMinutosJugados(datos.getMinutosJugados());
    estadistica.setGoles(datos.getGoles());
    estadistica.setAsistencias(datos.getAsistencias());
    estadistica.setTarjetasAmarillas(datos.getTarjetasAmarillas());
    estadistica.setTarjetasRojas(datos.getTarjetasRojas());
    estadistica.setJugador(datos.getJugador());
    estadistica.setPartido(datos.getPartido());

    estadisticasJugadorRepository.save(estadistica);
    return ResponseEntity.ok().body("ESTADÍSTICA CON ID: " + id + " ACTUALIZADA CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** {{baseUri}} /estadisticas/actualizar/5
- Body Type:** raw
- Request Body (JSON):**

```
{
  "minutosJugados": 88,
  "goles": 1,
  "asistencias": 2,
  "tarjetasAmarillas": 1,
  "tarjetasRojas": 0,
  "jugador": { "idJugador": 1 },
  "partido": { "idPartido": 2 }
}
```
- Response:** 200 OK, 41 ms, 293 B
- Response Body:** ESTADÍSTICA CON ID: 5 ACTUALIZADA CORRECTAMENTE

- DELETE /api/estadisticas/eliminar/{id} - Elimina un registro

```
// DELETE - ELIMINAR
public ResponseEntity<?> eliminarPorId(Integer id) { 1 usage
    if (!estadisticasJugadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("NO EXISTE ESTADÍSTICA CON ID: " + id);
    }
    estadisticasJugadorRepository.deleteById(id);
    return ResponseEntity.ok().body("ESTADÍSTICA CON ID: " + id + " ELIMINADA CORRECTAMENTE");
}
```

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** {{baseUrl}}/estadisticas/eliminar/3
- Send Button:** A blue button labeled "Send".
- Params Tab:** Active, showing "Query Params" with a table:

Key	Value	Description
Key	Value	Description

- Body Tab:** Active, showing the response body in "Raw" format.
- Status:** 200 OK
- Time:** 40 ms
- Size:** 299 B
- Response Body:** ESTADÍSTICA CON ID: 3 ELIMINADA CORRECTAMENTE

At the bottom of the interface, there are icons for Runner, Start Proxy, Cookies, Vault, Trash, and a help icon.

- CONSULTA NATIVA api/estadisticas/promedio/goles/{idJugador} –  
Obtiene el promedio de goles para un jugador en específico

```
// CONSULTA NATIVA 2 - Promedio de goles por jugador
public ResponseEntity<?> promedioGolesPorJugador(Integer idJugador) { 1 usage
    Double promedio = estadisticasJugadorRepository.promedioGolesPorJugador(idJugador);
    if (promedio == null) promedio = 0.0;
    return ResponseEntity.ok(body: "PROMEDIO DE GOLES DEL JUGADOR CON ID " + idJugador + ": " + promedio);
}
}
```

```
package com.example.Quiz2Linea3.Repository;

import com.example.Quiz2Linea3.Model.EstadisticasJugador;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository 2 usages
public interface EstadisticasJugadorRepository extends JpaRepository<EstadisticasJugador, Integer> {

    // CONSULTA NATIVA 1: Buscar estadísticas por jugador
    @Query(value = "SELECT * FROM EstadisticasJugador WHERE id_jugador = ?1", nativeQuery = true) 1 usage
    List<EstadisticasJugador> buscarPorJugador(Integer idJugador);

    // CONSULTA NATIVA 2: Calcular promedio de goles por jugador
    @Query(value = "SELECT AVG(goles) FROM EstadisticasJugador WHERE id_jugador = ?1", nativeQuery = true) 1 usage
    Double promedioGolesPorJugador(Integer idJugador);
}
```

The screenshot shows a REST client interface with a GET request to the endpoint `{{baseUrl}}/estadisticas/buscar/jugador/1`. The response is a 200 OK status with a JSON body containing two player statistics objects.

Key	Value	Description
Key	Value	Description

```

1  [
2    {
3      "idEstadistica": 5,
4      "minutosJugados": 80,
5      "goles": 1,
6      "asistencias": 2,
7      "tarjetasAmarillas": 1,
8      "tarjetasRojas": 0
9    },
10   {
11     "idEstadistica": 28,
12     "minutosJugados": 90,
13     "goles": 2,
14     "asistencias": 1,
15     "tarjetasAmarillas": 0,
16     "tarjetasRojas": 0
17   }
18 ]
  
```



## **CONCLUSIONES**

El proyecto Quiz2Linea3 cumple con los requerimientos de un sistema completo de gestión de datos relacionales implementado en Spring Boot. Las entidades se encuentran correctamente relacionadas mediante JPA, las consultas funcionan correctamente y los endpoints REST permiten el acceso total a las operaciones CRUD.