



**UNIVERSIDAD  
FRANCISCO GAVIDIA**  
Tecnología, Innovación y Calidad

**Facultad:** Ingeniería y Sistemas

**Módulo:** Desarrollo de aplicaciones web

**Docente:** Carlos Boris Martínez Calzadia

**Grupo:** 01

**Alumno:** Marco Arturo Cassio Cardona CC104809

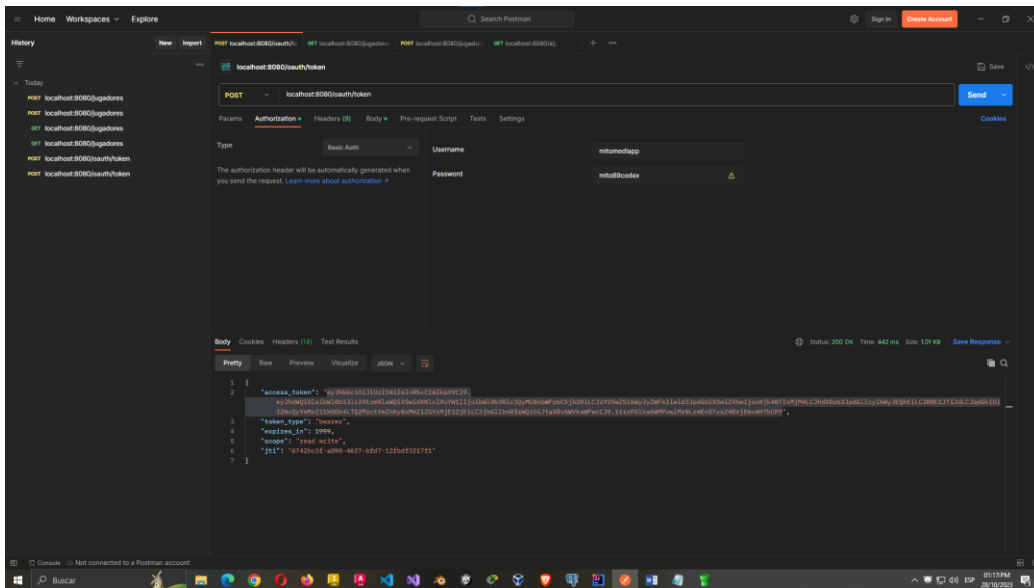
**Actividad:** Laboratorio 3

**Fecha de Entrega:** 28 de octubre de 2023

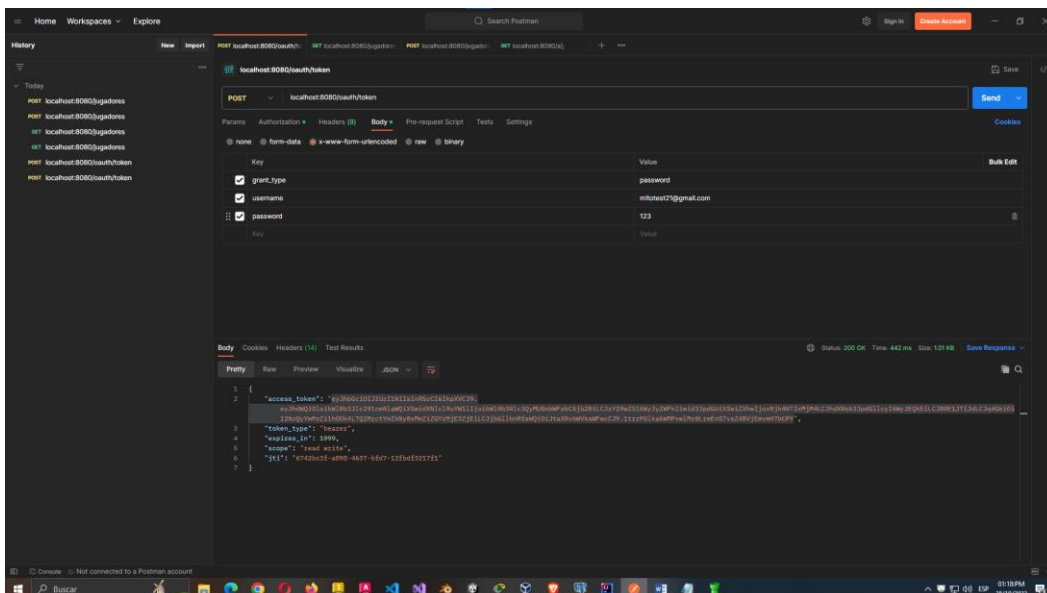
# Laboratorio 3

## Postman

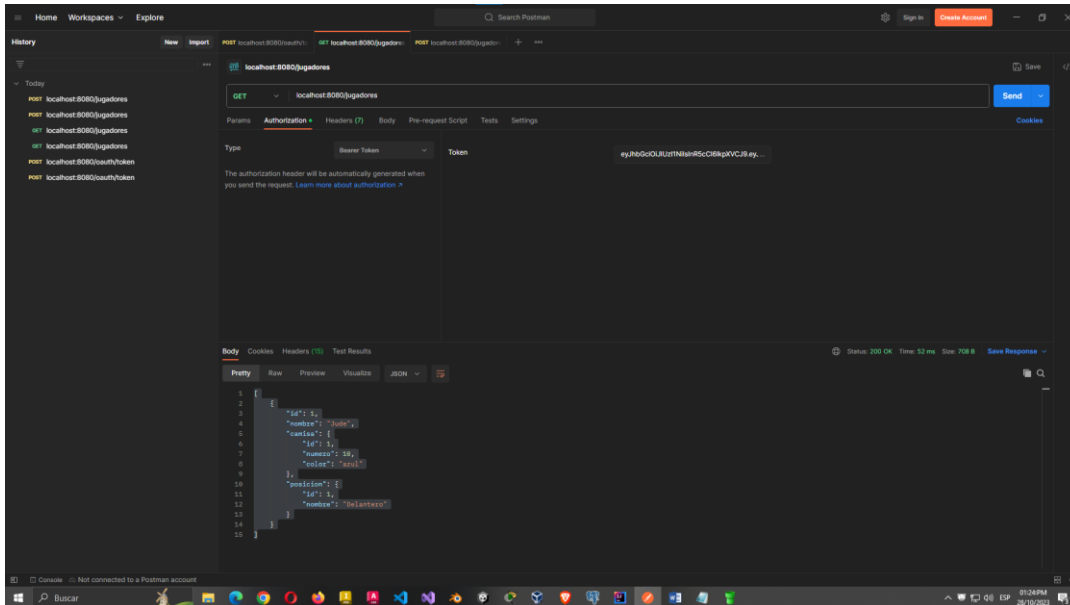
En POST en pestaña authorization generamos el Token



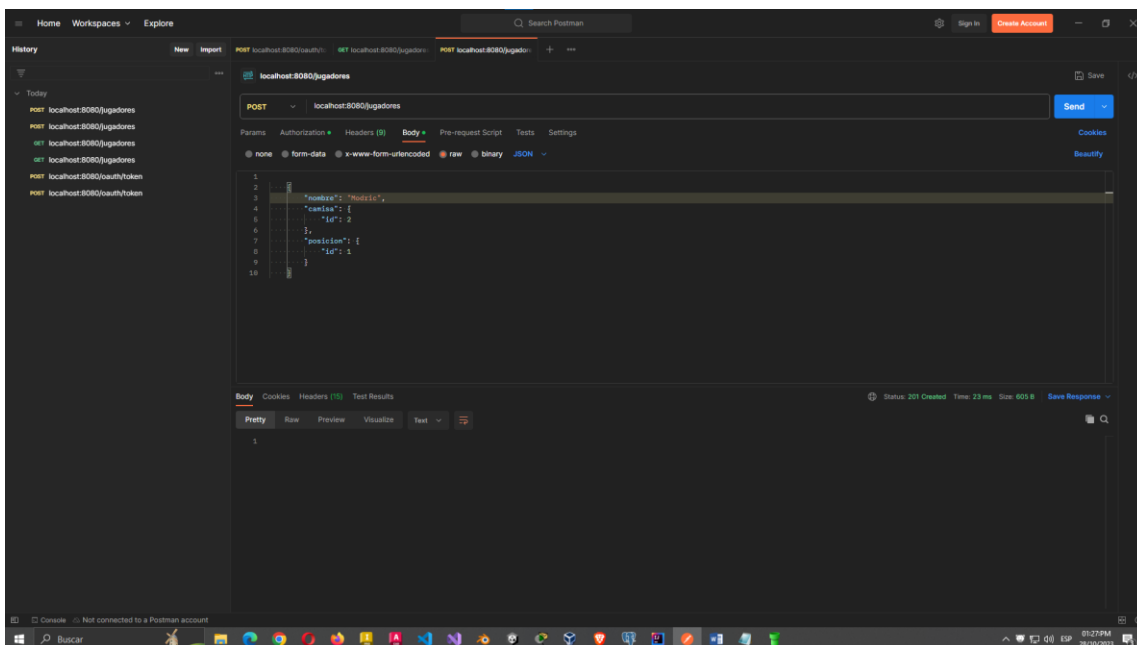
En el body digitamos las credenciales de la base de datos.



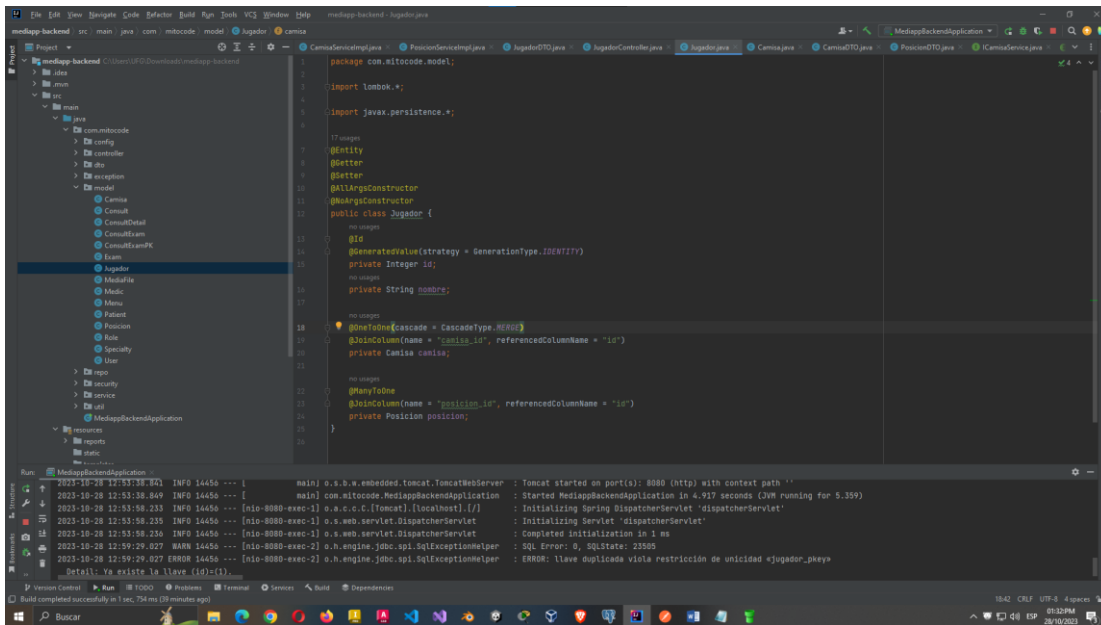
En el GET aparece generado el TOKEN y estructura que se trae de a base de datos. Nombre, camisa y posición.



En el POST en pestaña BODY creamos a los jugadores, en mi caso Modric.



Detallamos primeramente el Modelo que creamos:  
Modelo Jugador, su relación con camisa es Uno a Uno  
Y la relación con posición es Uno a Muchos



The screenshot shows an IDE with the following code in the `Jugador.java` file:

```
package com.mitocode.model;

import lombok.*;
import javax.persistence.*;

@Entity
@Table
@Getter
@Setter
@AllArgsConstructor
@EqualsAndHashCode
public class Jugador {

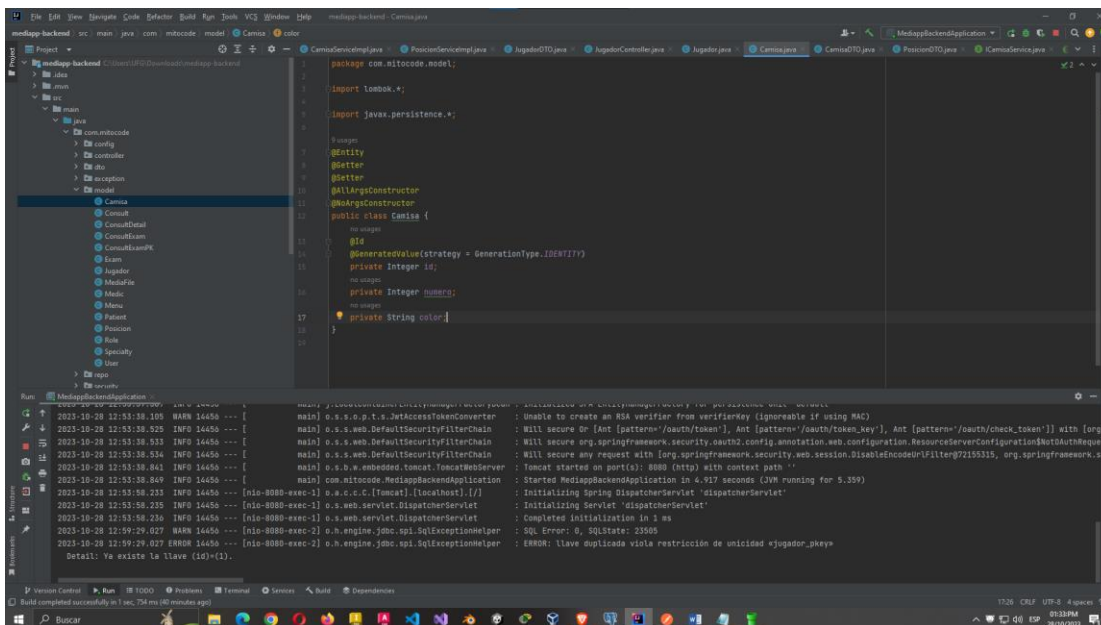
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "camisa_id", referencedColumnName = "id")
    private Camisa camisa;

    @ManyToOne
    @JoinColumn(name = "posicion_id", referencedColumnName = "id")
    private Posicion posicion;
}
```

The logs at the bottom show the application starting on port 8080. A warning is displayed: `ERROR: Llave duplicada viola restricción de unicidad «jugador_keys»` with details: `Detalle: Ya existe la llave (id)=(1).`

## Modelo Camisa



The screenshot shows an IDE with the following code in the `Camisa.java` file:

```
package com.mitocode.model;

import lombok.*;
import javax.persistence.*;

@Entity
@Table
@Getter
@Setter
@AllArgsConstructor
@EqualsAndHashCode
public class Camisa {

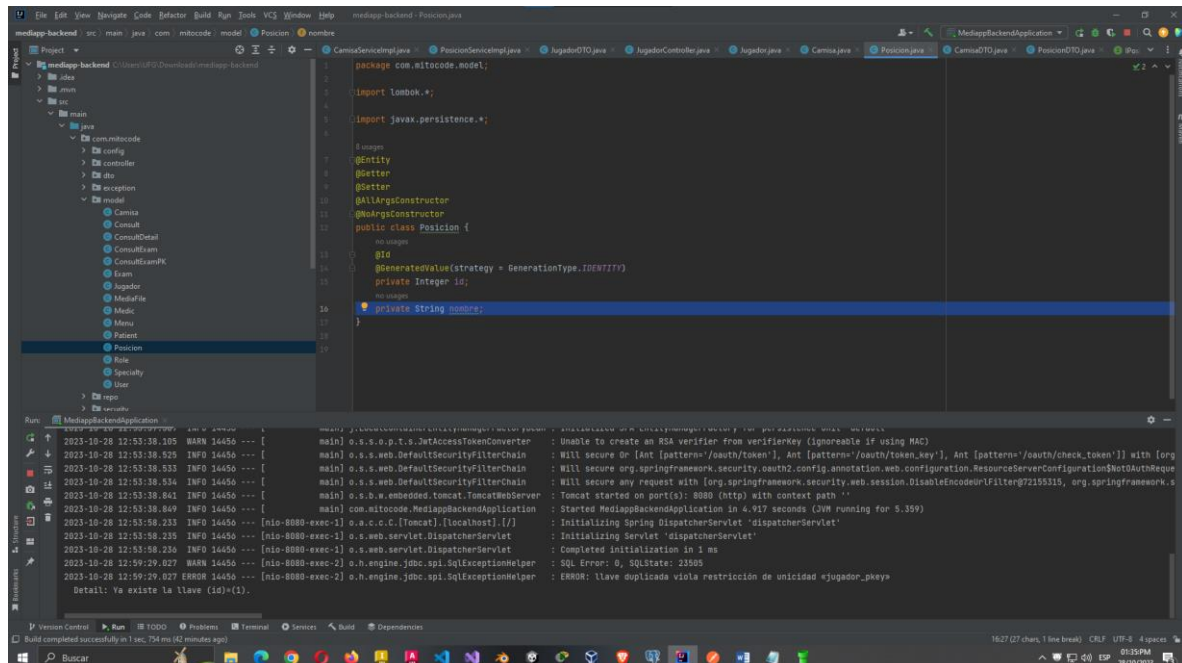
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private Integer numero;

    private String color;
}
```

The logs at the bottom show the application starting on port 8080. A warning is displayed: `ERROR: Llave duplicada viola restricción de unicidad «jugador_keys»` with details: `Detalle: Ya existe la llave (id)=(1).`

# Modelo Posición



```
1 package com.mitocode.model;
2
3 import lombok.*;
4
5 import javax.persistence.*;
6
7 @Entity
8 @Getter
9 @Setter
10 @AllArgsConstructor
11 @NoArgsConstructor
12 public class Posicion {
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Integer id;
15
16     private String nombre;
17 }
18
```

Run: MedappBackendApplication

2023-10-28 12:55:38.105 WARN 14656 --- [main] o.s.s.o.p.t.s.JwtAccessTokenConverter : Unable to create an RSA verifier from verifierKey (ignoreable if using MAC)

2023-10-28 12:55:38.525 INFO 14656 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure Url [ant [pattern='/oauth/token'], Ant [pattern='/oauth/check\_token']] with [org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfiguration\$NotOAuthRequestValidator]

2023-10-28 12:55:38.533 INFO 14656 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.session.DisableEncodeurFilter\$72155315, org.springframework.security.web.session.DisableEncodeurFilter\$72155315]

2023-10-28 12:55:38.641 INFO 14656 --- [main] com.mitocode.MedappBackendApplication : Started MedappBackendApplication in 4.917 seconds (JVM running for 5.359)

2023-10-28 12:55:38.233 INFO 14656 --- [nio-8080-exec-1] o.s.c.o.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'

2023-10-28 12:55:38.235 INFO 14656 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'

2023-10-28 12:55:38.236 INFO 14656 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

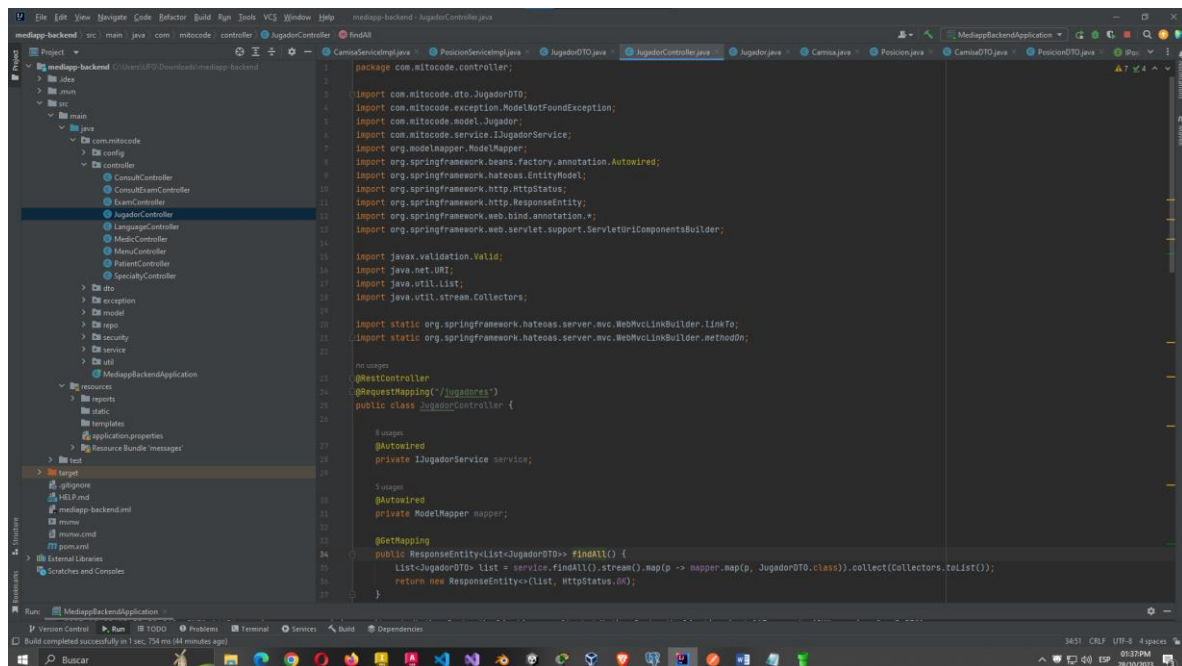
2023-10-28 12:59:29.027 WARN 14656 --- [nio-8080-exec-2] o.h.engine.jdbc.spi.SqlExceptionHelper : SQL Error: 0, SQLState: 23505

2023-10-28 12:59:29.027 ERROR 14656 --- [nio-8080-exec-2] o.h.engine.jdbc.spi.SqlExceptionHelper : ERROR: llave duplicada viola restricción de unicidad «jugador\_pkey»

Detail: Ya existe la llave (10)(1).

Luego el controller que se desglosa asi:

Primero las importaciones



```
1 package com.mitocode.controller;
2
3 import com.mitocode.dto.JugadorDTO;
4 import com.mitocode.exception.ModelNotFoundException;
5 import com.mitocode.model.Jugador;
6 import com.mitocode.service.IJugadorService;
7 import org.modelmapper.ModelMapper;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.hateoas.EntityModel;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.web.bind.annotation.*;
13 import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
14
15 import javax.validation.Valid;
16 import java.net.URI;
17 import java.util.List;
18 import java.util.stream.Collectors;
19
20 import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;
21 import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.methodOn;
22
23 @RestController
24 @RequestMapping("/jugadores")
25 public class JugadorController {
26
27     @Autowired
28     private IJugadorService service;
29
30     @Autowired
31     private ModelMapper mapper;
32
33     @GetMapping
34     public ResponseEntity<List<JugadorDTO>> findAll() {
35         List<JugadorDTO> list = service.findAll().stream().map(p -> mapper.map(p, JugadorDTO.class)).collect(Collectors.toList());
36         return new ResponseEntity<>(list, HttpStatus.OK);
37     }
38 }
39
```

Run: MedappBackendApplication

2023-10-28 12:55:38.105 WARN 14656 --- [main] o.s.s.o.p.t.s.JwtAccessTokenConverter : Unable to create an RSA verifier from verifierKey (ignoreable if using MAC)

2023-10-28 12:55:38.525 INFO 14656 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure Url [ant [pattern='/oauth/token'], Ant [pattern='/oauth/check\_token']] with [org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfiguration\$NotOAuthRequestValidator]

2023-10-28 12:55:38.533 INFO 14656 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.session.DisableEncodeurFilter\$72155315, org.springframework.security.web.session.DisableEncodeurFilter\$72155315]

2023-10-28 12:55:38.641 INFO 14656 --- [main] com.mitocode.MedappBackendApplication : Started MedappBackendApplication in 4.917 seconds (JVM running for 5.359)

2023-10-28 12:55:38.233 INFO 14656 --- [nio-8080-exec-1] o.s.c.o.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'

2023-10-28 12:55:38.235 INFO 14656 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'

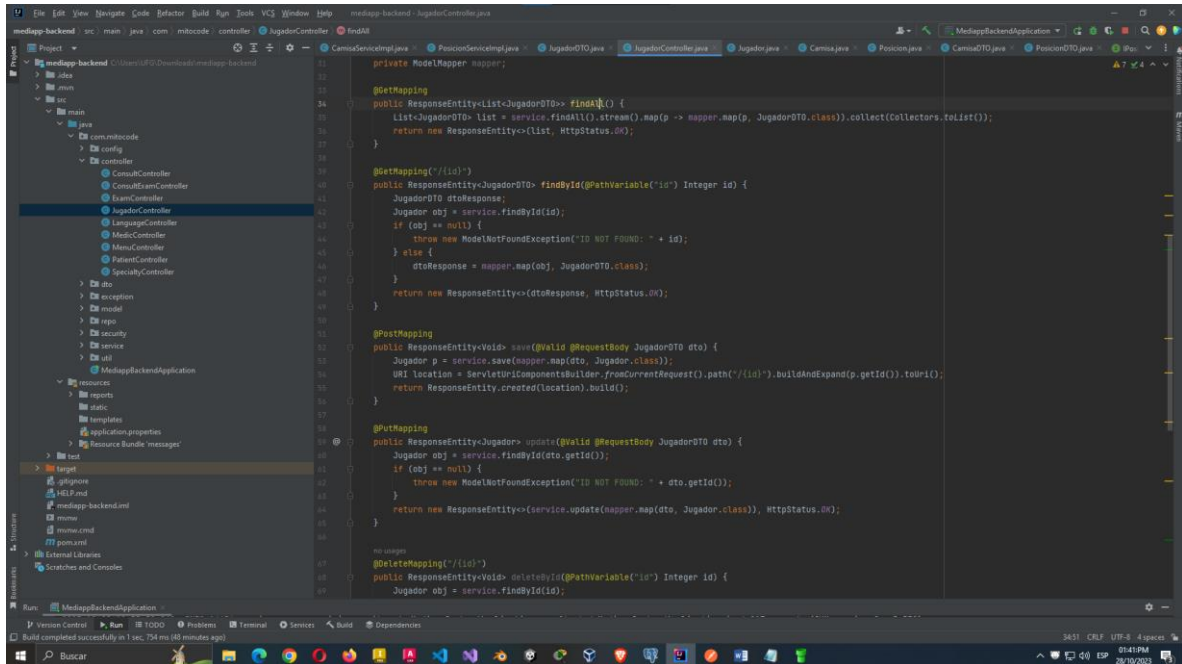
2023-10-28 12:55:38.236 INFO 14656 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

2023-10-28 12:59:29.027 WARN 14656 --- [nio-8080-exec-2] o.h.engine.jdbc.spi.SqlExceptionHelper : SQL Error: 0, SQLState: 23505

2023-10-28 12:59:29.027 ERROR 14656 --- [nio-8080-exec-2] o.h.engine.jdbc.spi.SqlExceptionHelper : ERROR: llave duplicada viola restricción de unicidad «jugador\_pkey»

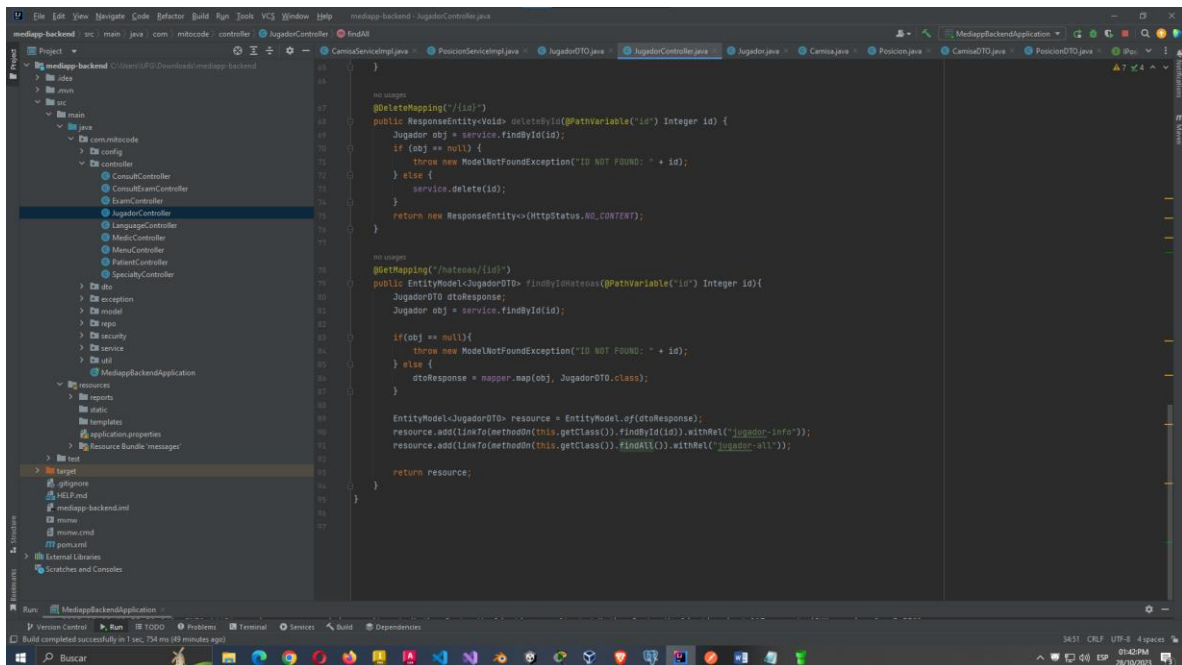
Detail: Ya existe la llave (10)(1).

# Tambien los get y post



```
11 private ModelMapper mapper;
12
13 @GetMapping
14 public ResponseEntity<List<JugadorDTO>> findAll() {
15     List<JugadorDTO> list = service.findAll().stream().map(p -> mapper.map(p, JugadorDTO.class)).collect(Collectors.toList());
16     return new ResponseEntity<>(list, HttpStatus.OK);
17 }
18
19 @GetMapping("/{id}")
20 public ResponseEntity<JugadorDTO> findById(@PathVariable("id") Integer id) {
21     JugadorDTO dtoResponse;
22     Jugador obj = service.findById(id);
23     if (obj == null) {
24         throw new ModelNotFoundException("ID NOT FOUND: " + id);
25     } else {
26         dtoResponse = mapper.map(obj, JugadorDTO.class);
27     }
28     return new ResponseEntity<>(dtoResponse, HttpStatus.OK);
29 }
30
31 @PostMapping
32 public ResponseEntity<Void> save(@Valid @RequestBody JugadorDTO dto) {
33     Jugador p = service.save(mapper.map(dto, Jugador.class));
34     URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(p.getId()).toUri();
35     return ResponseEntity.created(location).build();
36 }
37
38 @PutMapping
39 public ResponseEntity<Jugador> update(@Valid @RequestBody JugadorDTO dto) {
40     Jugador obj = service.findById(dto.getId());
41     if (obj == null) {
42         throw new ModelNotFoundException("ID NOT FOUND: " + dto.getId());
43     }
44     return new ResponseEntity<>(service.update(mapper.map(dto, Jugador.class)), HttpStatus.OK);
45 }
46
47 @DeleteMapping("/{id}")
48 public ResponseEntity<Void> deleteById(@PathVariable("id") Integer id) {
49     Jugador obj = service.findById(id);
50 }
```

The screenshot shows an IDE with a project named 'medapp-backend'. The file explorer on the left shows the project structure, including 'src/main/java/com/mibcode/controller' and 'MedappBackendApplication'. The main editor displays the 'JugadorController.java' file with the following code:



```
49 }
50
51 @DeleteMapping("/{id}")
52 public ResponseEntity<Void> deleteById(@PathVariable("id") Integer id) {
53     Jugador obj = service.findById(id);
54     if (obj == null) {
55         throw new ModelNotFoundException("ID NOT FOUND: " + id);
56     } else {
57         service.delete(id);
58     }
59     return new ResponseEntity<>(HttpStatus.NO_CONTENT);
60 }
61
62 @GetMapping("/{id}/resource/{id}")
63 public EntityModel<JugadorDTO> findByIdResource(@PathVariable("id") Integer id) {
64     JugadorDTO dtoResponse;
65     Jugador obj = service.findById(id);
66
67     if (obj == null) {
68         throw new ModelNotFoundException("ID NOT FOUND: " + id);
69     } else {
70         dtoResponse = mapper.map(obj, JugadorDTO.class);
71     }
72
73     EntityModel<JugadorDTO> resource = EntityModel.of(dtoResponse);
74     resource.add(linkTo(methodOn(this.getClass()).findById(id)).withRel("jugador-info"));
75     resource.add(linkTo(methodOn(this.getClass()).findAll()).withRel("jugador-all"));
76
77     return resource;
78 }
```

The screenshot shows the same IDE with the 'JugadorController.java' file. The code is now showing the 'deleteById' and 'findByIdResource' methods. The 'deleteById' method is implemented as follows: