

Documentación API REST Seguro

Autor: Manuel Bouza García



Índice

Índice.....	2
Entidad Seguro.....	3
Seguro Service.....	4
Seguro Controller.....	6
Pruebas funcionales.....	7

Entidad Seguro

```
@Entity
@Table(name = "seguros")
data class Seguro(

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idSeguro")
    val idSeguro: Int = 0,

    @Column(name = "nif", nullable = false, length = 10)
    val nif: String,

    @Column(name = "nombre", nullable = false, length = 100)
    val nombre: String,

    @Column(name = "ape1", nullable = false, length = 100)
    val ape1: String,

    @Column(name = "ape2", length = 100)
    val ape2: String? = null,

    @Column(name = "edad", nullable = false)
    @Min(1)
    val edad: Int,

    @Column(name = "num_hijos", nullable = false)
    @Min(0)
    val numHijos: Int,

    @Column(name = "fecha_creacion", nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    val fechaCreacion: Date,

    @Column(name = "sexo", nullable = false, length = 10)
    val sexo: String,

    @Column(name = "casado", nullable = false)
    val casado: Boolean,

    @Column(name = "embarazada", nullable = false)
    val embarazada: Boolean
)
```

Esta clase define una tabla SQL que representa un seguro. A continuación se definen las columnas.

idSeguro

- **@Id**: Indica que este atributo es la clave primaria de la entidad.
- **@GeneratedValue(strategy = GenerationType.IDENTITY)**: Configura la generación automática del valor del ID. La estrategia IDENTITY permite que la base de datos genere valores únicos para este campo.
- **@Column(name = "idSeguro")**: Especifica que este atributo se mapea a la columna idSeguro en la tabla.

- **val idSeguro: Int = 0:** El valor predeterminado es 0, lo que indica que es un ID inicial antes de ser persistido.

nif

- **@Column(name = "nif", nullable = false, length = 10):**
 - name = "nif": Define el nombre de la columna como nif.
 - nullable = false: La columna no puede contener valores nulos.
 - length = 10: Define un límite de longitud de 10 caracteres.
- **val nif: String:** Es un identificador obligatorio.

nombre

- **@Column(name = "nombre", nullable = false, length = 100):** Similar al campo anterior, define restricciones:
 - Nombre de columna: nombre.
 - No permite valores nulos.
 - Longitud máxima: 100 caracteres.
- **val nombre: String:** Es un campo obligatorio.

ape1 y ape2

- **@Column(name = "ape1", nullable = false, length = 100):**
 - Nombre de columna: ape1.
 - Es obligatorio y tiene una longitud máxima de 100 caracteres.
- **@Column(name = "ape2", length = 100):**
 - Nombre de columna: ape2.
 - Este campo es opcional (por defecto nullable = true) y puede contener un valor nulo, representado en Kotlin como String?.
- **val ape1: String y val ape2: String? = null:**
 - ape1 es obligatorio.
 - ape2 es opcional con un valor predeterminado null.

edad

- **@Column(name = "edad", nullable = false):**
 - La columna edad no permite valores nulos.
- **@Min(1):** Define una restricción de validación para que la edad mínima sea 1.
- **val edad: Int:** Es un campo obligatorio.

numHijos

- **@Column(name = "num_hijos", nullable = false):**
 - La columna num_hijos no permite valores nulos.
- **@Min(0):** Define una restricción de validación para que el número mínimo de hijos sea 0.
- **val numHijos: Int:** Es obligatorio.

fechaCreacion

- **@Column(name = "fecha_creacion", nullable = false):**
 - La columna fecha_creacion no permite valores nulos.
- **@Temporal(TemporalType.TIMESTAMP):** Indica que el tipo de dato en la base de datos será un *timestamp* (marca de tiempo) que incluye fecha y hora.
- **val fechaCreacion: Date:** Es obligatorio y representa un instante temporal.

sexo

- **@Column(name = "sexo", nullable = false, length = 10):**
 - Nombre de columna: sexo.
 - Es obligatorio.
 - Longitud máxima de 10 caracteres.
- **val sexo: String:** Define el género de la persona.

casado

- **@Column(name = "casado", nullable = false):**
 - Nombre de columna: casado.
 - Es obligatorio.
- **val casado: Boolean:** Representa el estado civil como un valor booleano (verdadero o falso).

embarazada

- **@Column(name = "embarazada", nullable = false):**
 - Nombre de columna: embarazada.
 - Es obligatorio.
- **val embarazada: Boolean:** Indica si la persona está embarazada.

Seguro Service

```
@Service
class SeguroService() {

    @Autowired
    private lateinit var seguroRepository: SeguroRepository

    fun getById(id: String): Seguro? {
        val idL = id.toLongOrNull() ?: throw ResponseStatusException(
            HttpStatus.BAD_REQUEST, "ID inválido: $id no es un número válido"
        )

        return seguroRepository.findByIdOrNull(idL)
            ?: throw ResponseStatusException(HttpStatus.NOT_FOUND, "Seguro no
encontrado con ID: $idL")
    }

    fun crearSeguro(seguro: Seguro): Seguro? {

        validarSeguro(seguro)

        return if (seguroRepository.findByIdOrNull(seguro.idSeguro.toLong()) ==
null) {
            seguroRepository.save(seguro)
        } else {
            null
        }
    }

    fun actualizarSeguro(id: String, seguro: Seguro): Seguro? {

        validarSeguro(seguro)

        val idL = id.toLongOrNull() ?: throw ResponseStatusException(
            HttpStatus.BAD_REQUEST, "ID inválido: $id no es un número válido"
        )

        if (seguro.idSeguro.toLong() != idL) return null

        return if (seguroRepository.findByIdOrNull(seguro.idSeguro.toLong()) !=
null) {
            seguroRepository.save(seguro)
        } else {
            throw ResponseStatusException(HttpStatus.NOT_FOUND, "Seguro no
encontrado con ID: $idL")
        }
    }

    fun obtenerTodosLosSeguros(): List<Seguro> {
        return seguroRepository.findAll()
    }
}
```

```

fun eliminarSeguro(id: Long): ResponseEntity<Unit> {
    if (!seguroRepository.existsById(id)) {
        throw ResponseStatusException(HttpStatus.NOT_FOUND, "Seguro no encontrado")
    }
    seguroRepository.deleteById(id)
    return ResponseEntity.noContent().build()
}

fun validarSeguro(seguro: Seguro) {
    if (!DniValidator.esDniValido(seguro.nif)) throw
ResponseStatusException(HttpStatus.BAD_REQUEST, "El DNI introducido no es válido")
    if (seguro.edad in 0..17) throw
ResponseStatusException(HttpStatus.BAD_REQUEST, "No es posible ser menor de edad para hacer un seguro")
    if (seguro.numHijos < 0) throw
ResponseStatusException(HttpStatus.BAD_REQUEST, "El número de hijos no puede ser menor que 0")
    if (!seguro.casado) {
        if (seguro.numHijos > 0) throw
ResponseStatusException(HttpStatus.BAD_REQUEST, "Si el estado civil es soltero, el número de hijos debe ser 0")
    }
    if (seguro.embarazada) {
        if (seguro.sexo != "Mujer") throw
ResponseStatusException(HttpStatus.BAD_REQUEST, "Si está embarazada, el sexo debe ser Mujer")
    }
}
}

```

SeguroService representa un servicio cuyo objetivo es encapsular la lógica de negocio relacionada con la entidad Seguro y ser un intermediario entre el controlador y el repositorio. Esta clase se encargará de asegurarse de que las peticiones son válidas y devolver un error HTTP o la información pedida dependiendo de la solicitud recibida.

Seguro Controller

```
@RestController
@RequestMapping("/seguros")
class SeguroController() {

    @Autowired
    private lateinit var seguroService: SeguroService

    @GetMapping("/{id}")
    fun getById(
        @PathVariable id: String?
    ): Seguro?{

        if (id.isNullOrEmpty()){
            return null
        }

        return seguroService.getById(id)
    }

    @PostMapping
    fun crearSeguro(@RequestBody seguro: Seguro): ResponseEntity<Seguro> {
        return ResponseEntity.ok(seguroService.crearSeguro(seguro))
    }

    @PutMapping("/{id}")
    fun actualizarSeguro(@PathVariable id: String, @RequestBody seguro: Seguro):
    ResponseEntity<Seguro> {
        return ResponseEntity.ok(seguroService.actualizarSeguro(id, seguro))
    }

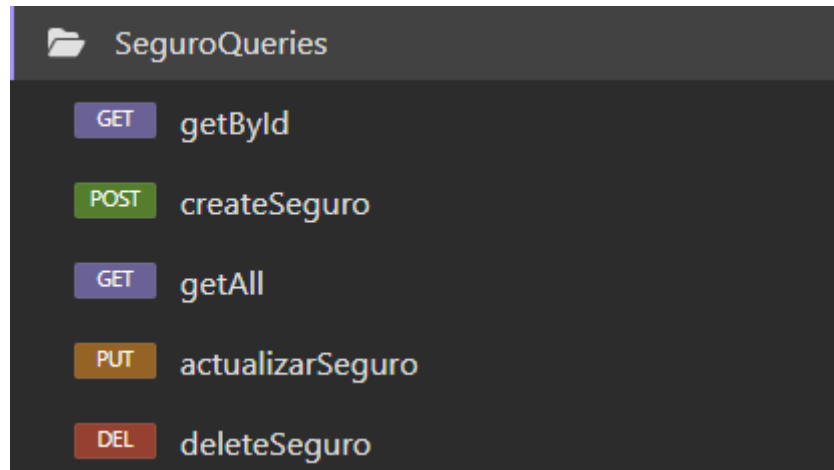
    @GetMapping
    fun obtenerTodosLosSeguros(): ResponseEntity<List<Seguro>> {
        return ResponseEntity.ok(seguroService.obtenerTodosLosSeguros())
    }

    @DeleteMapping("/{id}")
    fun eliminarSeguro(@PathVariable id: Long): ResponseEntity<Unit> {
        seguroService.eliminarSeguro(id)
        return ResponseEntity.noContent().build()
    }
}
```

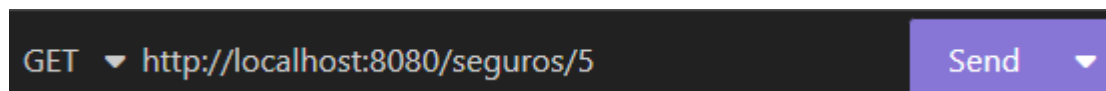
SeguroController tiene como objetivo exponer las funcionalidades relacionadas con la entidad Seguro a través de una API web, actuando como el punto de entrada para las solicitudes HTTP mediante los métodos GET, POST, PUT y DELETE.

Pruebas funcionales

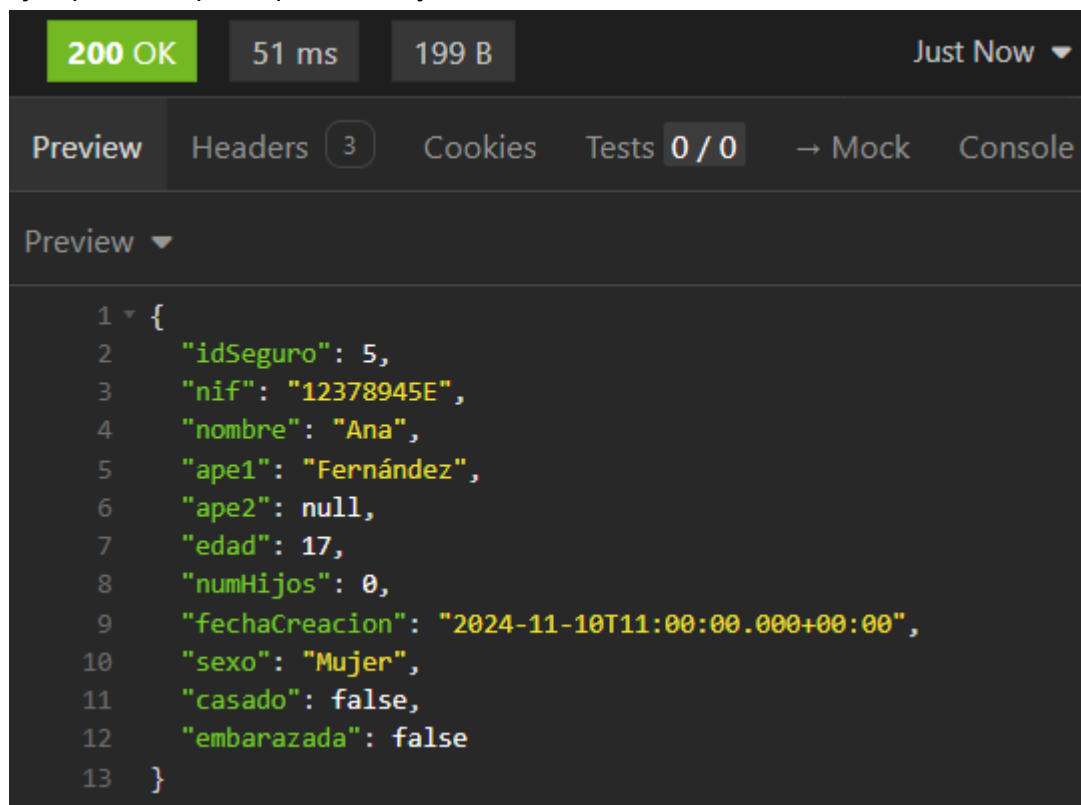
Para la realización de las pruebas funcionales se ha usado Insomnia y se han creado múltiples HTTP Requests para probar la funcionalidad correcta de nuestra API.



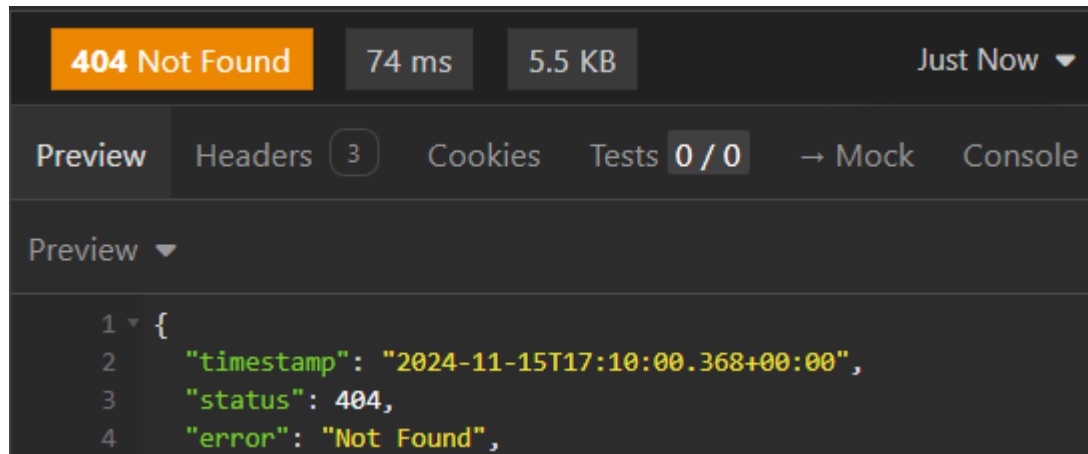
Request getByld



Ejemplo de request que se ha ejecutado correctamente



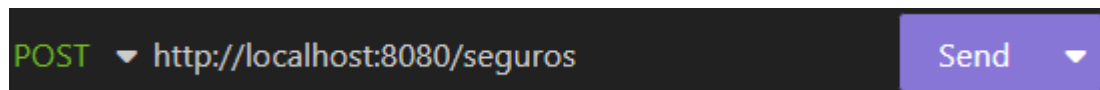
Ejemplo de request que no ha encontrado ningún seguro con la Id introducida



The screenshot shows a REST client interface with a status bar at the top indicating a **404 Not Found** response, a response time of **74 ms**, and a size of **5.5 KB**. Below the status bar, there are tabs for **Preview**, **Headers** (3), **Cookies**, **Tests** (0 / 0), **→ Mock**, and **Console**. The **Preview** tab is selected, showing a JSON response:

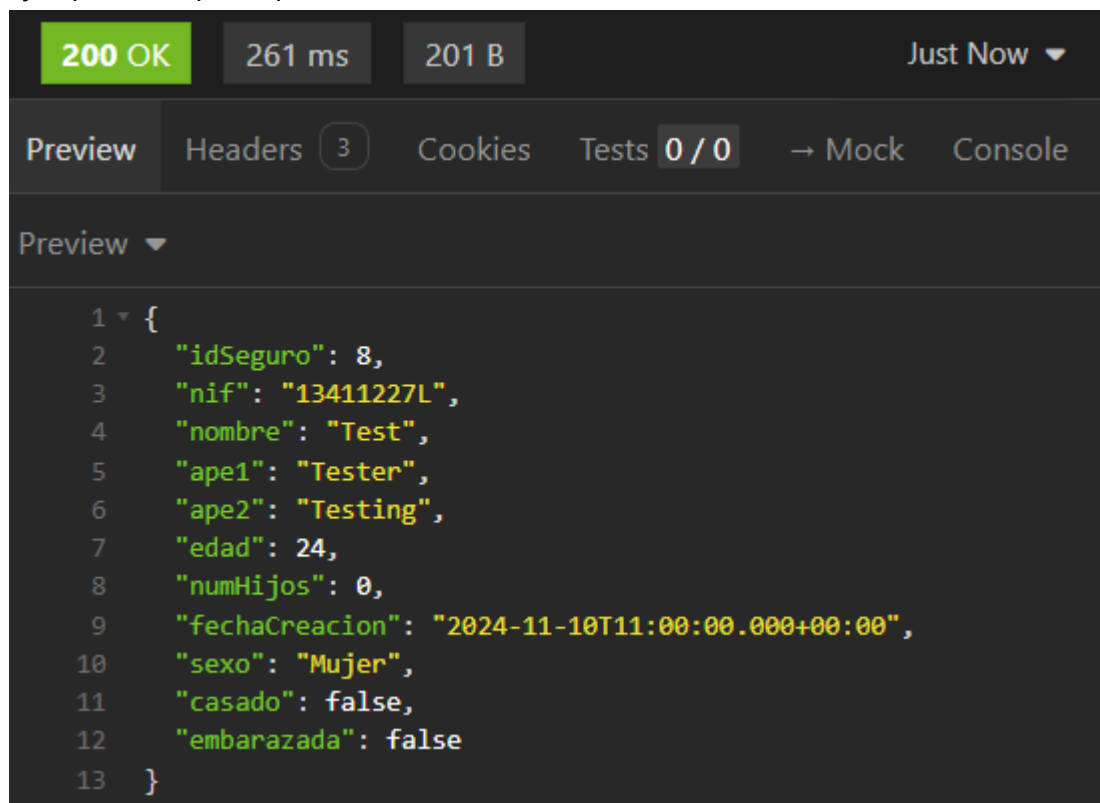
```
1 {  
2   "timestamp": "2024-11-15T17:10:00.368+00:00",  
3   "status": 404,  
4   "error": "Not Found",  
}
```

Request createSeguro



The screenshot shows a REST client interface with a status bar at the top indicating a **POST** request to **http://localhost:8080/seguros**. There is a **Send** button on the right.

Ejemplo de request que ha insertado correctamente



The screenshot shows a REST client interface with a status bar at the top indicating a **200 OK** response, a response time of **261 ms**, and a size of **201 B**. Below the status bar, there are tabs for **Preview**, **Headers** (3), **Cookies**, **Tests** (0 / 0), **→ Mock**, and **Console**. The **Preview** tab is selected, showing a JSON response:

```
1 {  
2   "idSeguro": 8,  
3   "nif": "13411227L",  
4   "nombre": "Test",  
5   "ape1": "Tester",  
6   "ape2": "Testing",  
7   "edad": 24,  
8   "numHijos": 0,  
9   "fechaCreacion": "2024-11-10T11:00:00.000+00:00",  
10  "sexo": "Mujer",  
11  "casado": false,  
12  "embarazada": false,  
13 }
```

Ejemplo de request que no ha insertado porque ya existía un seguro con la Id del el seguro introducido.

POST

http://localhost:8080/seguros

Send

200 OK

53 ms

0 B

Params

Body

Auth

Headers4

Scripts

Docs

Preview

Headers2

Cookies

T

JSON

Preview

1

2

3

4

5

6

7

8

9

10

11

12

13

{

"idSeguro": 1,

"nif": "13411227L",

"nombre": "Test",

"ape1": "Tester",

"ape2": "Testing",

"edad": 24,

"numHijos": 0,

"fechaCreacion": "2024-11-10T11:00:00.000+00:00",

"sexo": "Mujer",

"casado": false,

"embarazada": false

}

No body returned for response

Request getAll

GET

http://localhost:8080/seguros

Send

Ejemplo de request que ha funcionado correctamente.

200 OK

53 ms

2024 B

4 Days Ago

Preview

Headers3

Cookies

Tests0/0

Mock

Console

Preview

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

[

{

"idSeguro": 1,

"nif": "12345678A",

"nombre": "Juan",

"ape1": "Pérez",

"ape2": "García",

"edad": 35,

"numHijos": 2,

"fechaCreacion": "2024-11-01T09:00:00.000+00:00",

"sexo": "Hombre",

"casado": true,

"embarazada": false

},

{

"idSeguro": 2,

"nif": "87654321B",

"nombre": "María",

"ape1": "López",

"ape2": null,

"edad": 28,

"numHijos": 1,

"fechaCreacion": "2024-10-20T12:30:00.000+00:00",

"sexo": "Mujer",

"casado": true,

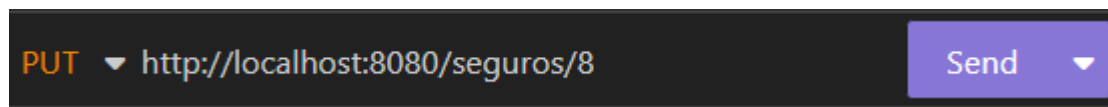
"embarazada": true

},

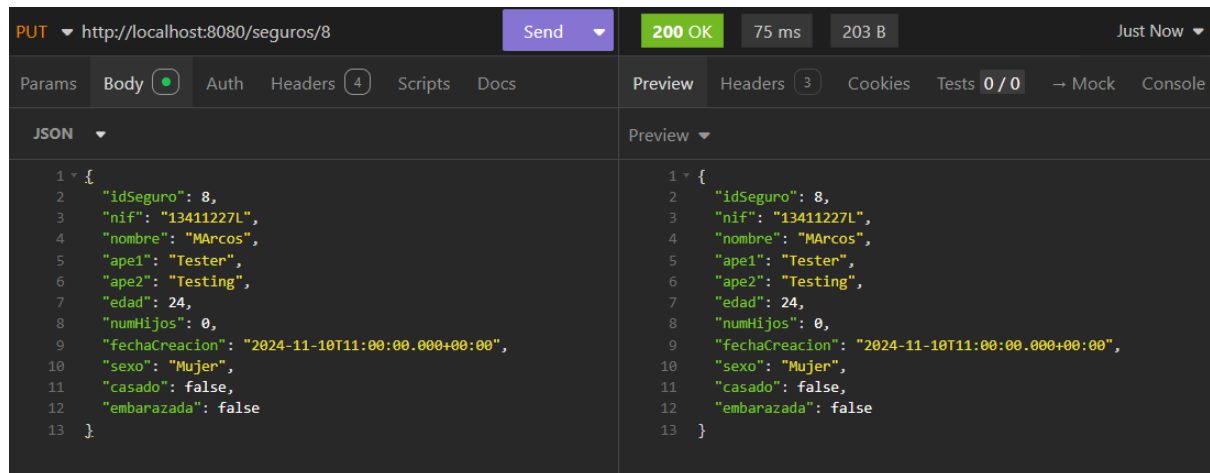
]

La request getAll no puede funcionar de forma errónea, siempre devolverá una lista con todos los seguros, haya o no seguros.

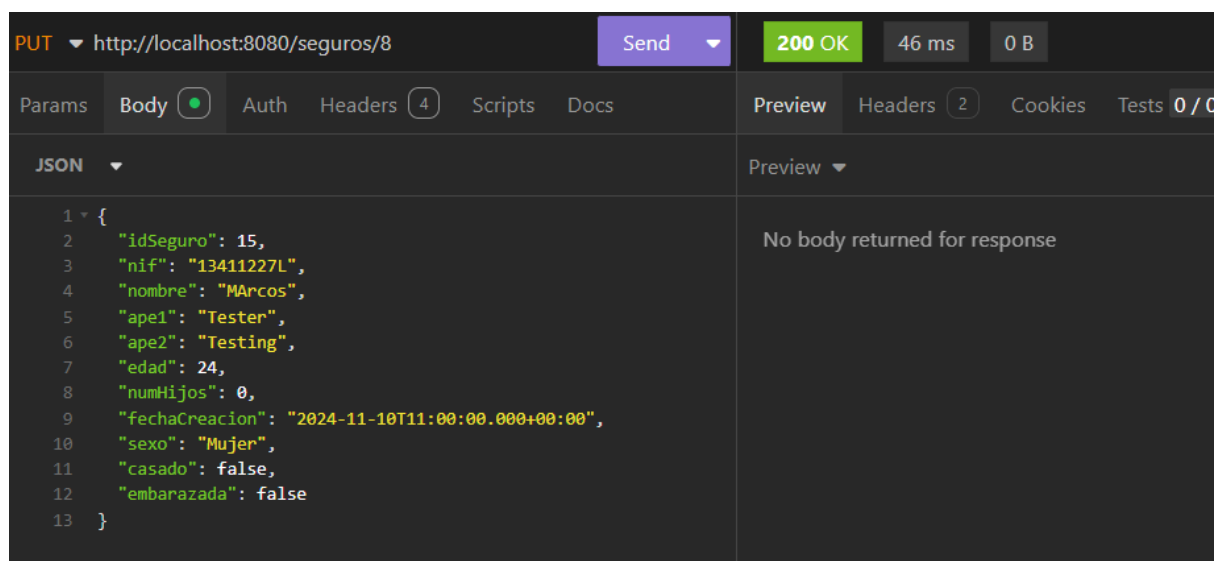
Request actualizarSeguro



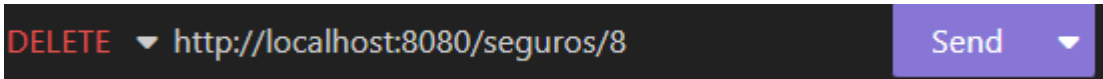
Ejemplo de request que ha funcionado correctamente y actualizado el seguro en la base de datos.



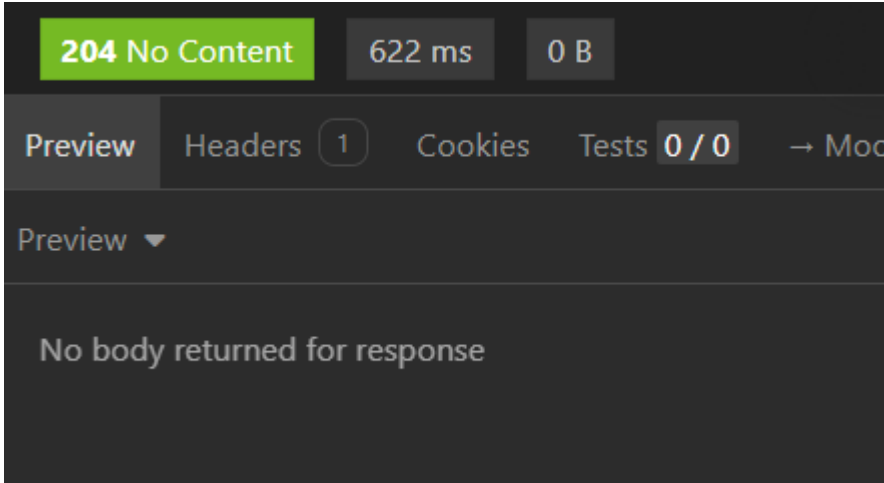
Ejemplo de request que no ha actualizado ningún seguro ya que no existe ningún seguro con la Id establecida.



Request deleteSeguro



Ejemplo de la request que ha funcionado correctamente.



Ejemplo de ejecución con erronea ya que no existe ningún seguro con la Id introducida.

