QUIZ2LINEA3

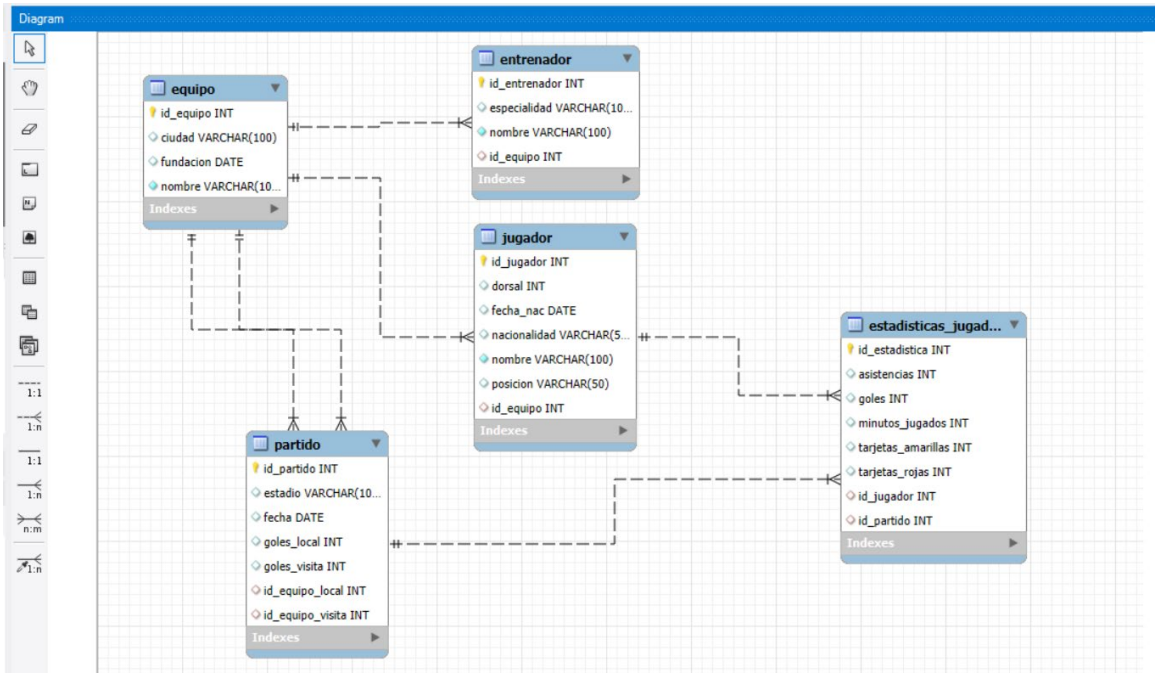
RAUL ALEXANDER SUESCUN PEREZ
LÍNEA DE PROFUNDIZACIÓN 3
FECHA: 14 DE OCTUBRE DE 2025
DESARROLLADO CON SPRING BOOT Y MYSQL

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

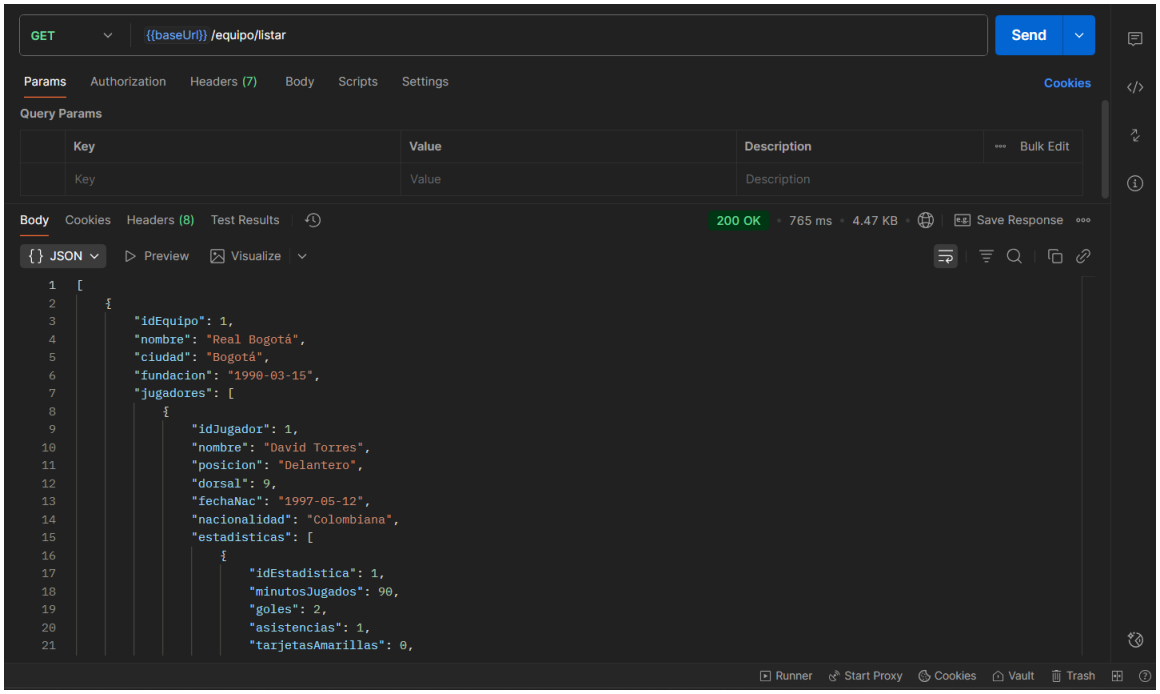
.env:
DB_URL=jdbc:mysql://localhost:3315/quiz2linea3
DB_USERNAME=root
DB_PASSWORD=1234

COLECCIÓN DE CONSULTAS, METODOS, PARÁMETROS Y RESULTADOS

1. EQUIPO

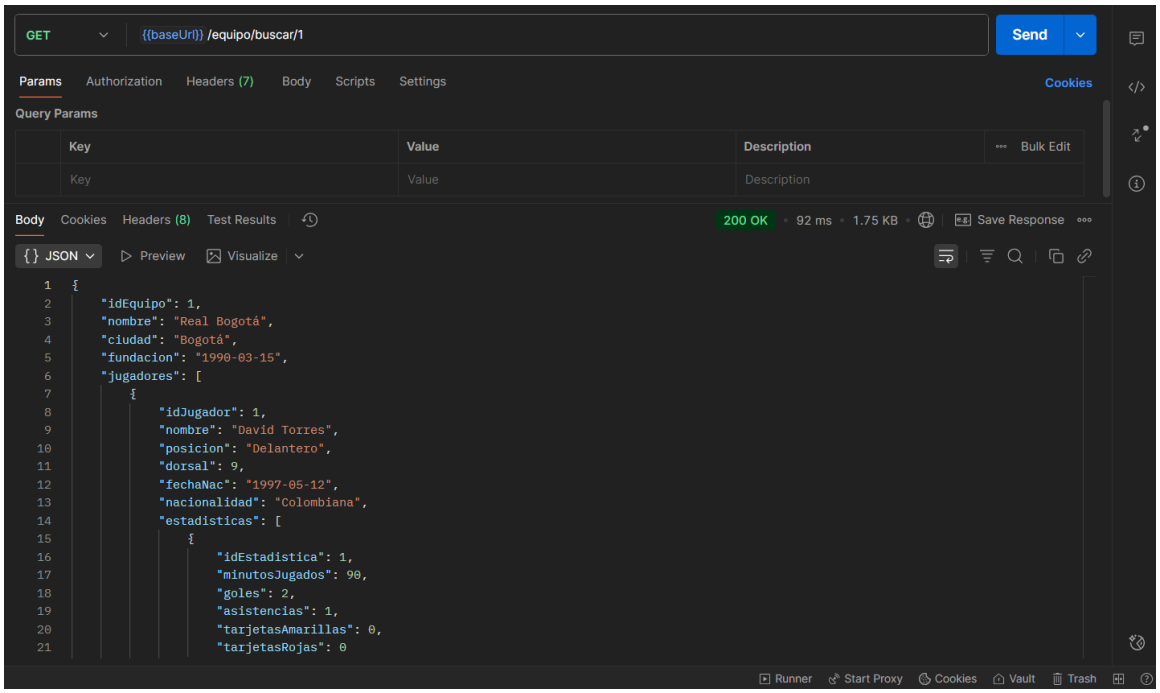
- GET /api/equipo/listar - Lista todos los registros

```java
// 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);
}
```



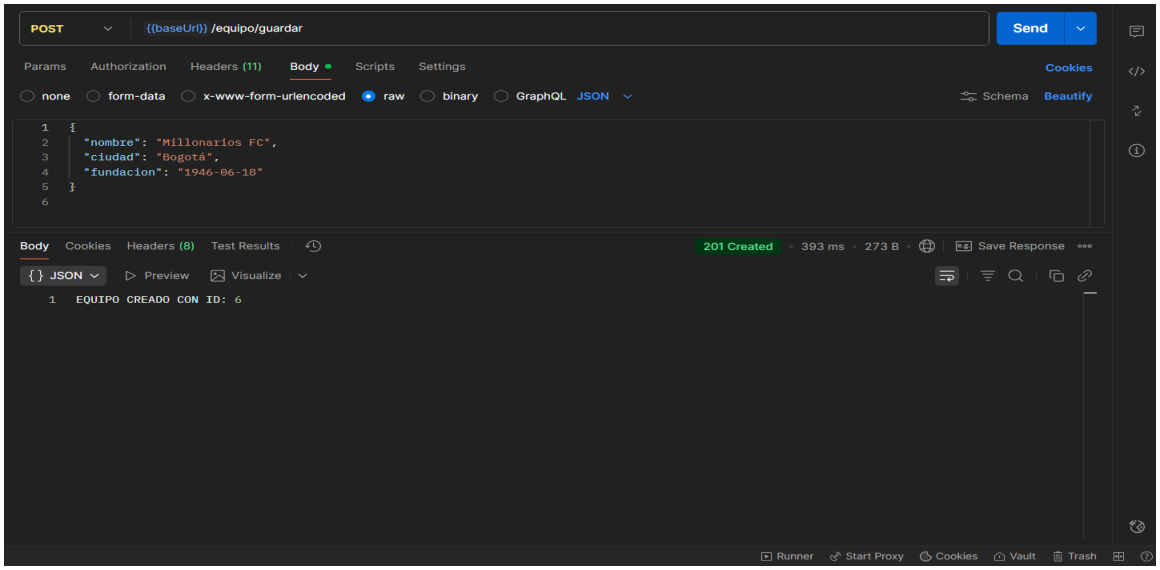
- GET /api/equipo/buscar/{id} - Busca un registro por ID

```java
// 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());
    }
}
```

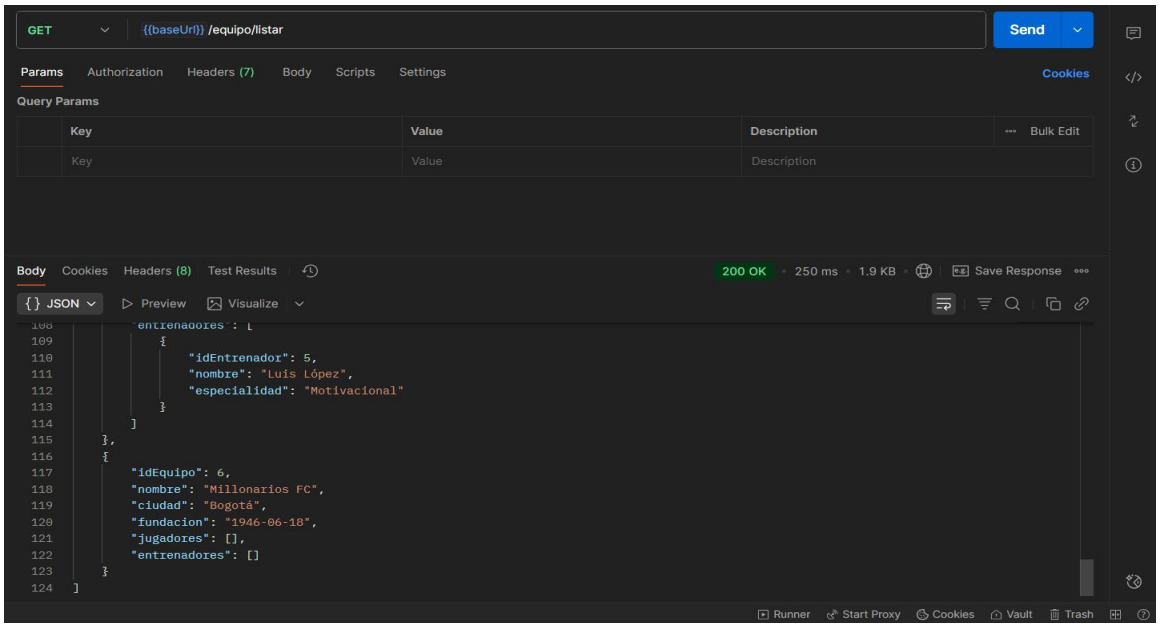


- POST /api/equipo/guardar - Crea un nuevo registro

```java
// 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());
    }
}
```

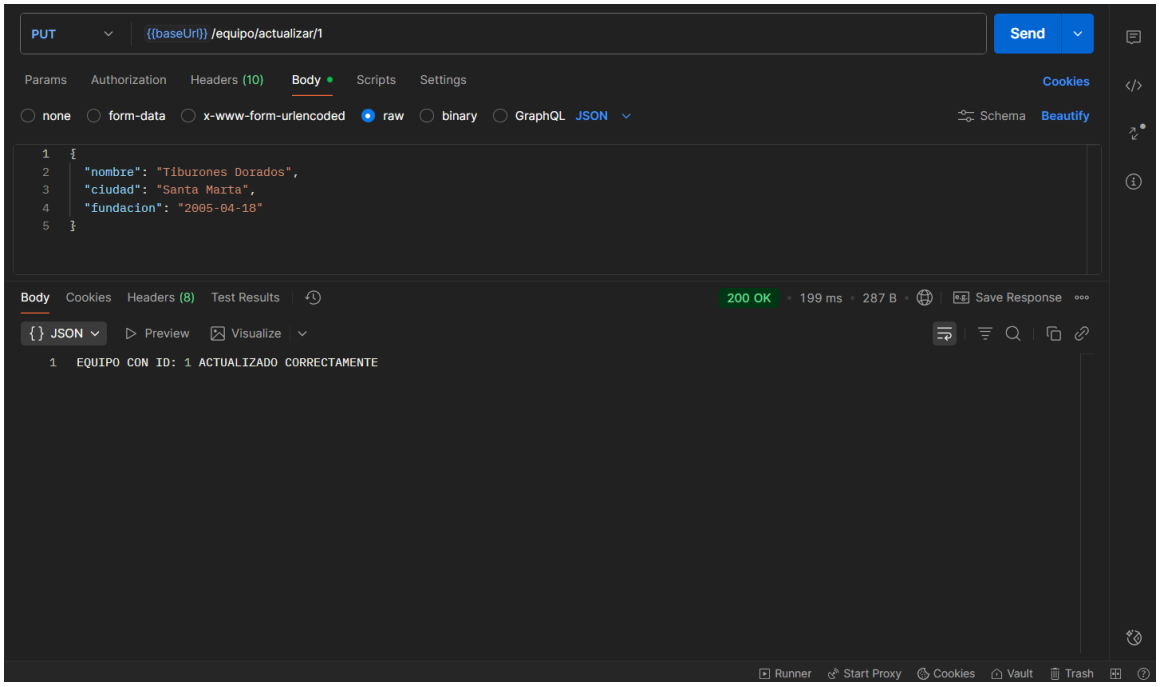


Verificado al listar:

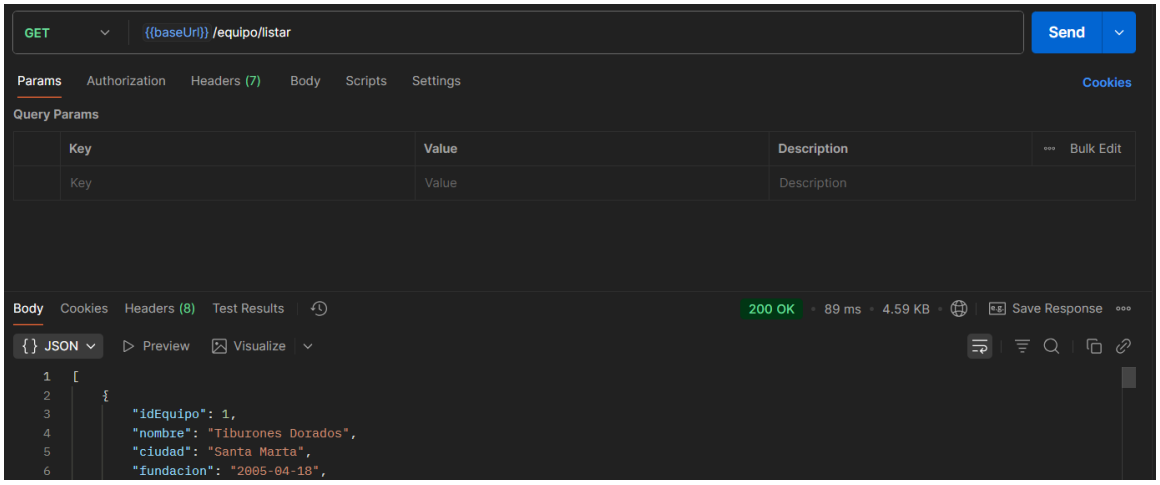


- PUT /api/equipo/actualizar/{id} - Actualiza un registro existente

```java
// 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");
}
```

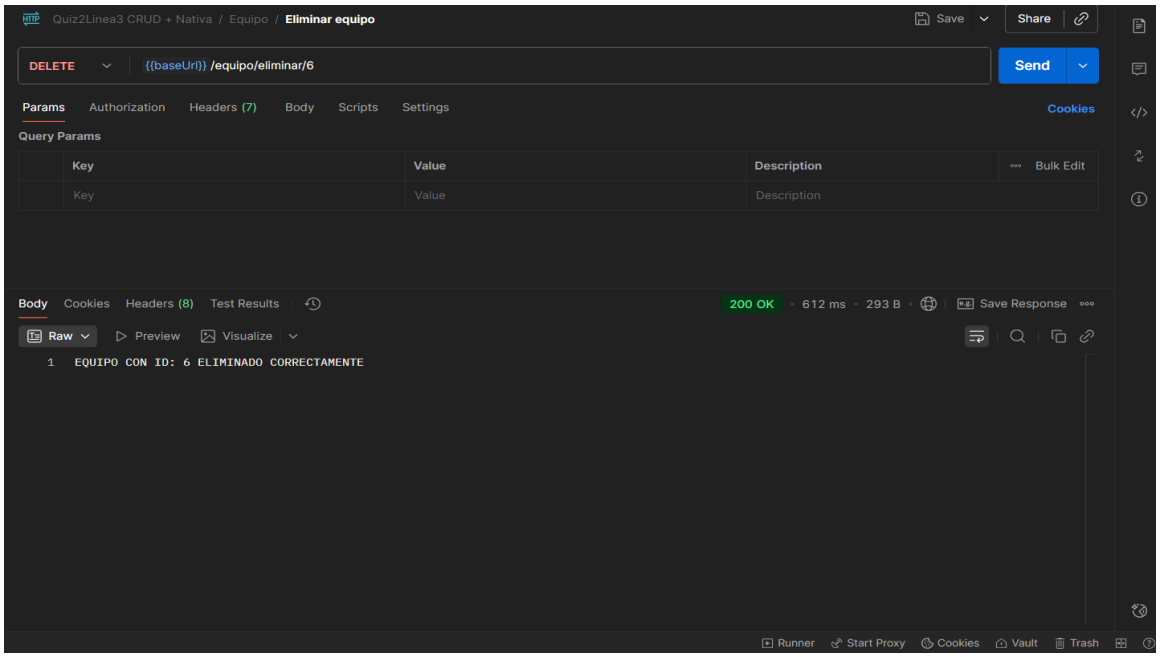


Verificado al listar

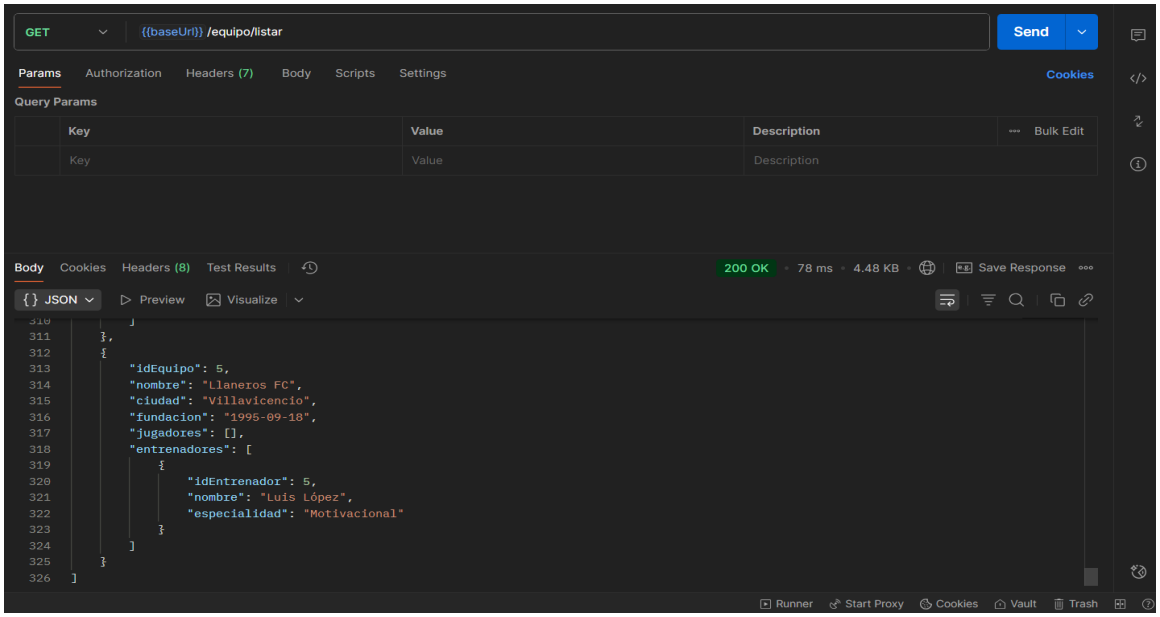


- DELETE /api/equipo/eliminar/{id} - Elimina un registro

```java
// 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");
}
```



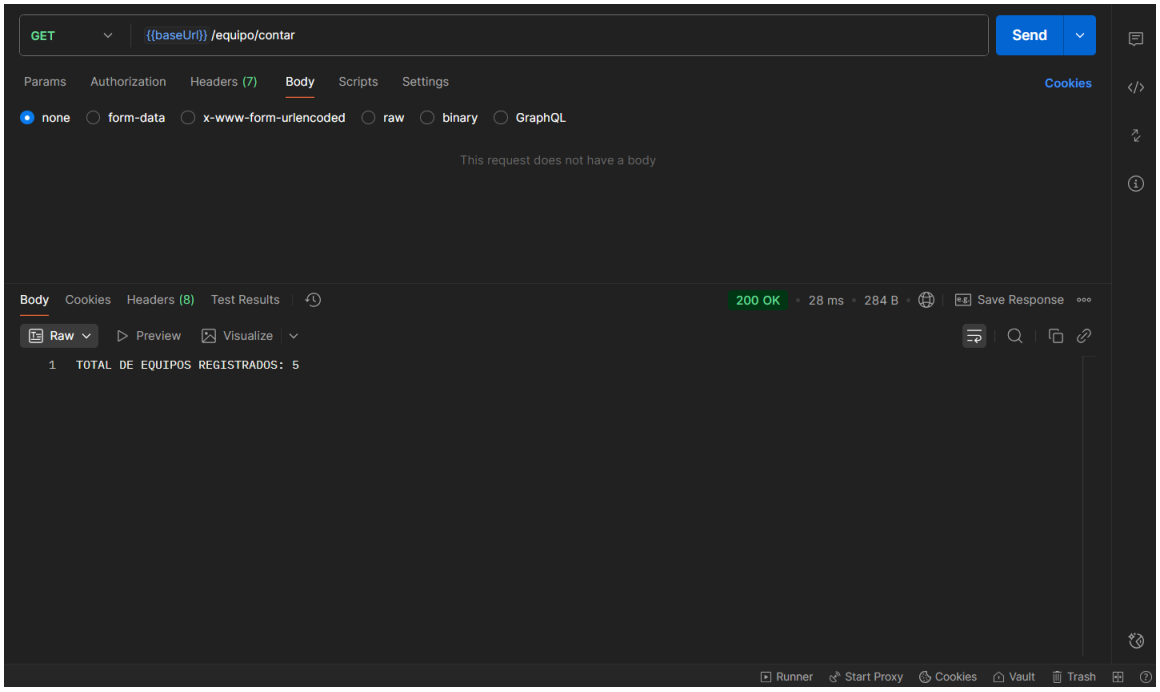
Verificado al listar



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

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

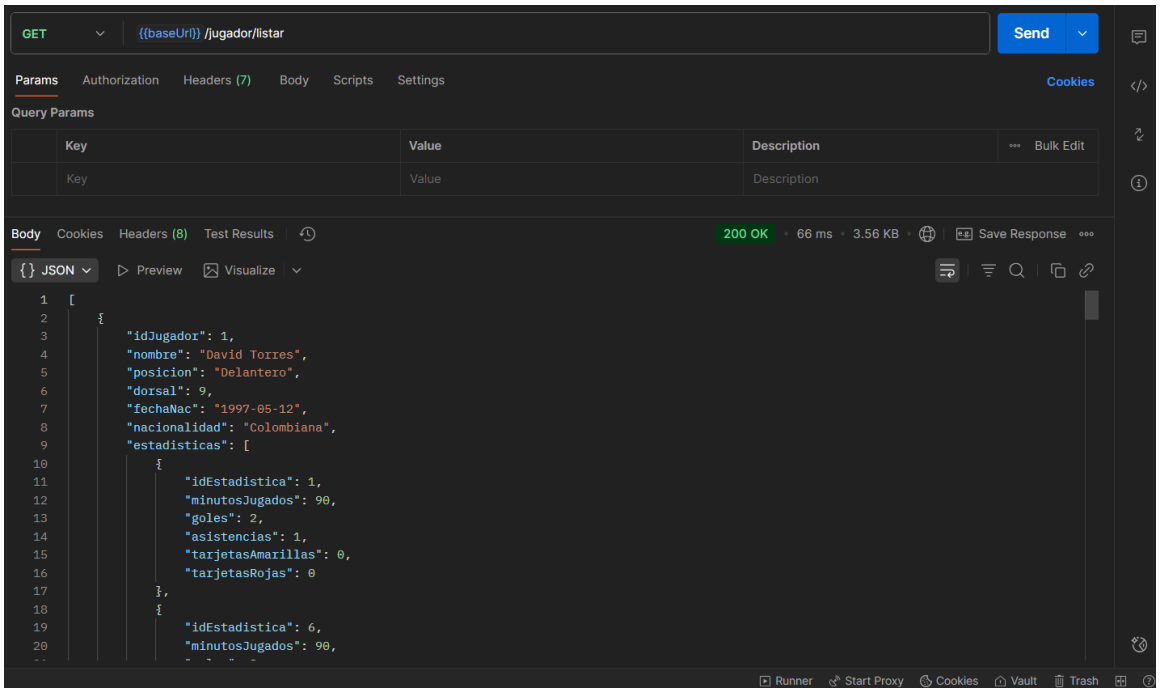
```java
package com.example.Quiz2Linea3.Repository;

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();
}
```



2. JUGADOR

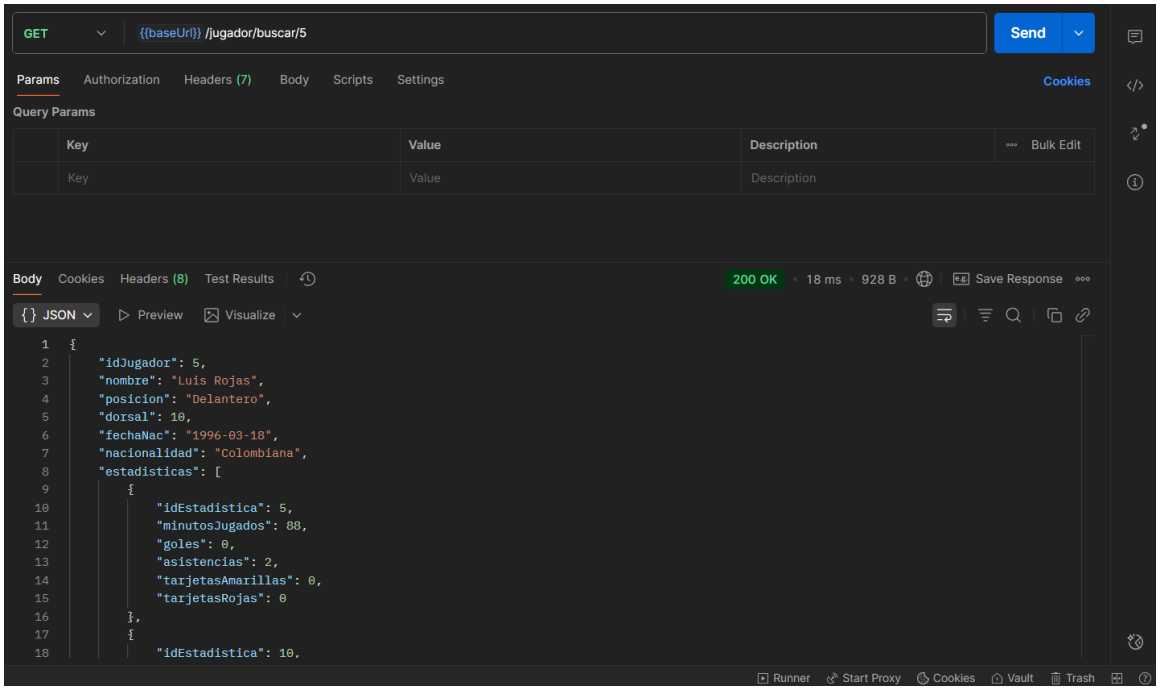
- GET /api/jugador/listar - Lista todos los registros

```java
// 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);
}
```



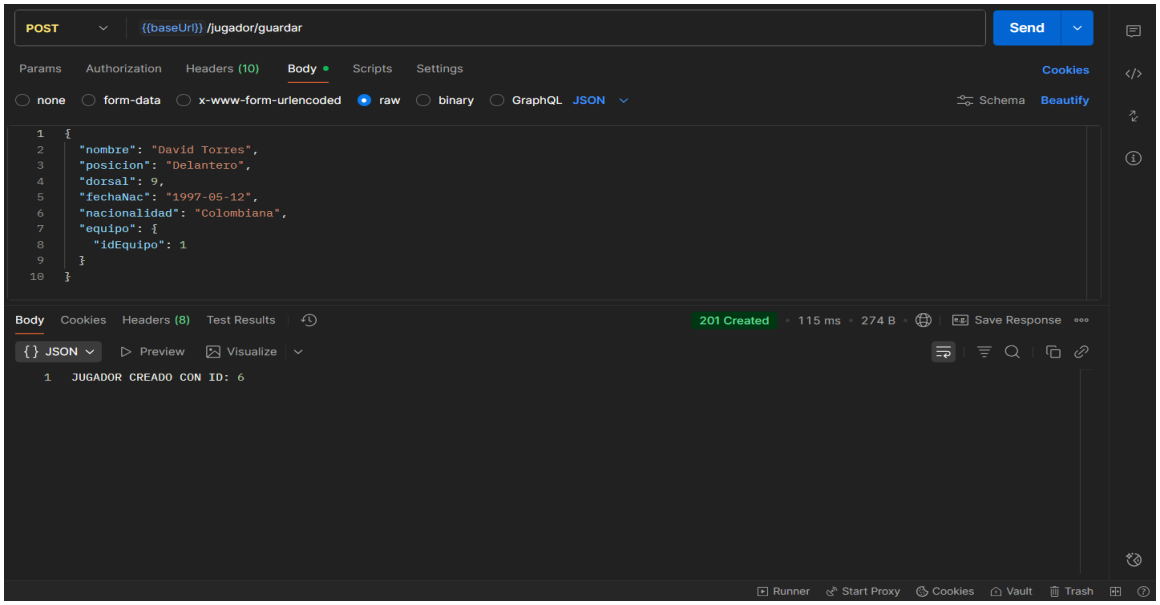
- GET /api/jugador/buscar/{id} - Busca un registro por ID

```java
// 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());
    }
}
```



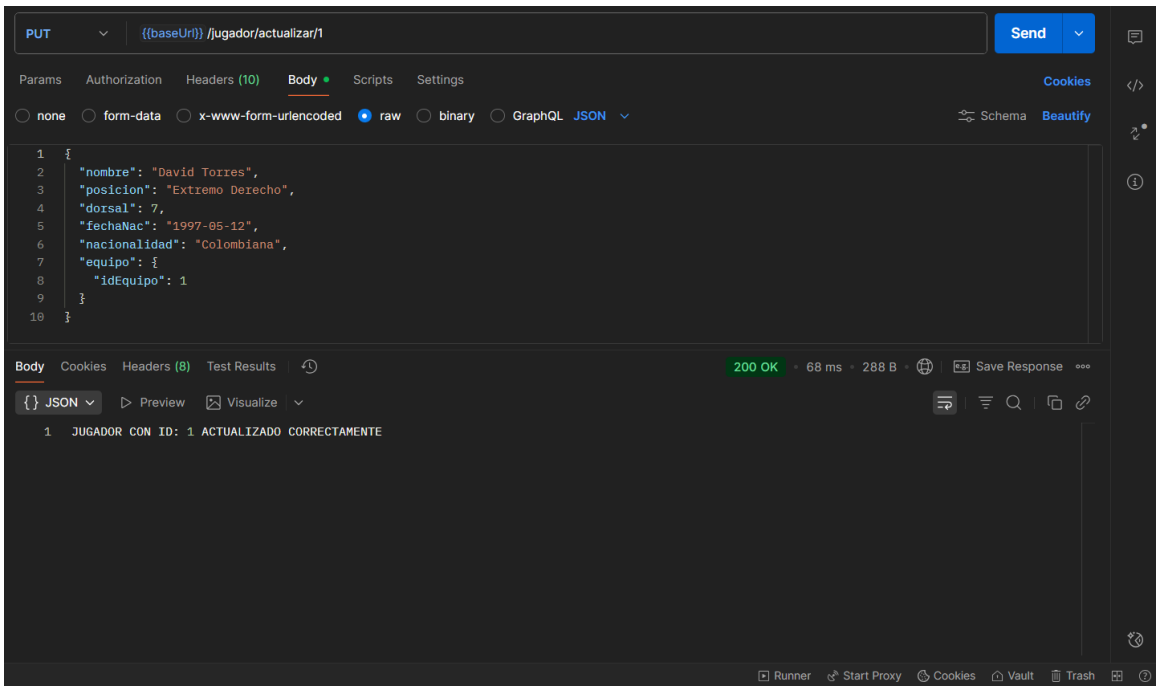
- POST /api/jugador/guardar - Crea un nuevo registro

```java
// 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());
    }
}
```



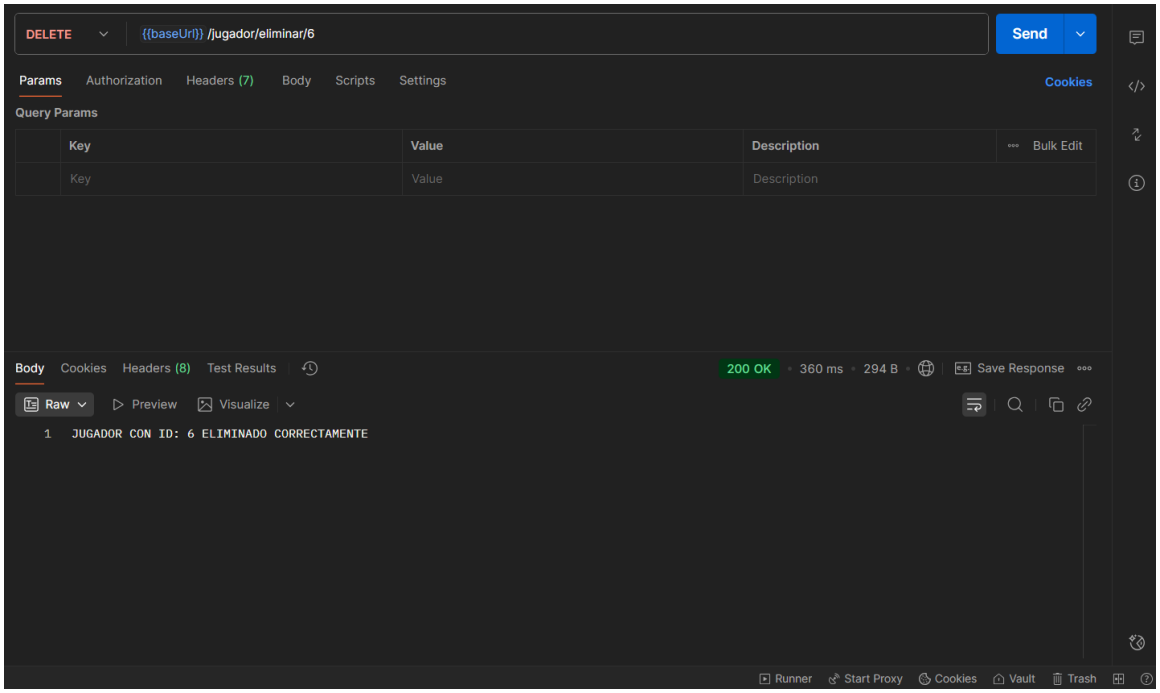
- PUT /api/jugador/actualizar/{id} - Actualiza un registro existente

```java
// 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");
}
```



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

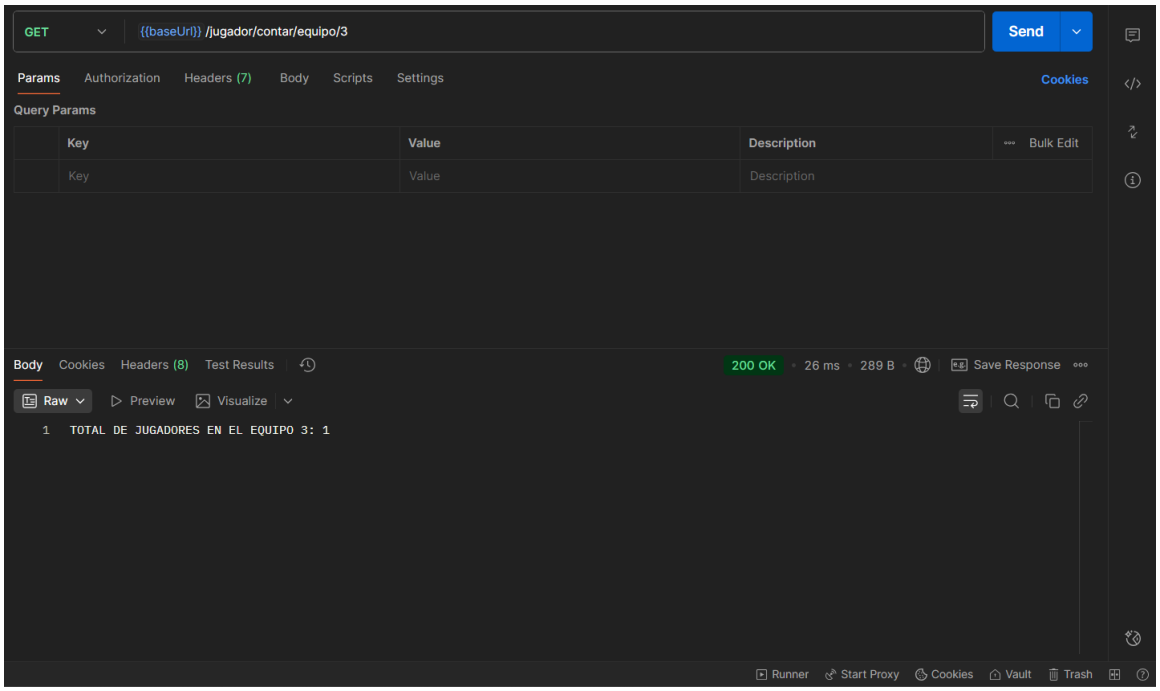
```java
// DELETE - ELIMINAR JUGADOR
public ResponseEntity<?> eliminarPorId(Integer id) {  1 usage
    if (!jugadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO EXISTE JUGADOR CON ID: " + id);
    }
    jugadorRepository.deleteById(id);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



- 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);
    }
}
```

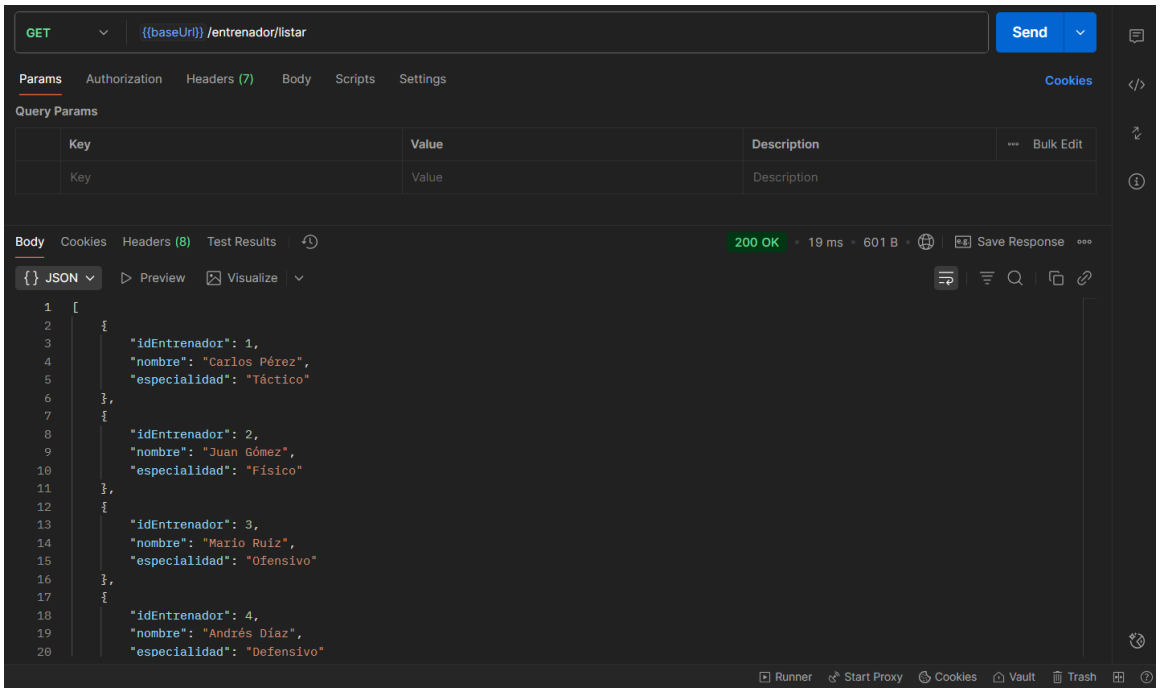
```
package com.example.Quiz2Linea3.Repository;

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);
}
```



3. ENTRENADOR

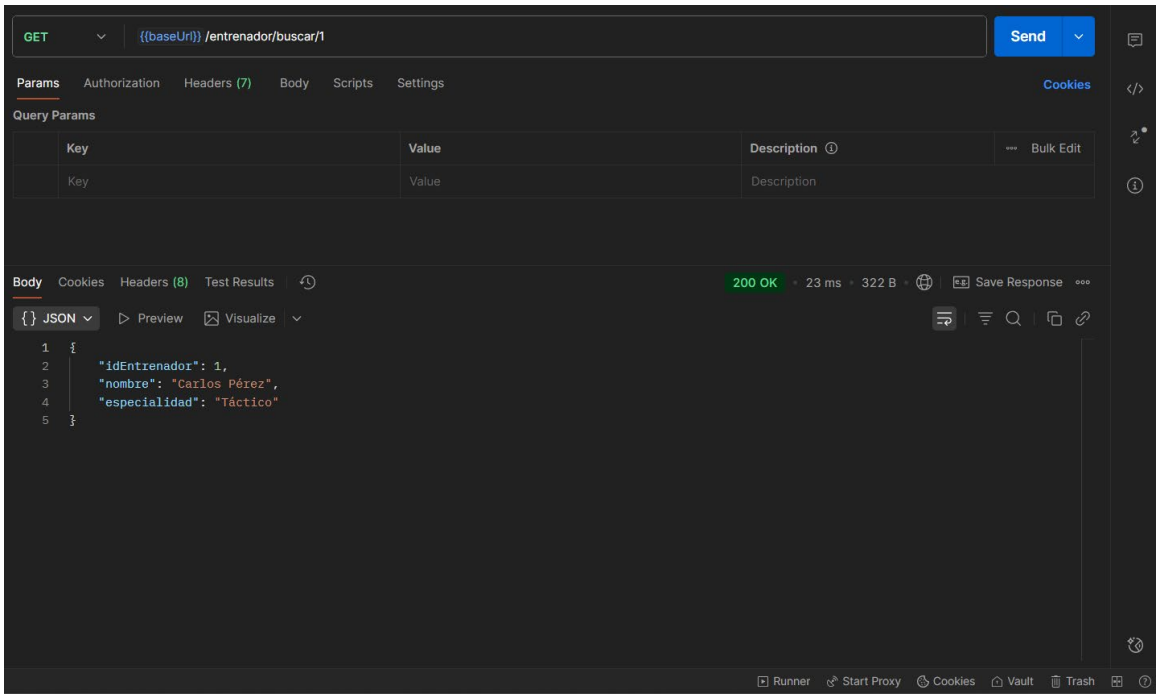
- GET /api/entrenador/listar - Lista todos los registros

```java
// 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);
}
```



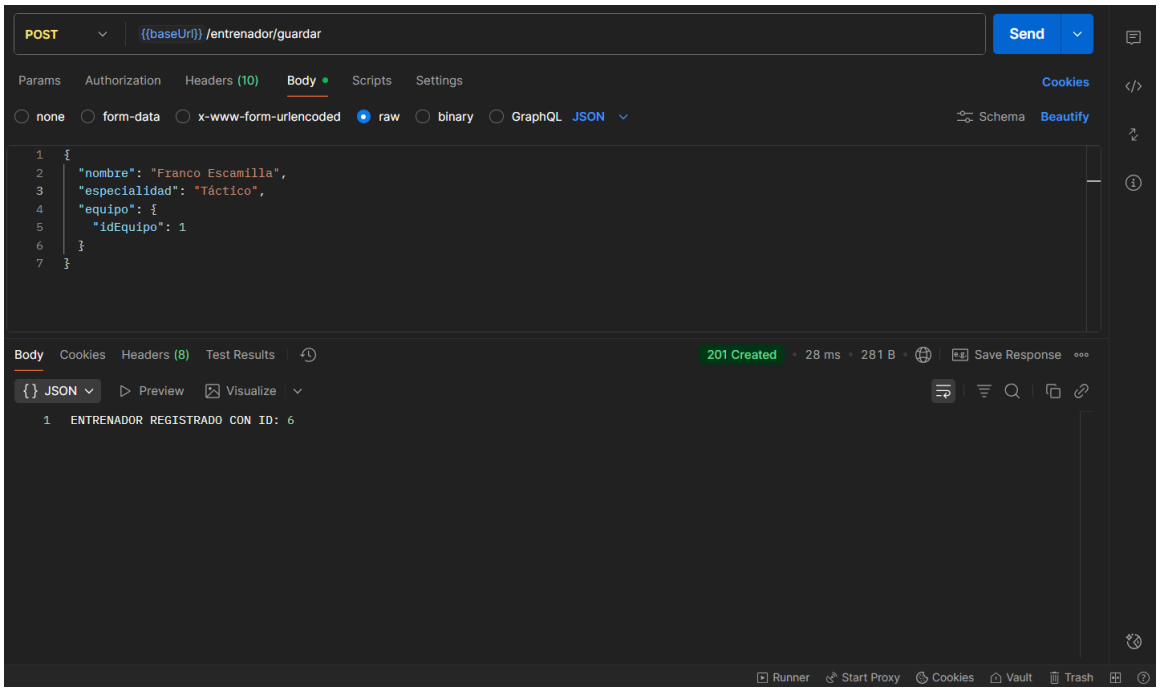
- GET /api/entrenador/buscar/{id} - Busca un registro por ID

```java
// 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);
    }
}
```



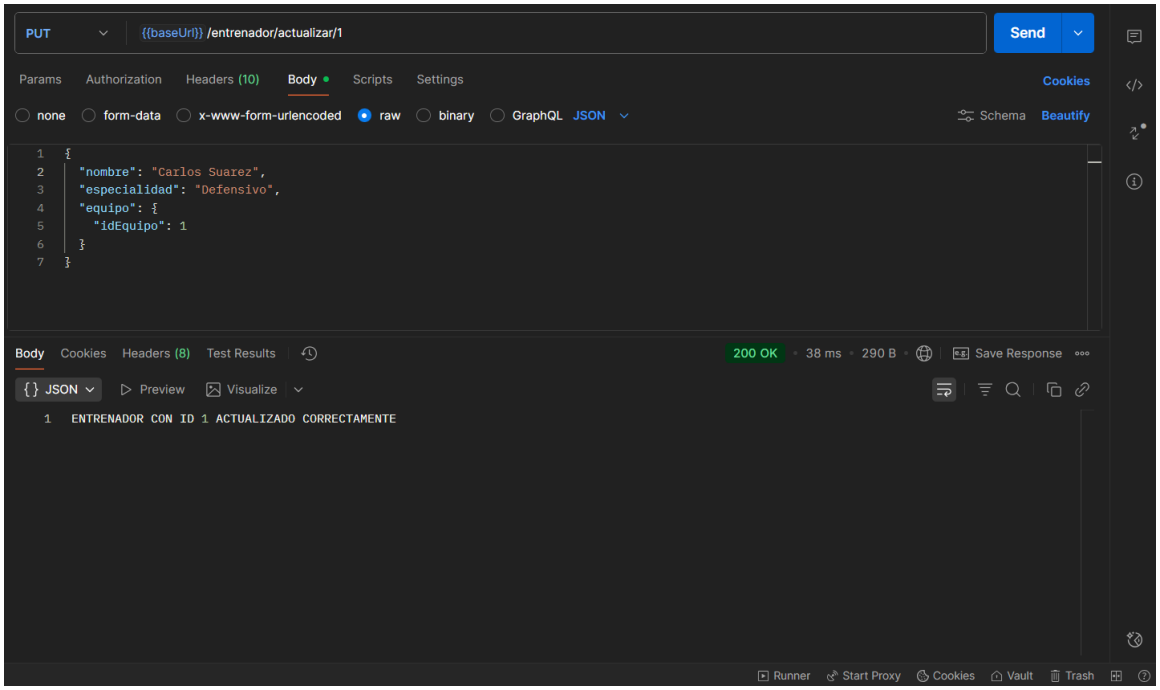
- POST /api/entrenador/guardar - Crea un nuevo registro

```java
// 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());
    }
}
```



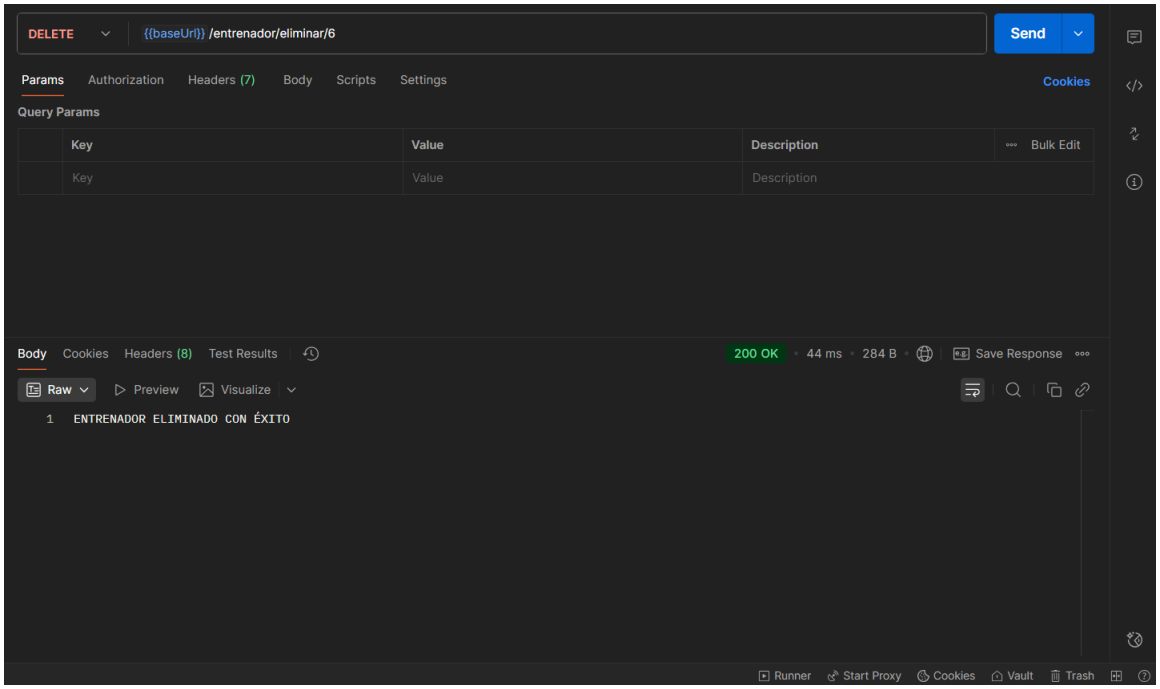
- PUT /api/entrenador/actualizar/{id} - Actualiza un registro existente

```java
// 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");
}
```



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

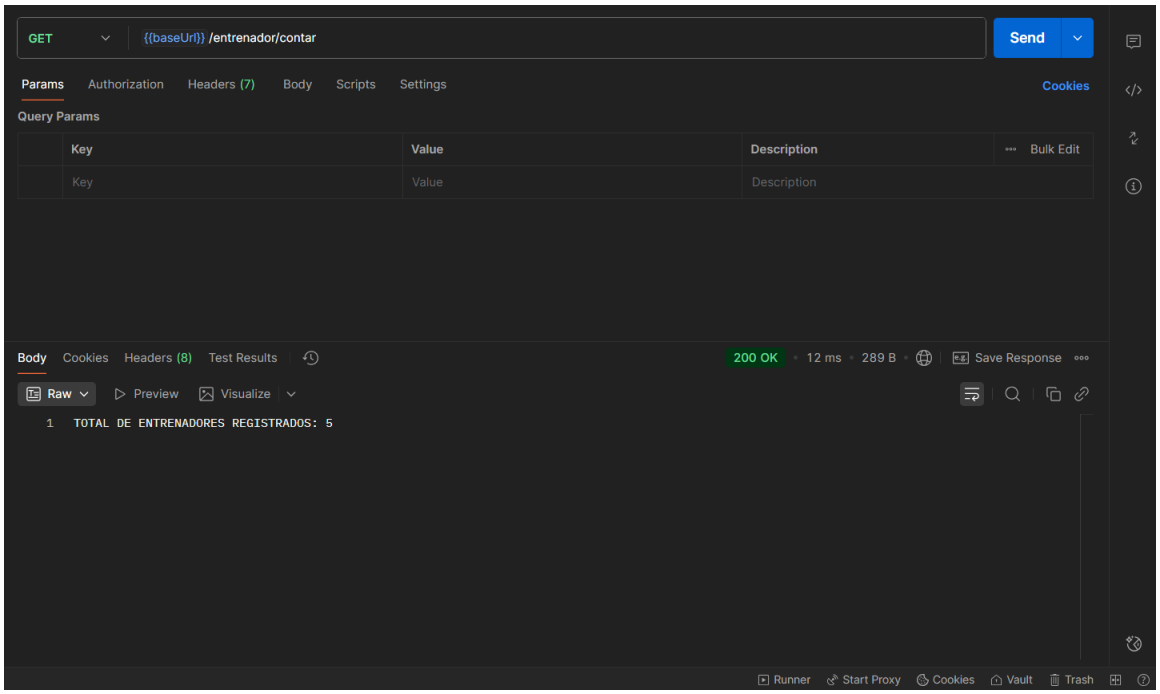
```java
// DELETE - ELIMINAR ENTRENADOR
public ResponseEntity<?> eliminarPorId(Integer id) {  1 usage
    if (!entrenadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO EXISTE ENTRENADOR CON ID: " + id);
    }
    entrenadorRepository.deleteById(id);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



- CONSULTA NATIVA api/entrenador/contar – Cuenta entrenadores

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

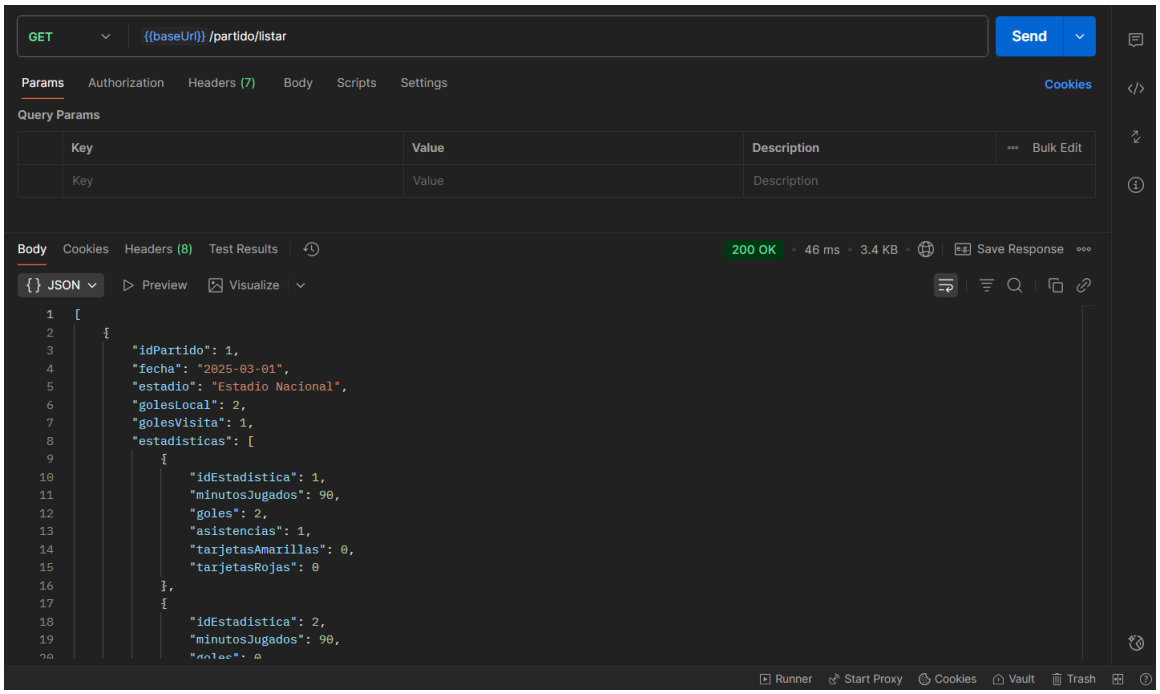
```java
package com.example.Quiz2Linea3.Repository;

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  2 usages
public interface EntrenadorRepository extends JpaRepository<Entrenador, Integer> {

    @Query(value = "SELECT * FROM Entrenador WHERE especialidad = :especialidad", nativeQuery = true)  1 usa
    List<Entrenador> buscarPorEspecialidad(@Param("especialidad") String especialidad);

    @Query(value = "SELECT COUNT(*) FROM Entrenador", nativeQuery = true)  1 usage
    Integer contarEntrenadores();

    @Query(value = "SELECT COUNT(*) FROM Entrenador WHERE id_equipo = :idEquipo", nativeQuery = true)  1 usa
    Integer contarPorEquipo(@Param("idEquipo") Integer idEquipo);
}
```



4. PARTIDO

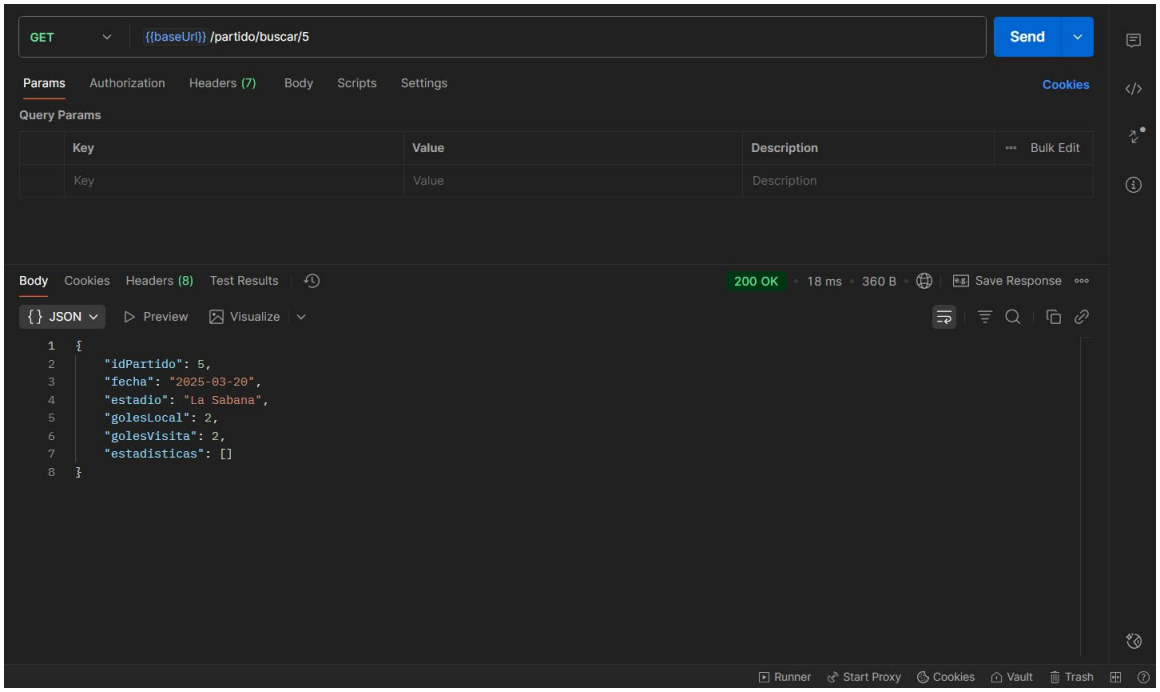
- GET /api/partido/listar - Lista todos los registros

```java
// 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);
}
```

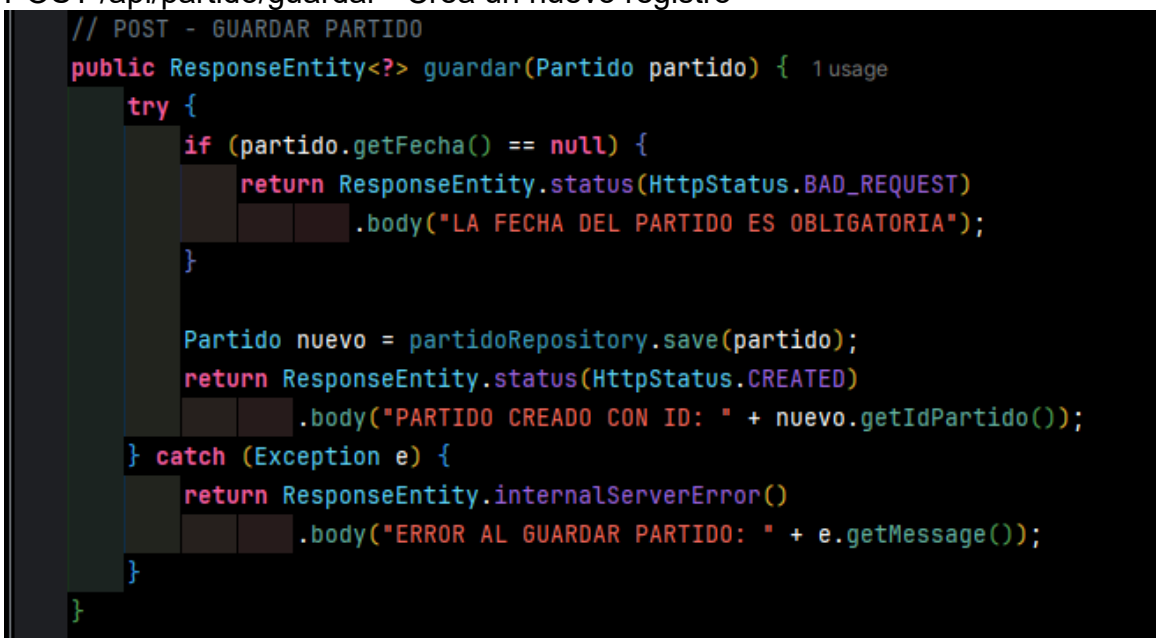


- GET /api/partido/buscar/{id} - Busca un registro por ID

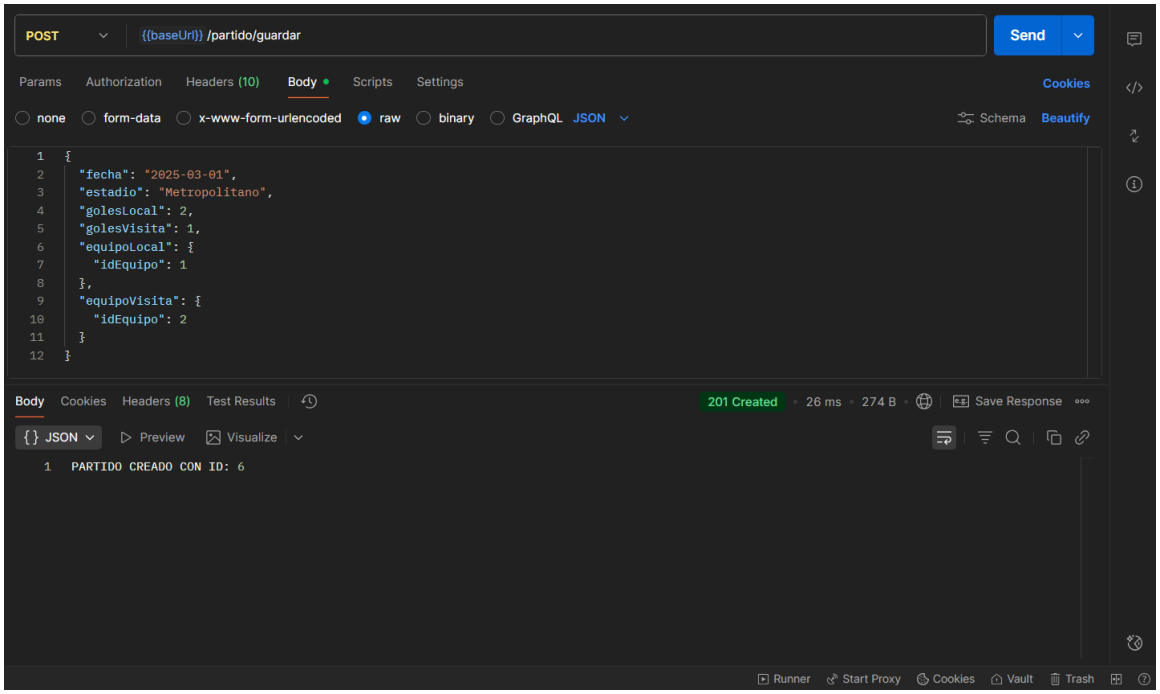
```java
// GET - BUSCAR POR ID
public ResponseEntity<?> buscarPorId(Integer id) {  1 usage
    Optional<Partido> partido = partidoRepository.findById(id);
    if (partido.isPresent()) {
        return ResponseEntity.ok(partido.get());
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)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



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

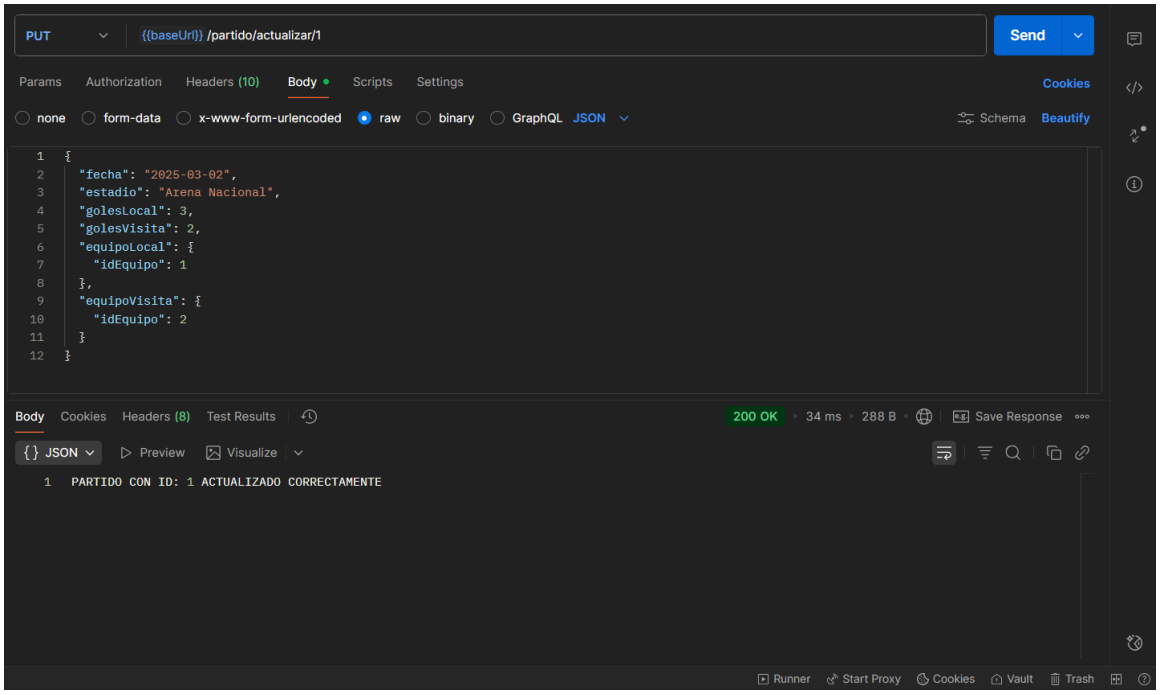


```java
// POST - GUARDAR PARTIDO
public ResponseEntity<?> guardar(Partido partido) {  1 usage
    try {
        if (partido.getFecha() == null) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST)
                    .body("LA FECHA DEL PARTIDO ES OBLIGATORIA");
        }

        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());
    }
}
```



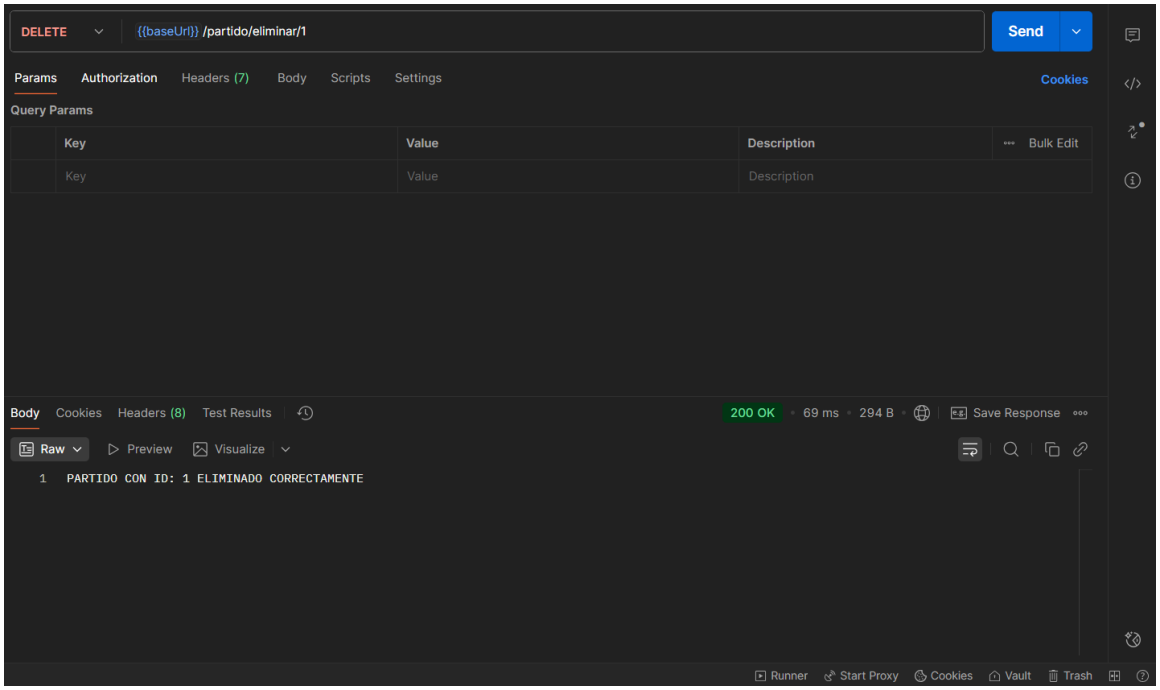
- PUT /api/partido/actualizar/{id} - Actualiza un registro existente

```java
// 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");
}
```



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

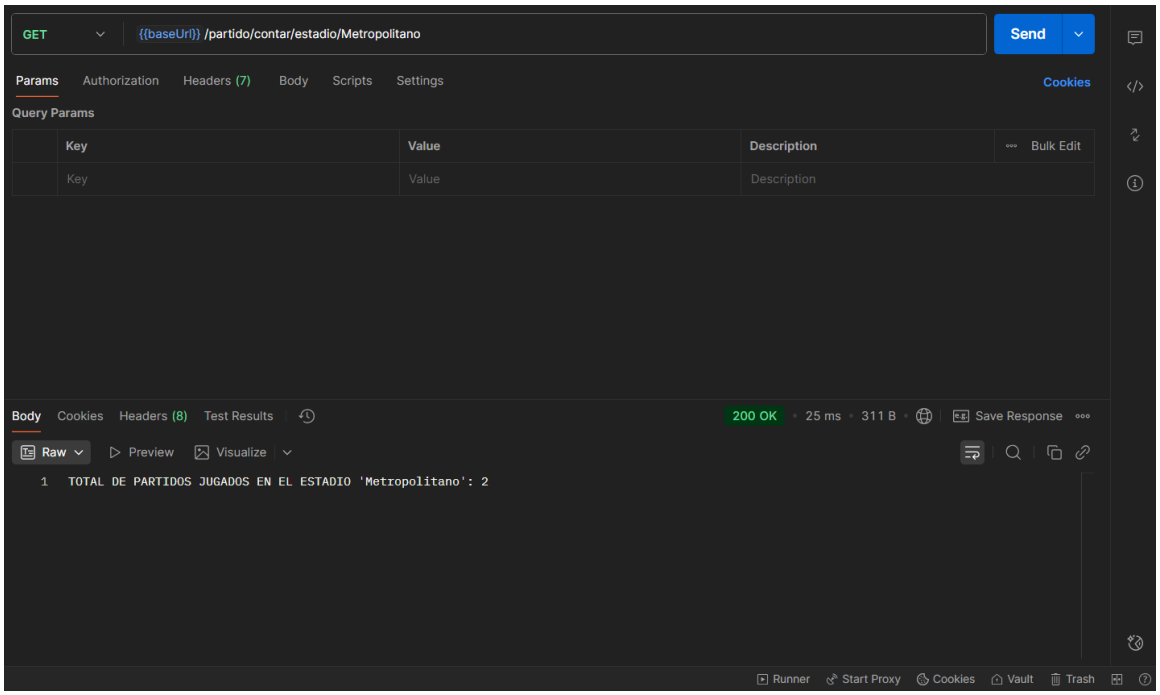
```java
// DELETE - ELIMINAR PARTIDO
public ResponseEntity<?> eliminarPorId(Integer id) {  1 usage
    if (!partidoRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO EXISTE PARTIDO CON ID: " + id);
    }
    partidoRepository.deleteById(id);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



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

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

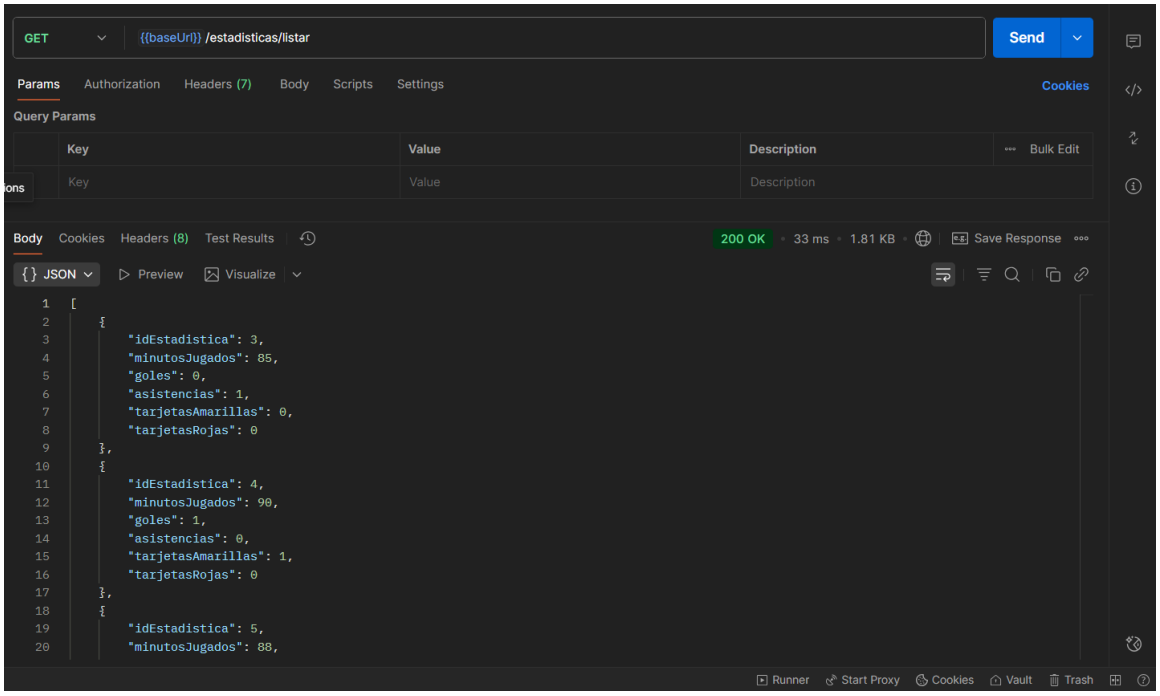
```java
package com.example.Quiz2Linea3.Repository;

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  2 usages
public interface PartidoRepository extends JpaRepository<Partido, Integer> {

    // CONSULTA NATIVA 1: Buscar partidos por fecha
    @Query(value = "SELECT * FROM Partido WHERE fecha = ?1", nativeQuery = true)  1 usage
    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)  1 usage
    Integer contarPorEstadio(String estadio);
}
```



5. ESTADISTICASJUGADOR

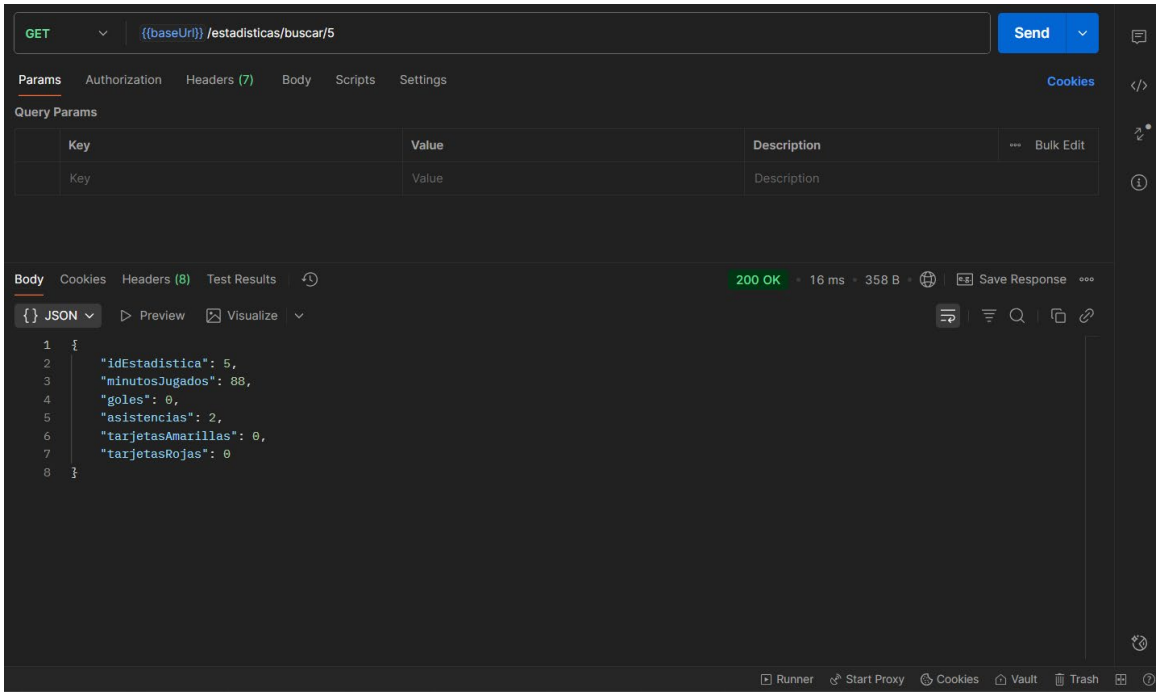
- GET /api/estadisticas/listar - Lista todos los registros

```java
// 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);
}
```



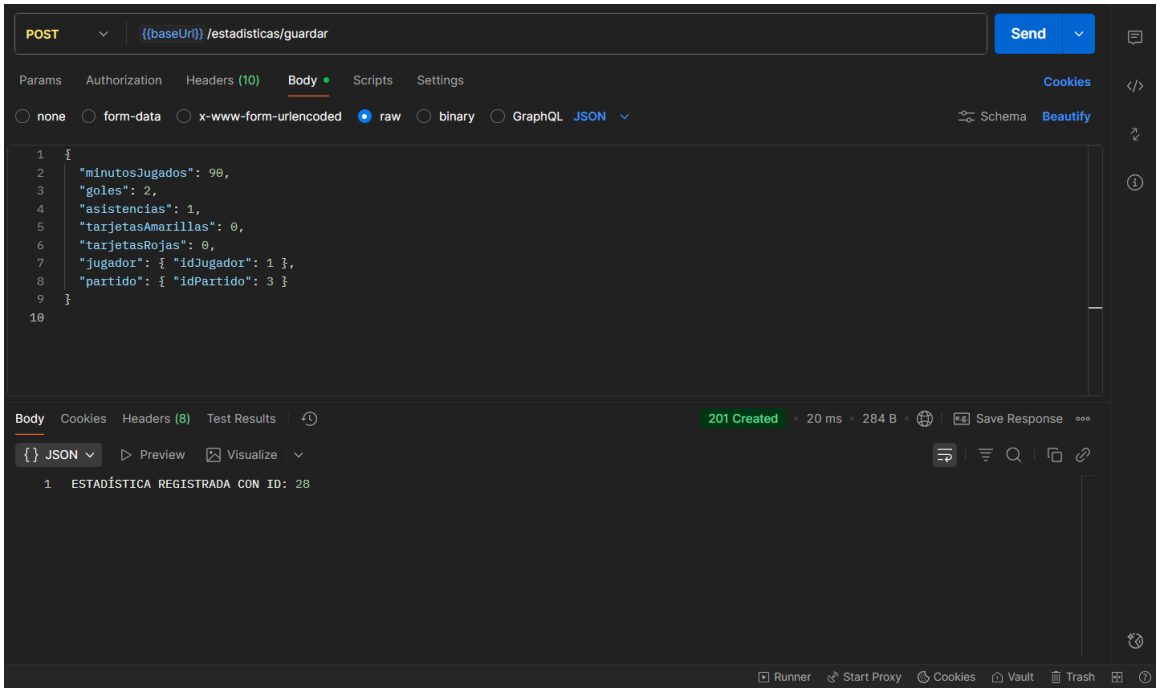
- GET /api/estadisticas/buscar/{id} - Busca un registro por ID

```java
// 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);
    }
}
```



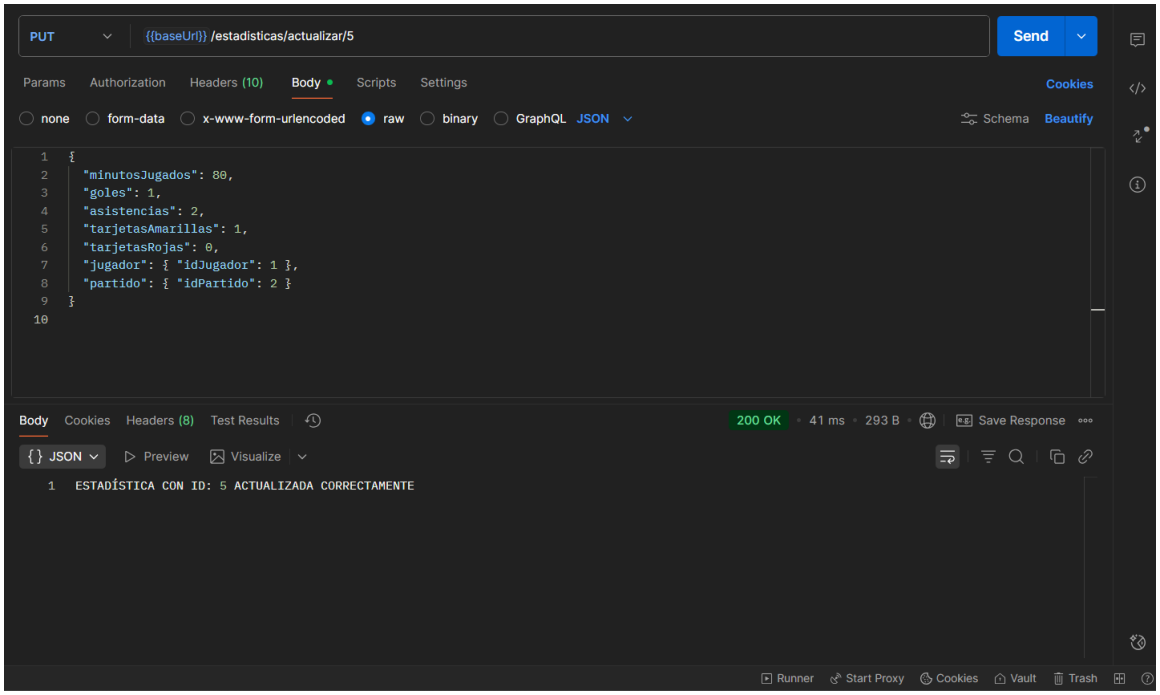
- POST /api/estadisticas/guardar - Crea un nuevo registro

```java
// 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());
    }
}
```



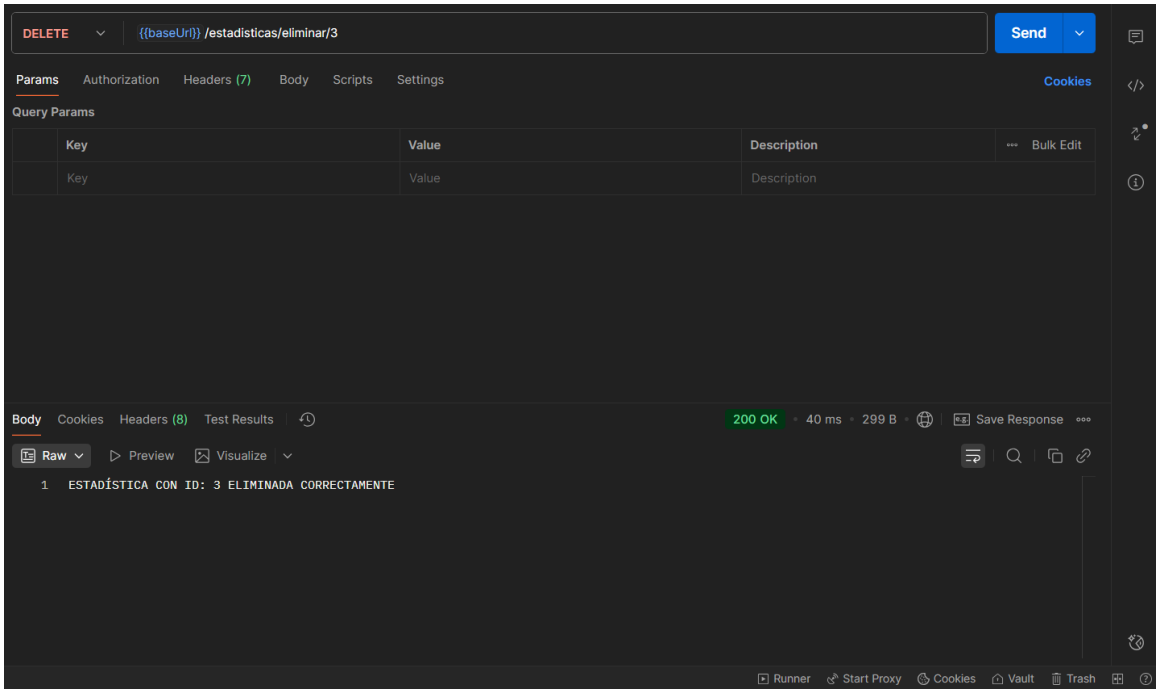
- PUT /api/estadisticas/actualizar/{id} - Actualiza un registro existente

```java
// 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");
}
```



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

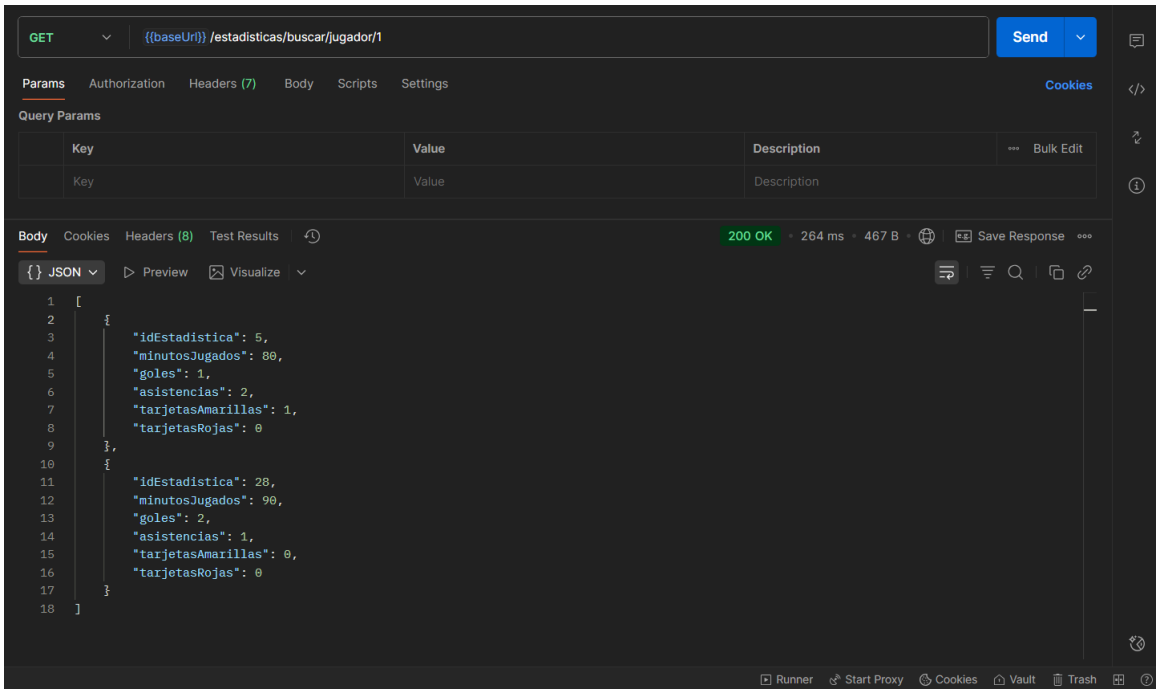
```java
// DELETE - ELIMINAR
public ResponseEntity<?> eliminarPorId(Integer id) {  1 usage
    if (!estadisticasJugadorRepository.existsById(id)) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
                .body("NO EXISTE ESTADÍSTICA CON ID: " + id);
    }
    estadisticasJugadorRepository.deleteById(id);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



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

```java
    // 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);
    }
}
```

```java
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);
}
```



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.