



Nombre: Franklin Alfredo Castillo Cristino –
CC101020

Materia: Desarrollo Aplicaciones Web

Docente: Ing. Carlos Boris Martinez Calzadia

Actividad: Laboratorio 3

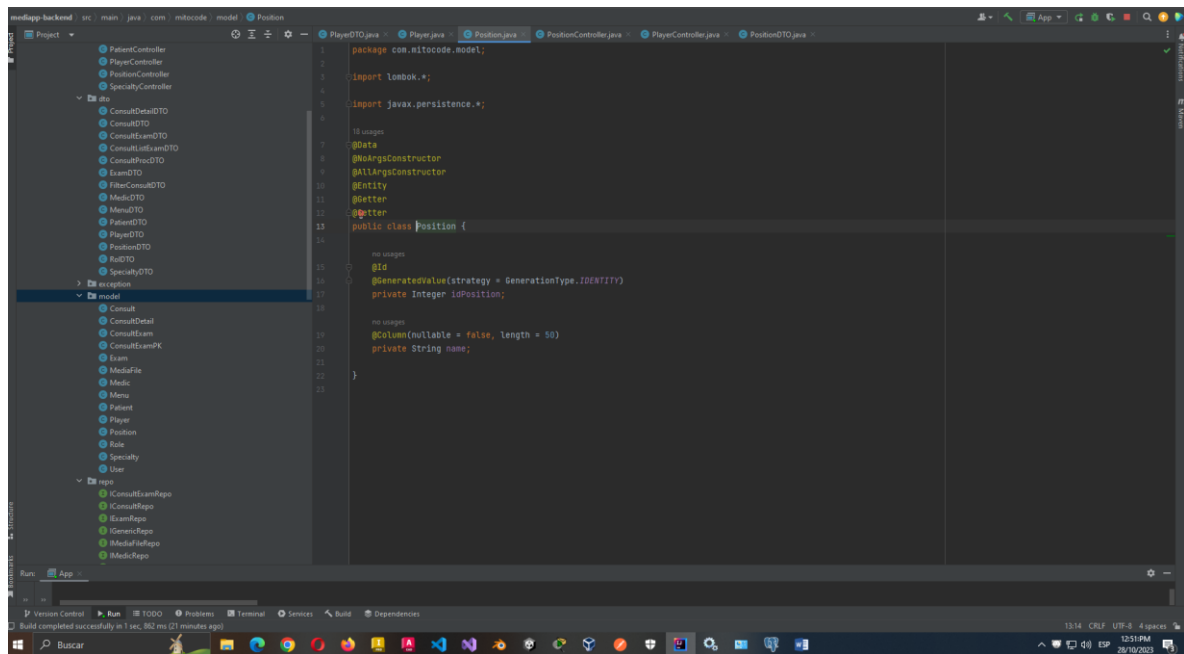
Fecha: 28 de octubre de 2023

Contenido

Modelos.....	3
Modelo Position	3
Modelo Player	3
Controllers:.....	4
Position Controller	4
Player Controller	5
DTOS.....	6
Position DTO:.....	6
Player DTO.....	6
Postman	8
OAuth2 POSTMAN	8
GET TOKEN	8
POSITION PLAYER POSTMAN	9
POST – Crear Posicion Jugador.....	9
GET – Posiciones Jugadores	9
PUT – Actualizar Posicion	10
DELETE – Eliminar Posicion	10
PLAYER POSTMAN	12
POST – Crear Jugador	12
GET – Obtener Jugadores.....	12
PUT – Actualizar Jugador.....	13
DELETE – Eliminar Jugador	13

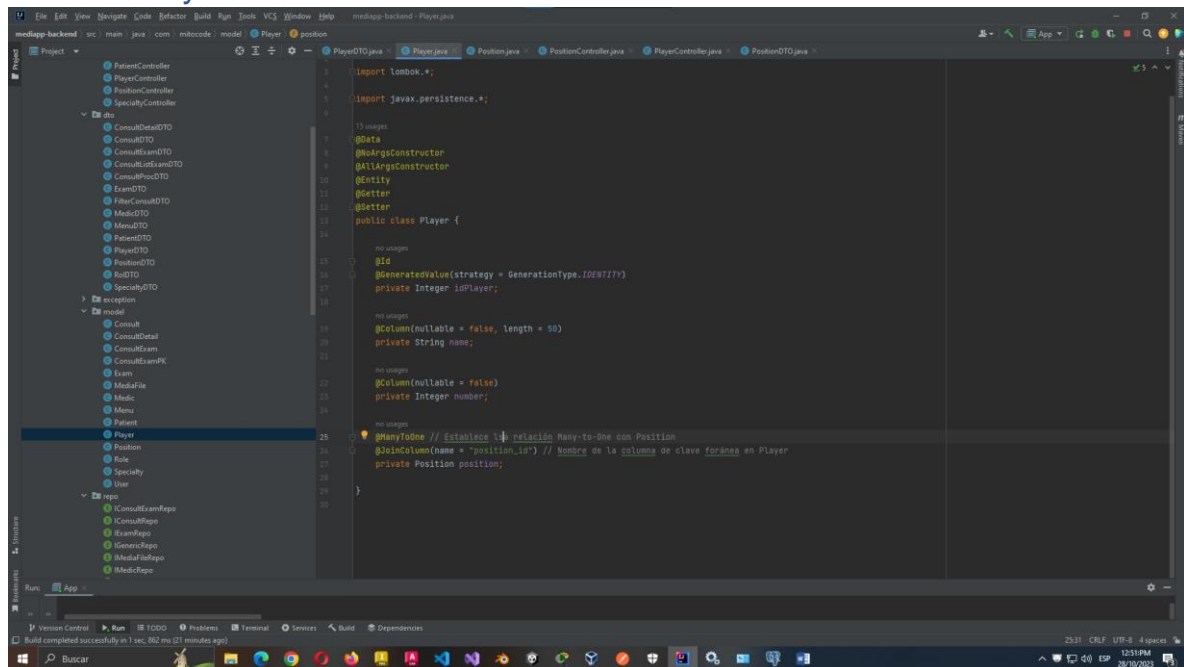
Modelos:

Modelo Position



```
1 package com.mitocode.model;
2
3 import lombok.*;
4
5 import javax.persistence.*;
6
7 @Data
8 @NoArgsConstructor
9 @AllArgsConstructor
10 @Entity
11 @Getter
12 @Setter
13 public class Position {
14
15     no usages
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Integer idPosition;
19
20     no usages
21     @Column(nullable = false, length = 50)
22     private String name;
23 }
```

Modelo Player



```
1 import lombok.*;
2
3 import javax.persistence.*;
4
5 @Data
6 @NoArgsConstructor
7 @AllArgsConstructor
8 @Entity
9 @Getter
10 @Setter
11 public class Player {
12
13     no usages
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Integer idPlayer;
17
18     no usages
19     @Column(nullable = false, length = 50)
20     private String name;
21
22     no usages
23     @Column(nullable = false)
24     private Integer number;
25
26     no usages
27     @ManyToOne // Establece la relación Many-to-One con Position
28     @JoinColumn(name = "position_id") // Nombre de la columna de clave foránea en Player
29     private Position position;
30 }
```

Position Controller:

```

1  package com.mtcode.controller;
2
3  import org.springframework.web.bind.annotation.*;
4  import org.springframework.web.servlet.mvc.annotation.annotation.AnnotationMethodProcessingMethodProcessor;
5  import org.springframework.web.servlet.mvc.annotation.annotation.AnnotationMethodProcessingMethodProcessor;
6
7  @RestController
8  @RequestMapping("/positions")
9  public class PositionController {
10
11     @Autowired
12     private IPositionService service;
13
14     @Autowired
15     private IMapper mapper;
16
17     @GetMapping
18     public ResponseEntity<List-PositionDTO> findAll() {
19         List-PositionDTO list = service.findAll().stream().map(p -> mapper.map(p, PositionDTO.class)).collect(Collectors.toList());
20         return new ResponseEntity<>(list, HttpStatus.OK);
21     }
22
23     @PostMapping
24     public ResponseEntity<void> save(@Valid @RequestBody PositionDTO dto) {
25         Position p = service.save(mapper.map(dto, Position.class));
26         //localhost:8080/positions/1
27         URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(p.getId()).toUri();
28         return ResponseEntity.created(location).build();
29     }
30
31     @PutMapping
32     public ResponseEntity-Position- update(@Valid @RequestBody PositionDTO dto) {
33         Position obj = service.findById(dto.getId());
34         if (obj == null) {
35             throw new ModelNotFoundException("ID NOT FOUND: " + dto.getId());
36         }
37         return new ResponseEntity<>(service.update(mapper.map(dto, Position.class)), HttpStatus.OK);
38     }
39
40     @DeleteMapping
41     public void delete(@Valid @RequestBody PositionDTO dto) {
42         service.delete(dto.getId());
43     }
44 }

```

The screenshot shows an IDE with the following components:

- Top Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Top Toolbar:** Run, Update, and other utility icons.
- Left Sidebar (Project Explorer):**
 - medapp-backend
 - src
 - main
 - java
 - com.mitocode
 - config
 - controller
 - ConsultController
 - ConsultExamController
 - ExamController
 - LanguageController
 - MedicController
 - MedicDTO
 - PatientController
 - PlayerController
 - PositionController
 - SpecialtyController

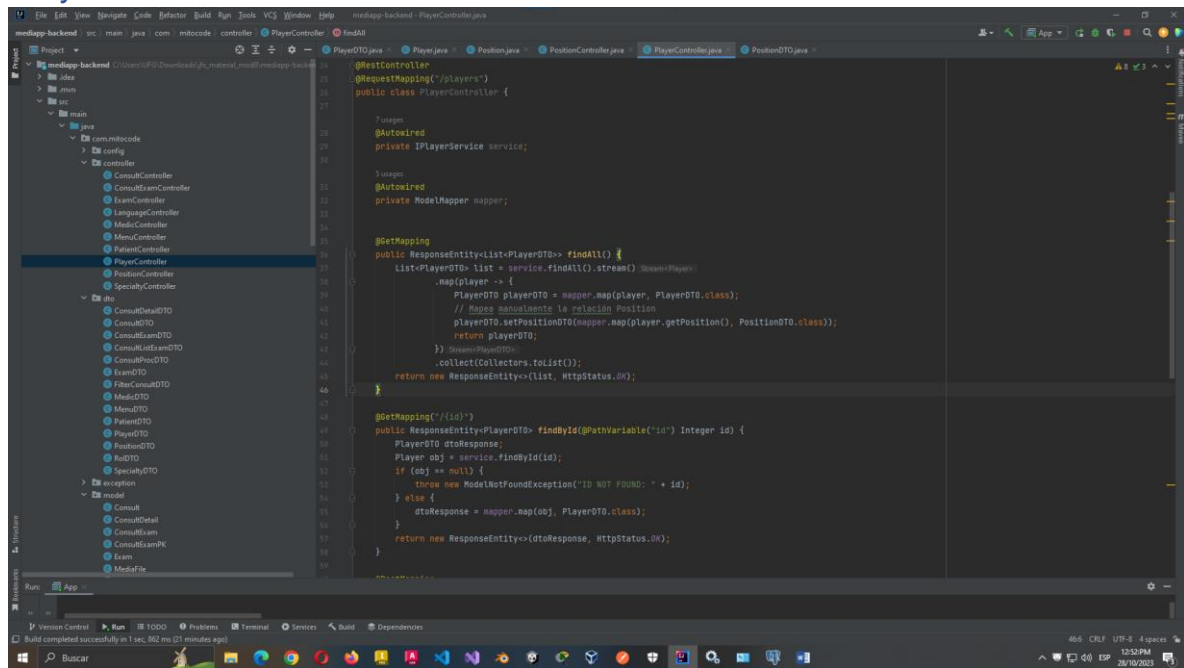
- Main Editor (PositionController.java):**

```

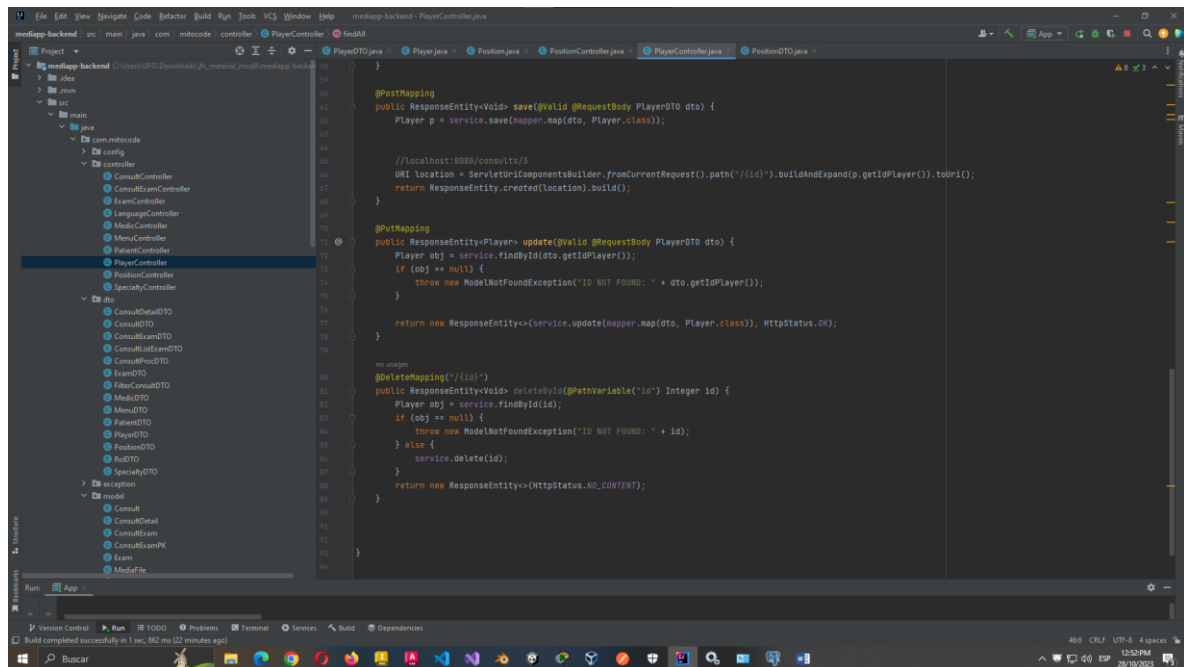
18
19
20 @PostMapping
21 public ResponseEntity<Void> save(@Valid @RequestBody PositionDTO dto) {
22     Position p = service.save(napper.map(dto, Position.class));
23     //location= http://ip:port/
24     URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(p.getIdPosition()).toUri();
25     return ResponseEntity.created(location).build();
26 }
27
28 @PutMapping
29 public ResponseEntity<Position> update(@Valid @RequestBody PositionDTO dto) {
30     Position obj = service.findById(dto.getIdPosition());
31     if (obj == null) {
32         throw new ModelNotFoundException("ID NOT FOUND: " + dto.getIdPosition());
33     }
34     return new ResponseEntity<>(service.update(napper.map(dto, Position.class)), HttpStatus.OK);
35 }
36
37 no usages
38 @DeleteMapping("/{id}")
39 public ResponseEntity<Void> deleteById(@PathVariable("id") Integer id) {
40     Position obj = service.findById(id);
41     if (obj == null) {
42         throw new ModelNotFoundException("ID NOT FOUND: " + id);
43     } else {
44         service.delete(id);
45     }
46     return new ResponseEntity<>(HttpStatus.NO_CONTENT);
47 }
48
49 }
50

```
- Bottom Status Bar:**
- Run: App
- Build completed successfully in 3s, 862 ms (1 minutes ago)
- 556 CRFP, UTF-8, 4 spaces
- 12:51 PM 10/20/2023

Player Controller



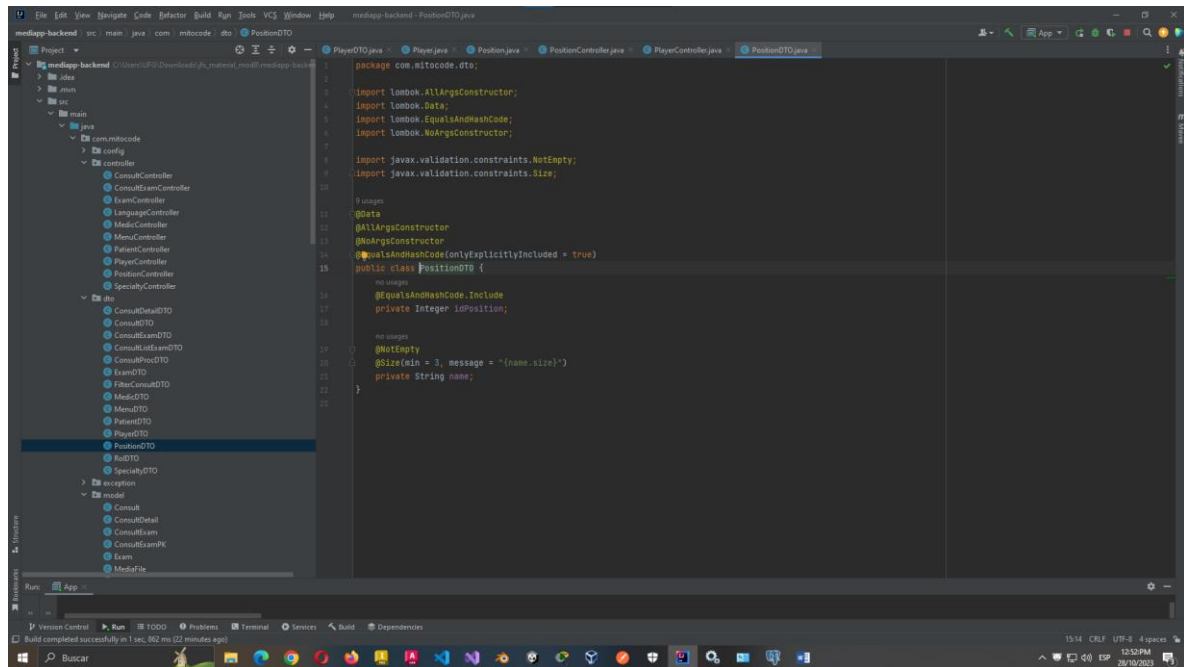
```
14  @RestController
15  @RequestMapping("/players")
16  public class PlayerController {
17
18      7 usages
19      @Autowired
20      private IPlayerService service;
21
22      3 usages
23      @Autowired
24      private IMapper mapper;
25
26      @GetMapping
27      public ResponseEntity<List<PlayerDTO>> findAll() {
28          List<PlayerDTO> list = service.findAll().stream().map(player -> {
29              PlayerDTO playerDTO = mapper.map(player, PlayerDTO.class);
30              // Mapeo manual de la relación Position
31              playerDTO.setPosition(mapper.map(player.getPosition(), PositionDTO.class));
32              return playerDTO;
33          }).collect(Collectors.toList());
34          return new ResponseEntity<>(list, HttpStatus.OK);
35      }
36
37      @GetMapping("/{id}")
38      public ResponseEntity<PlayerDTO> findById(@PathVariable("id") Integer id) {
39          PlayerDTO dtoResponse;
40          Player obj = service.findById(id);
41          if (obj == null) {
42              throw new ModelNotFoundException("ID NOT FOUND: " + id);
43          } else {
44              dtoResponse = mapper.map(obj, PlayerDTO.class);
45          }
46          return new ResponseEntity<>(dtoResponse, HttpStatus.OK);
47      }
48  }
```



```
50  }
51
52  @PostMapping
53  public ResponseEntity<Void> save(@Valid @RequestBody PlayerDTO dto) {
54      Player p = service.save(mapper.map(dto, Player.class));
55
56      // localhost:8080/commit/3
57      URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(p.getIdPlayer()).toUri();
58      return ResponseEntity.created(location).build();
59  }
60
61  @PutMapping
62  public ResponseEntity<Player> update(@Valid @RequestBody PlayerDTO dto) {
63      Player obj = service.findById(dto.getIdPlayer());
64      if (obj == null) {
65          throw new ModelNotFoundException("ID NOT FOUND: " + dto.getIdPlayer());
66      }
67      return new ResponseEntity<>(service.update(mapper.map(dto, Player.class)), HttpStatus.OK);
68  }
69
70  no usages
71  @DeleteMapping("/{id}")
72  public ResponseEntity<Void> deleteById(@PathVariable("id") Integer id) {
73      Player obj = service.findById(id);
74      if (obj == null) {
75          throw new ModelNotFoundException("ID NOT FOUND: " + id);
76      } else {
77          service.delete(id);
78      }
79      return new ResponseEntity<>(HttpStatus.NO_CONTENT);
80  }
81  }
```

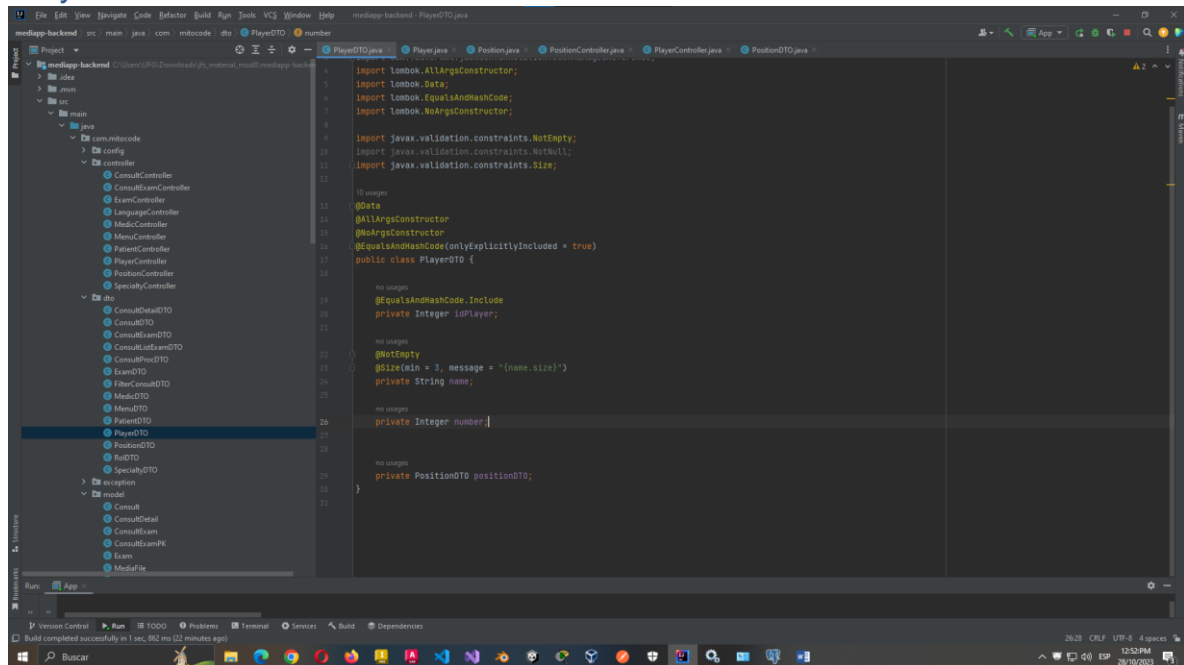
DTOS

Position DTO:



```
1 package com.mitocode.dto;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.EqualsAndHashCode;
6 import lombok.NoArgsConstructor;
7
8 import javax.validation.constraints.NotEmpty;
9 import javax.validation.constraints.Size;
10
11 @Data
12 @AllArgsConstructor
13 @NoArgsConstructor
14 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
15 public class PositionDTO {
16     // no usages
17     @EqualsAndHashCode.Include
18     private Integer idPosition;
19
20     // no usages
21     @NotEmpty
22     @Size(min = 3, message = "{name.size}")
23     private String name;
24 }
```

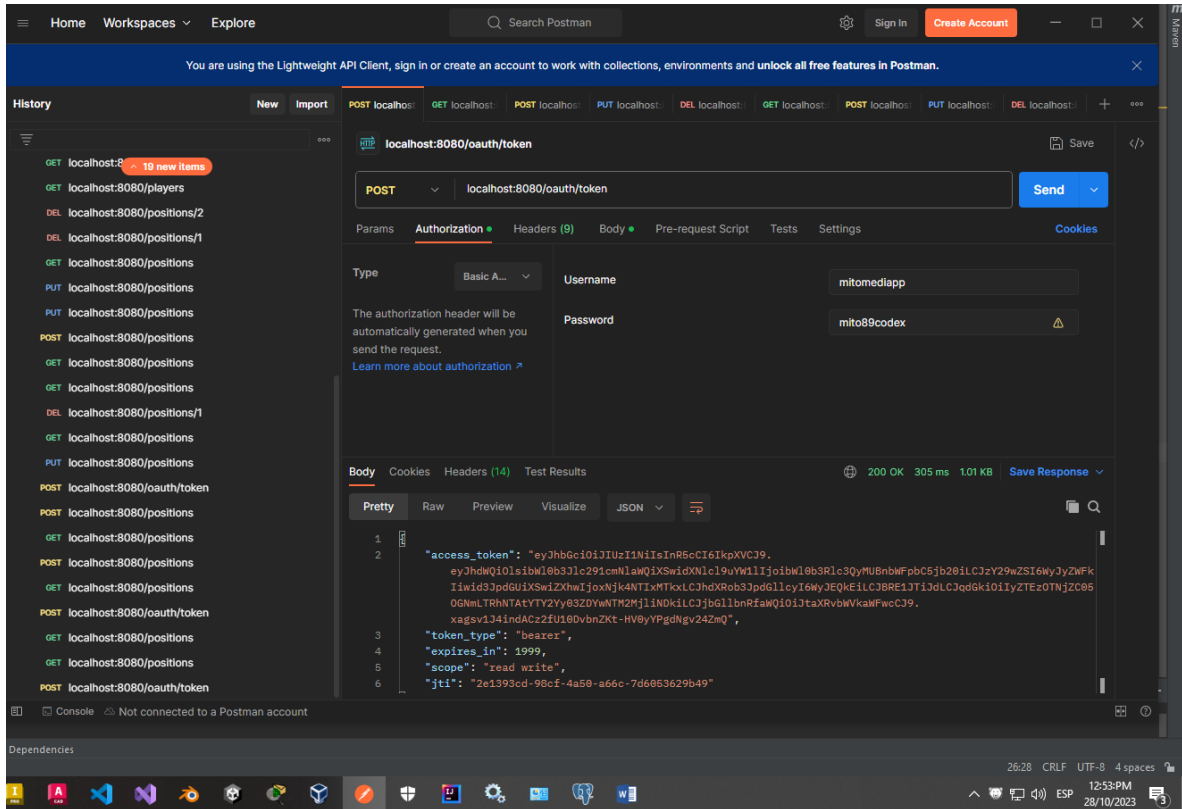
Player DTO



```
1 package com.mitocode.dto;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.EqualsAndHashCode;
6 import lombok.NoArgsConstructor;
7
8 import javax.validation.constraints.NotEmpty;
9 import javax.validation.constraints.NotNull;
10 import javax.validation.constraints.Size;
11
12 @Data
13 @AllArgsConstructor
14 @NoArgsConstructor
15 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
16 public class PlayerDTO {
17     // no usages
18     @EqualsAndHashCode.Include
19     private Integer idPlayer;
20
21     // no usages
22     @NotEmpty
23     @Size(min = 3, message = "{name.size}")
24     private String name;
25
26     // no usages
27     private Integer number;
28
29     // no usages
30     private PositionDTO position;
31 }
```

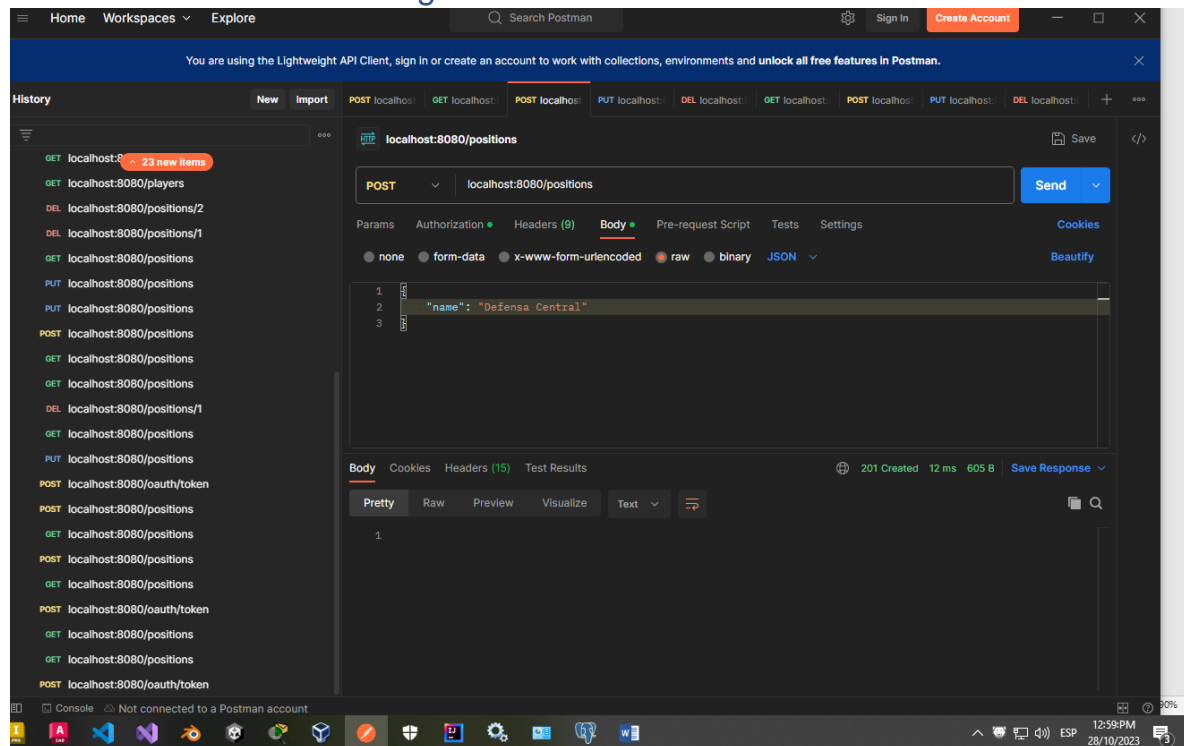
POSTMAN

OAUTH2 POSTMAN GET TOKEN

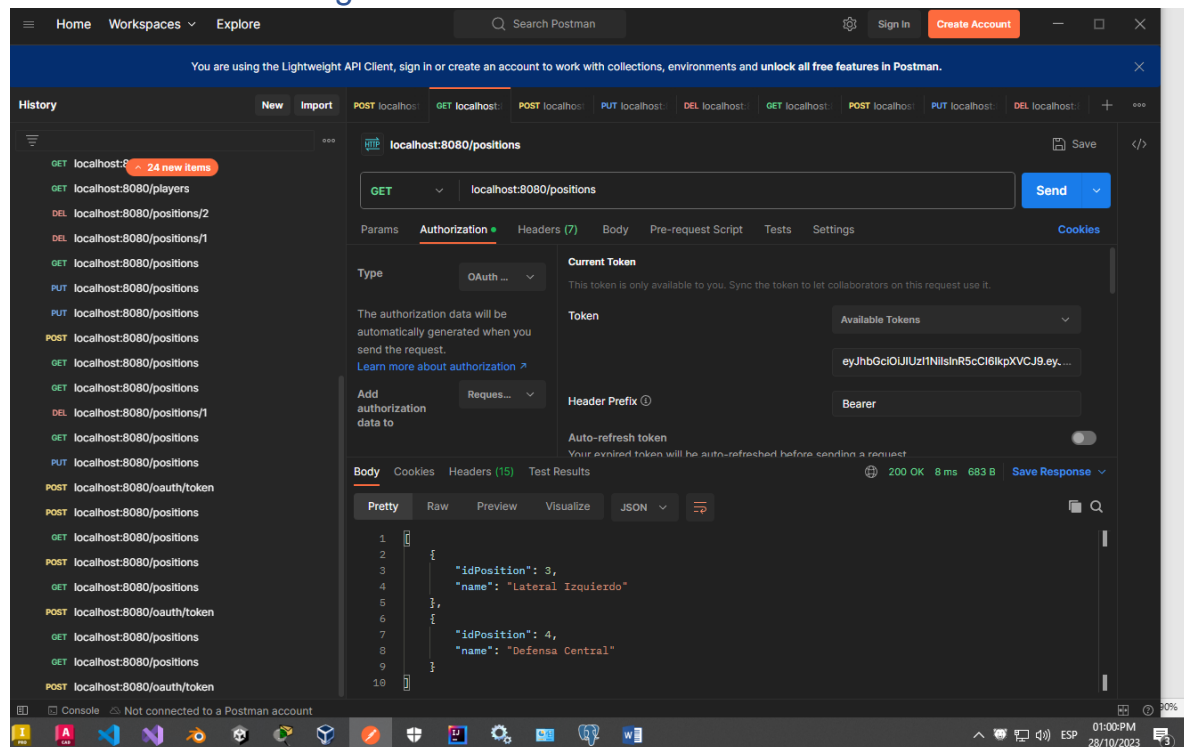


POSITION PLAYER POSTMAN

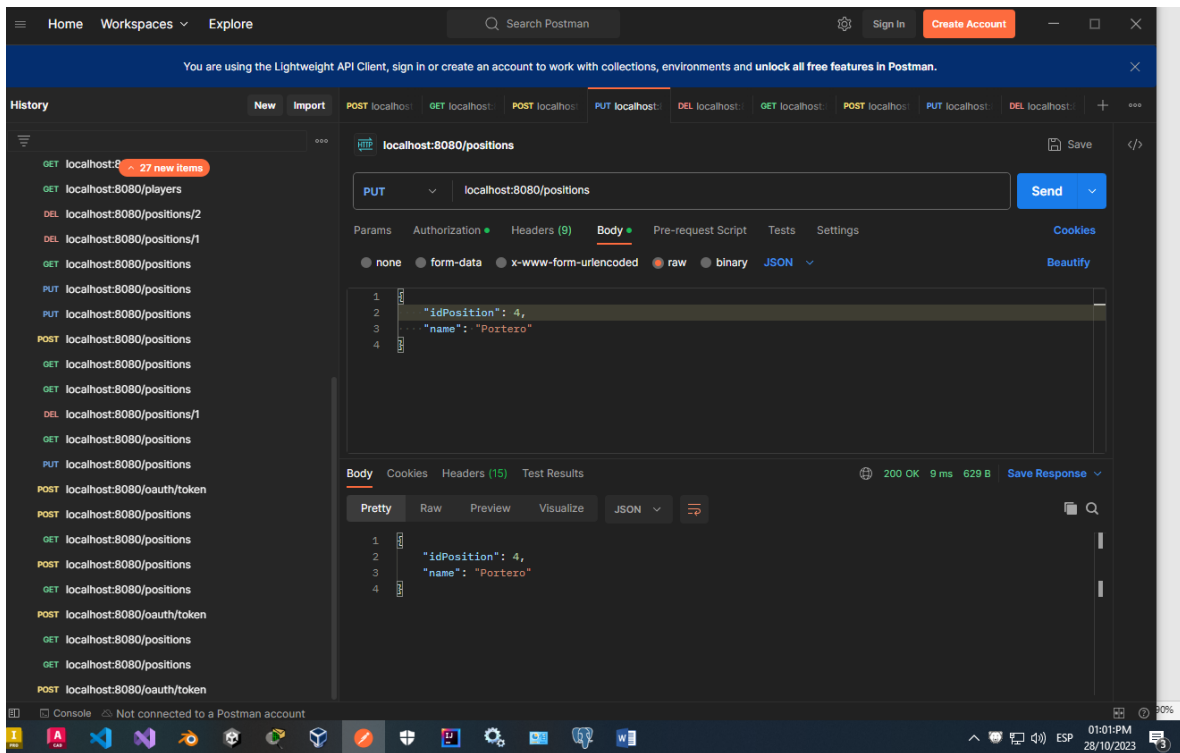
POST – Crear Posicion Jugador



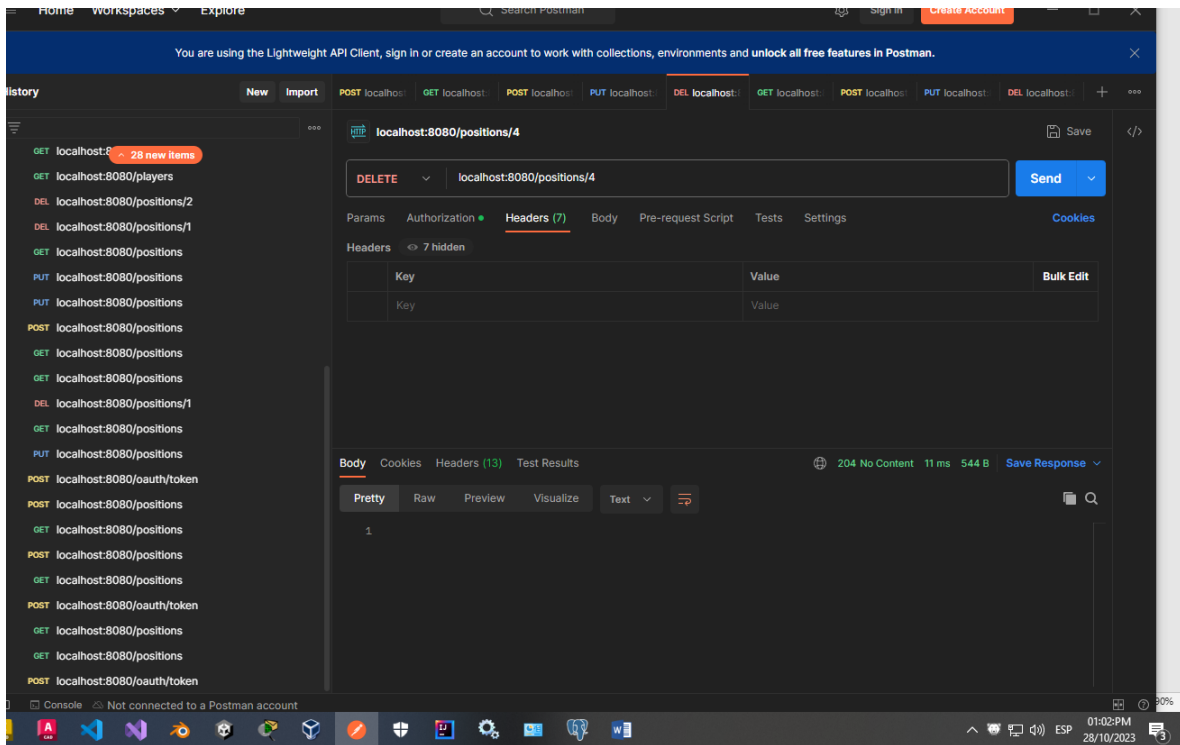
GET – Posiciones Jugadores



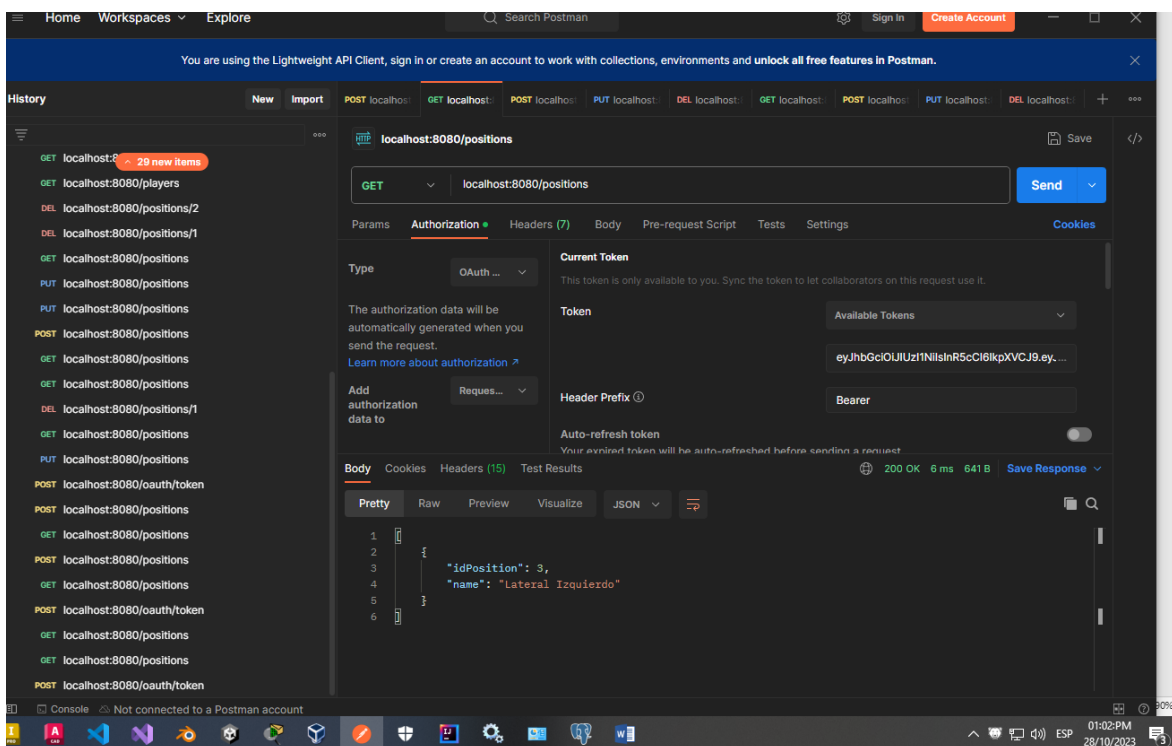
PUT – Actualizar Posicion



DELETE – Eliminar Posicion

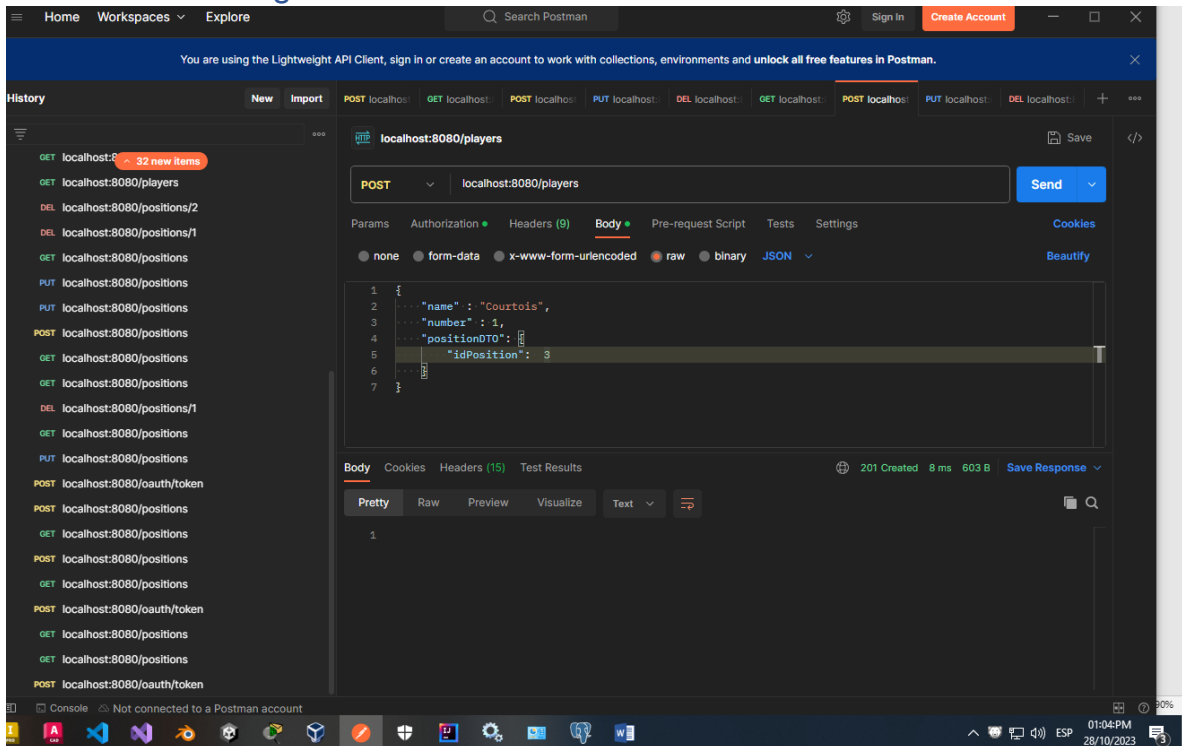


Revisamos que se borre correctamente:

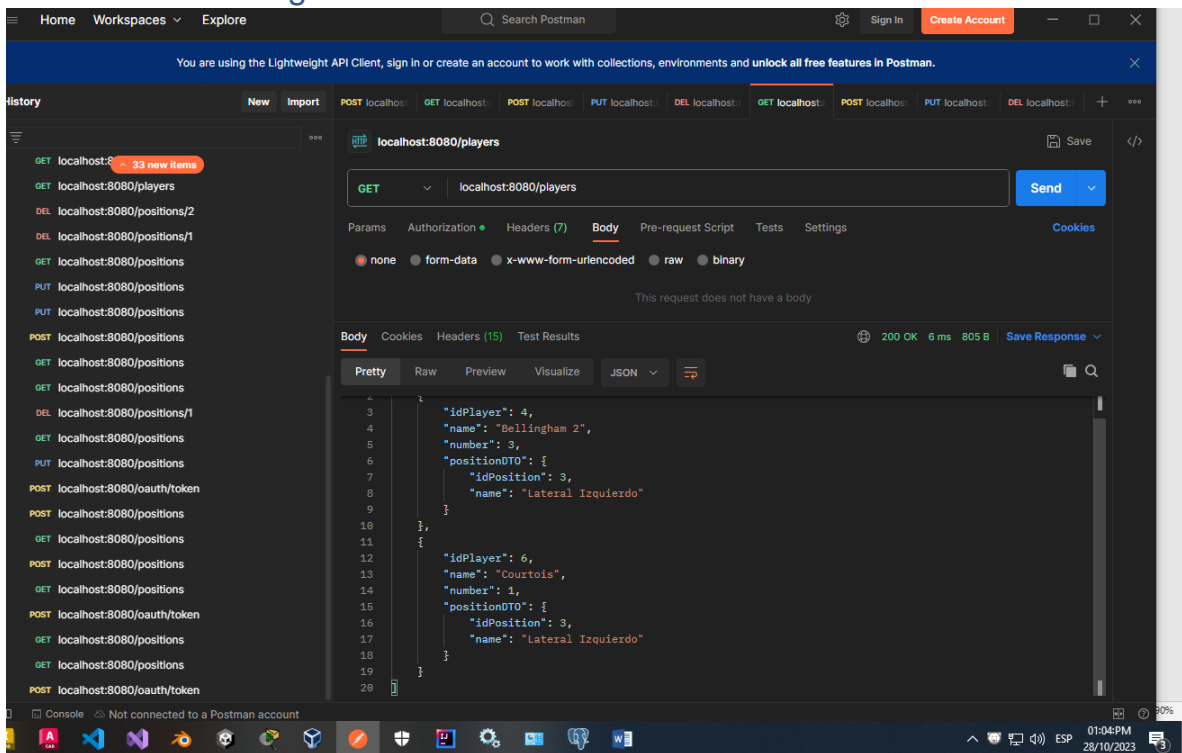


PLAYER POSTMAN

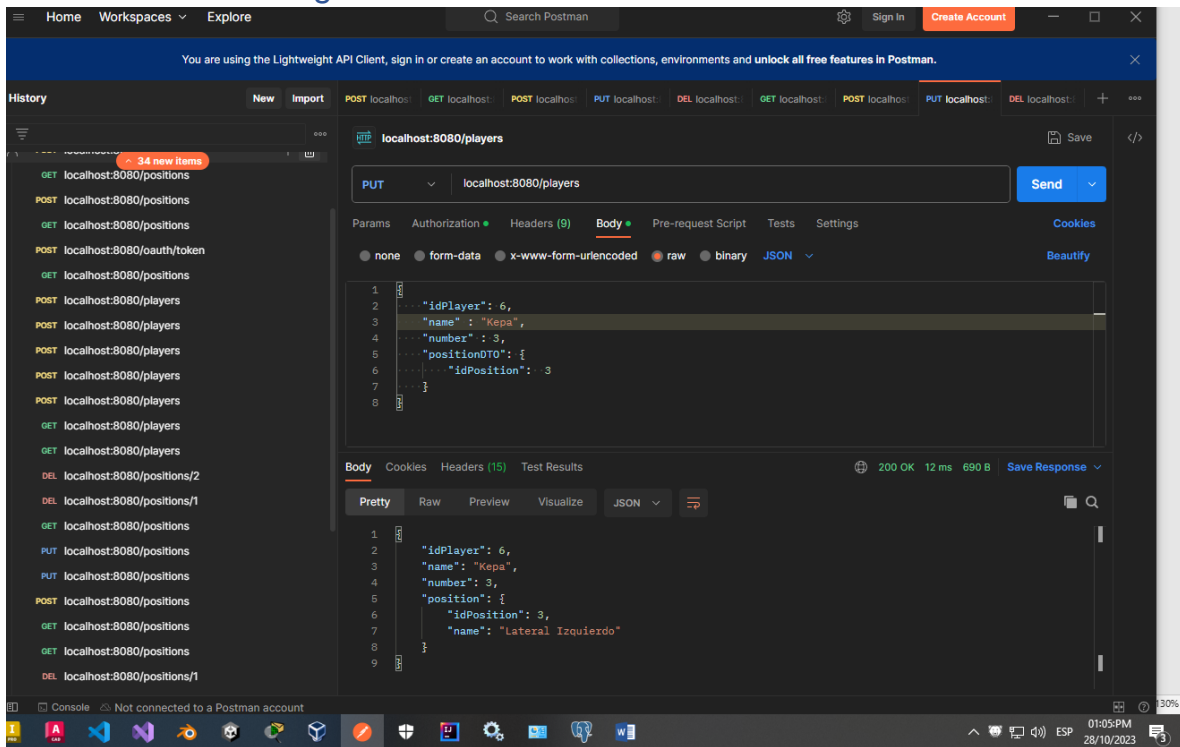
POST – Crear Jugador



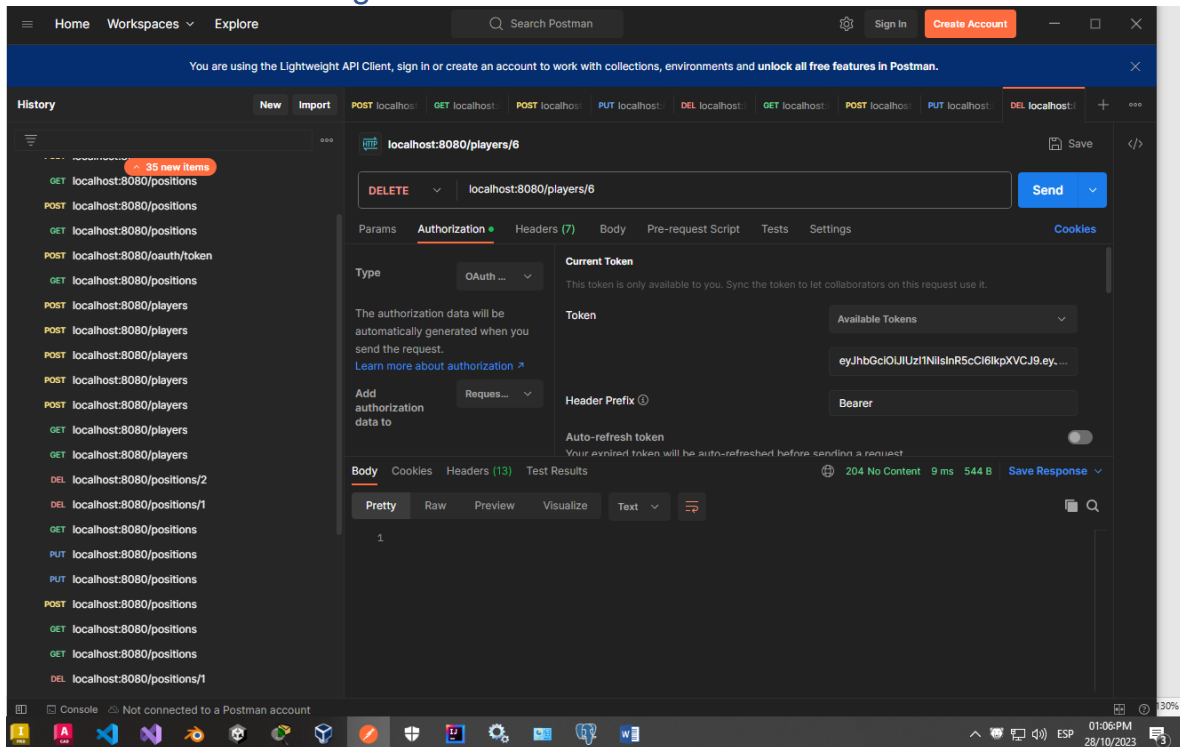
GET – Obtener Jugadores



PUT – Actualizar Jugador



DELETE – Eliminar Jugador



Revisamos que se haya borrado correctamente

