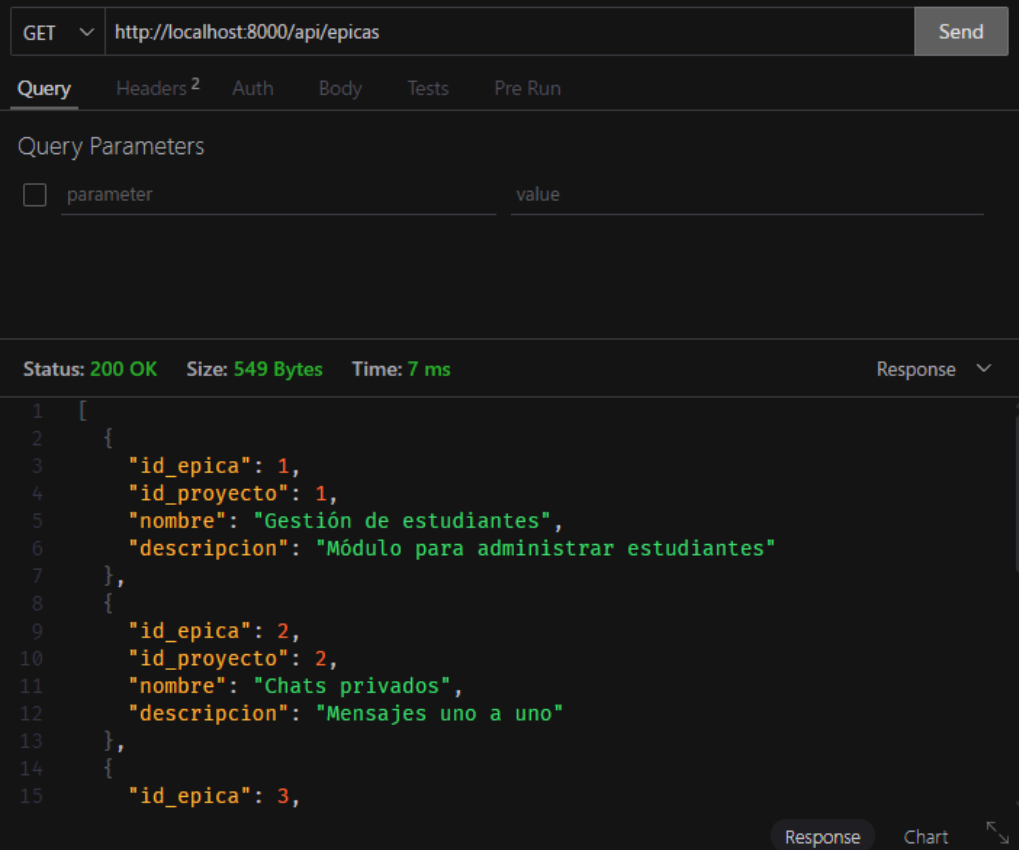


## Mariana / Épicas

### 1. Obtener todas las épicas

GET 

/api/epicas



GET ⌵ http://localhost:8000/api/epicas Send


Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 549 Bytes Time: 7 ms Response ⌵

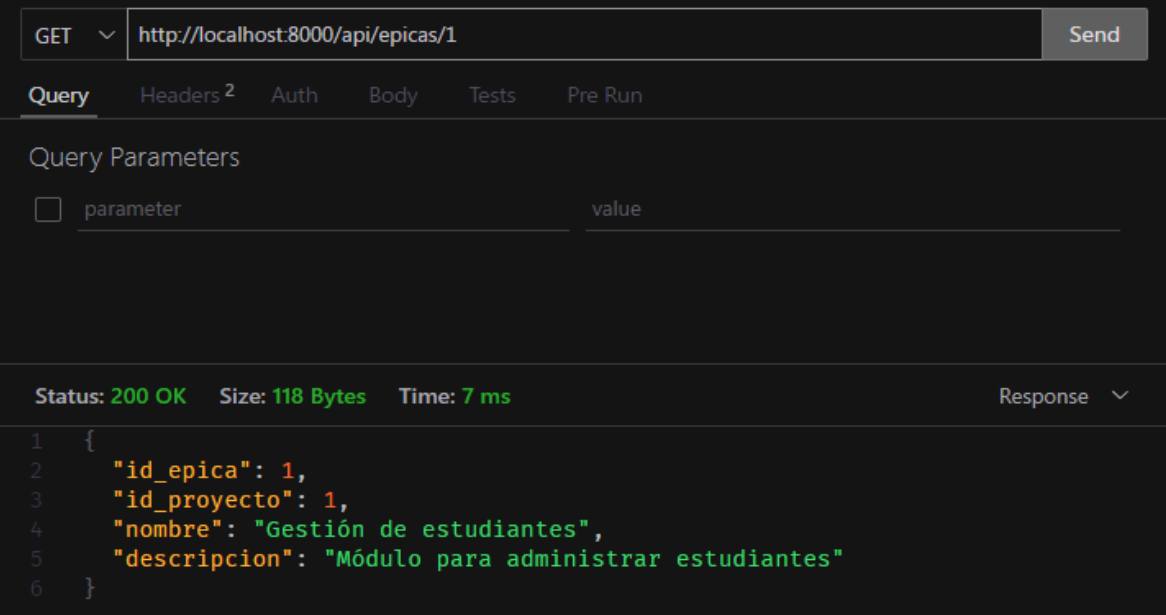
```
1  [  
2    {  
3      "id_epica": 1,  
4      "id_proyecto": 1,  
5      "nombre": "Gestión de estudiantes",  
6      "descripcion": "Módulo para administrar estudiantes"  
7    },  
8    {  
9      "id_epica": 2,  
10     "id_proyecto": 2,  
11     "nombre": "Chats privados",  
12     "descripcion": "Mensajes uno a uno"  
13   },  
14   {  
15     "id_epica": 3,
```

Response Chart 

### 2. Obtener una épica por ID

GET 

/api/epicas/{id}



GET ⌵ http://localhost:8000/api/epicas/1 Send

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 118 Bytes Time: 7 ms Response ⌵

```
1  {  
2    "id_epica": 1,  
3    "id_proyecto": 1,  
4    "nombre": "Gestión de estudiantes",  
5    "descripcion": "Módulo para administrar estudiantes"  
6  }
```

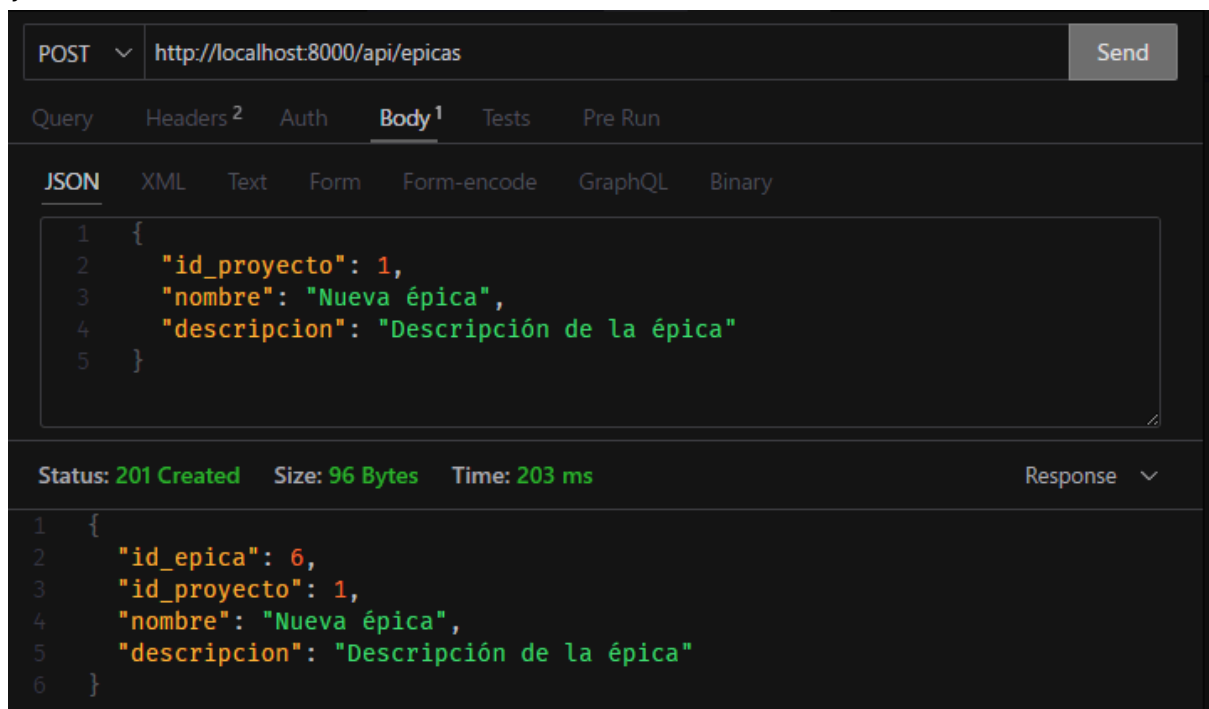
### 3. Crear una nueva épica

**POST** ✓

/api/epicas

**Body (JSON):**

```
{
  "id_proyecto": 1,
  "nombre": "Nueva épica",
  "descripcion": "Descripción de la épica"
}
```




### 4. Actualizar una épica existente

**PUT** ✓

/api/epicas/{id}

**Body (JSON):**


```
{
  "nombre": "Nombre actualizado",
  "descripcion": "Descripción actualizada"
}
```

PUT  http://localhost:8000/api/epicas/6 Send

Query Headers <sup>2</sup> Auth Body <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "nombre": "Nombre actualizado",
3   "descripcion": "Descripción actualizada"
4 }
```


Status: 200 OK Size: 87 Bytes Time: 20 ms Response 

```
1 {
2   "id_epica": "6",
3   "nombre": "Nombre actualizado",
4   "descripcion": "Descripción actualizada"
5 }
```

## 5. Eliminar una épica

**DELETE** 


/api/epicas/{id}

DELETE  http://localhost:8000/api/epicas/6 Send

Query Headers <sup>2</sup> Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1
```

Status: 200 OK Size: 30 Bytes Time: 16 ms Response 

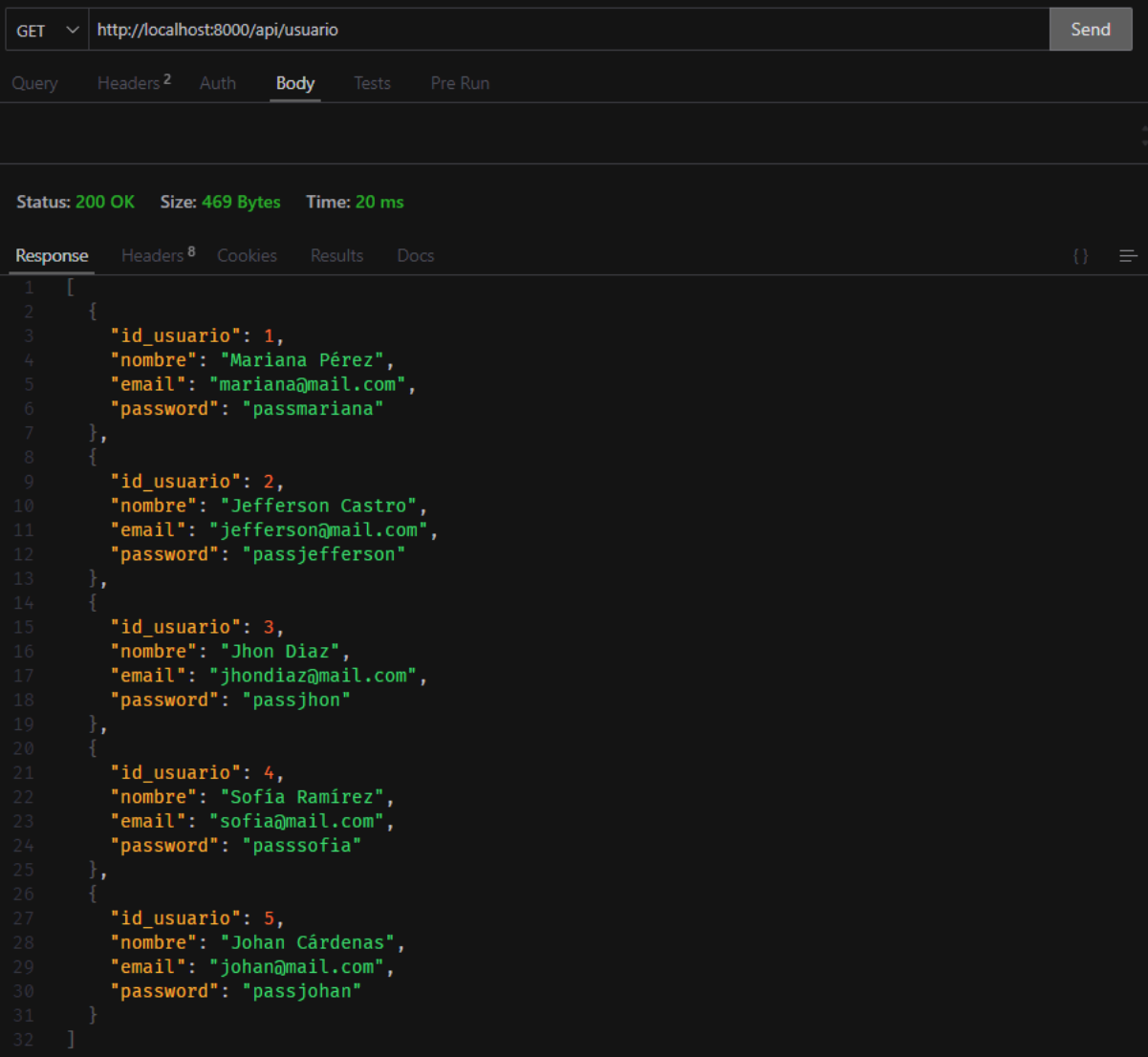
```
1 {
2   "message": "Épica eliminada"
3 }
```


## Sofia / Usuario

1. Listar todos los usuarios

GET 

/api/usuario



GET  http://localhost:8000/api/usuario Send

Query Headers <sup>2</sup> Auth Body Tests Pre Run

Status: 200 OK Size: 469 Bytes Time: 20 ms

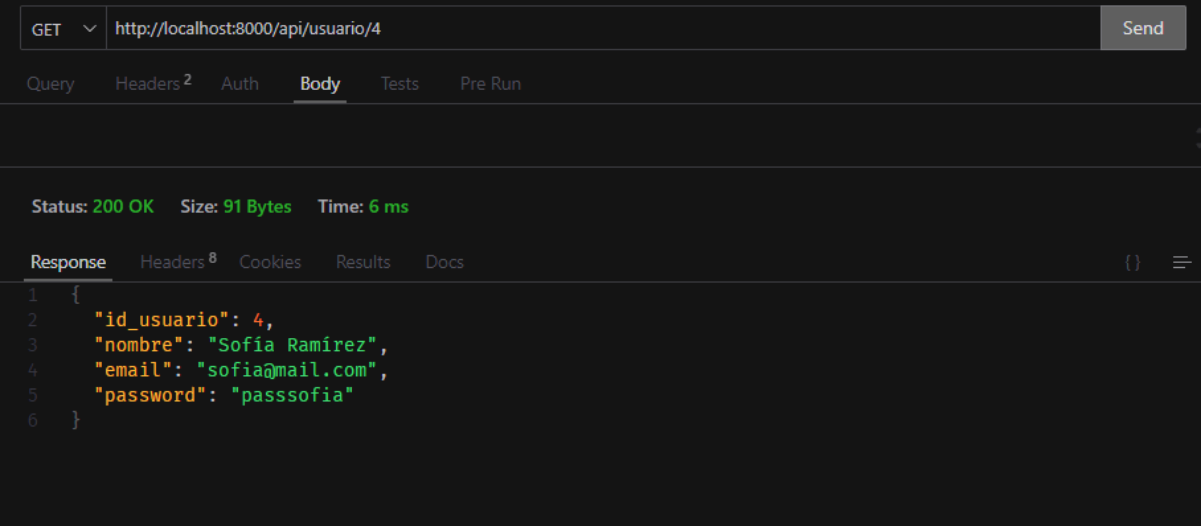
Response Headers <sup>8</sup> Cookies Results Docs {} ≡

```
1  [
2    {
3      "id_usuario": 1,
4      "nombre": "Mariana Pérez",
5      "email": "mariana@mail.com",
6      "password": "passmariana"
7    },
8    {
9      "id_usuario": 2,
10     "nombre": "Jefferson Castro",
11     "email": "jefferson@mail.com",
12     "password": "passjefferson"
13   },
14   {
15     "id_usuario": 3,
16     "nombre": "Jhon Diaz",
17     "email": "jhondiaz@mail.com",
18     "password": "passjhon"
19   },
20   {
21     "id_usuario": 4,
22     "nombre": "Sofia Ramírez",
23     "email": "sofia@mail.com",
24     "password": "passsofia"
25   },
26   {
27     "id_usuario": 5,
28     "nombre": "Johan Cárdenas",
29     "email": "johan@mail.com",
30     "password": "passjohan"
31   }
32 ]
```

## 2. Obtener un usuario por ID

GET 

/api/usuario/4



GET ⌵ http://localhost:8000/api/usuario/4 Send

Query Headers <sup>2</sup> Auth **Body** Tests Pre Run

Status: 200 OK Size: 91 Bytes Time: 6 ms

Response Headers <sup>8</sup> Cookies Results Docs {} ≡

```
1 {
2   "id_usuario": 4,
3   "nombre": "Sofia Ramirez",
4   "email": "sofia@mail.com",
5   "password": "passsofia"
6 }
```

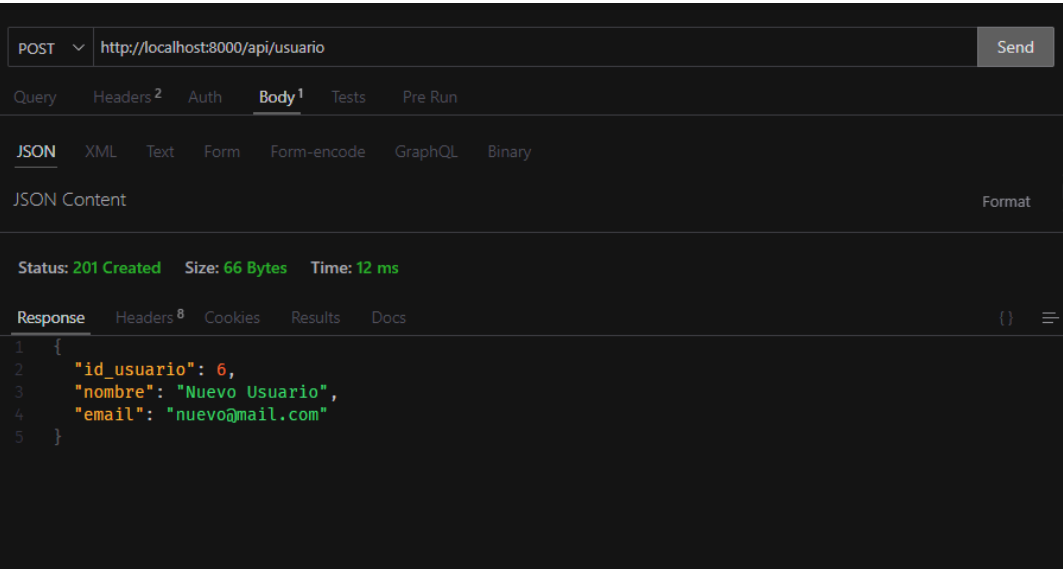
## 3. Crear un usuario nuevo

POST 

/api/usuario

Body (JSON):

```
{
  "nombre": "Nuevo usuario a",
  "email": "nuevo@gmail.com",
  "password": "12332423"
}
```



POST ⌵ http://localhost:8000/api/usuario Send

Query Headers <sup>2</sup> Auth **Body** <sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

Status: 201 Created Size: 66 Bytes Time: 12 ms

Response Headers <sup>8</sup> Cookies Results Docs {} ≡

```
1 {
2   "id_usuario": 6,
3   "nombre": "Nuevo Usuario",
4   "email": "nuevo@mail.com"
5 }
```

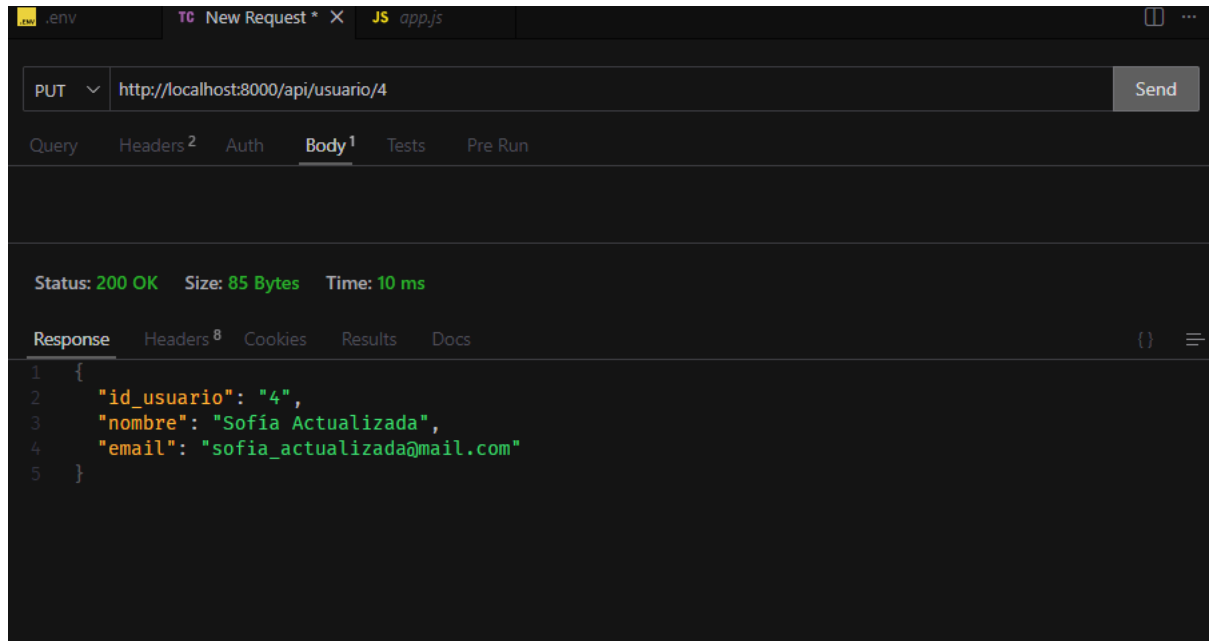
#### 4. Actualizar un usuario

**PUT** ✓

/api/usuario/4

**Body (JSON):**

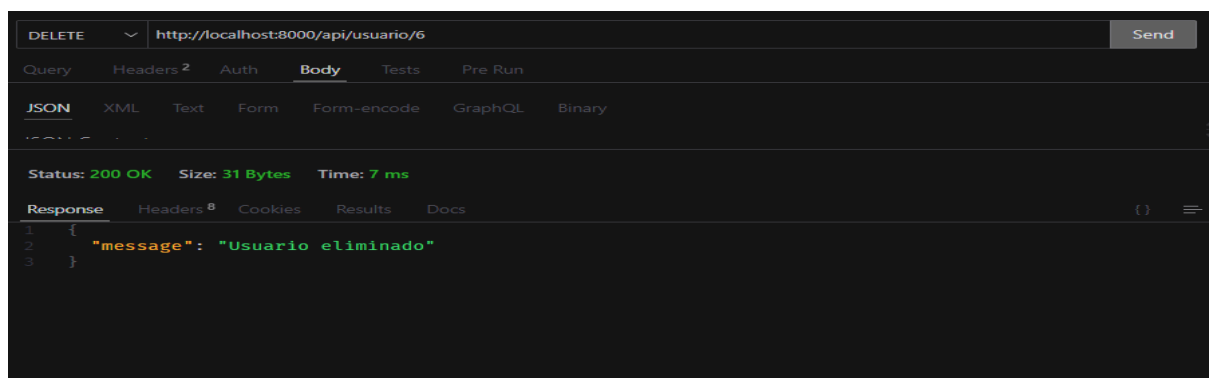
```
{  
  "nombre": "Sofía Actualizada",  
  "email": "sofia_actualizada@mail.com"  
}
```



#### 5. Eliminar un usuario

**DELETE** ✓

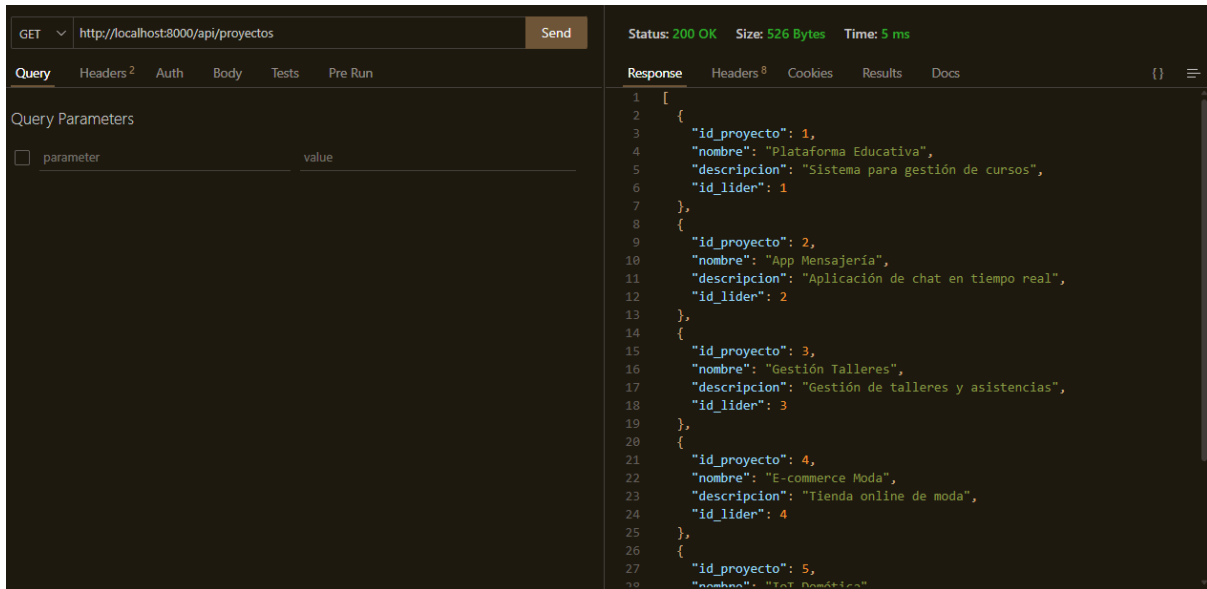
/api/usuario/6



# Jefferson / Proyecto

## 1. Listar todos proyectos: GET:

/api/proyectos

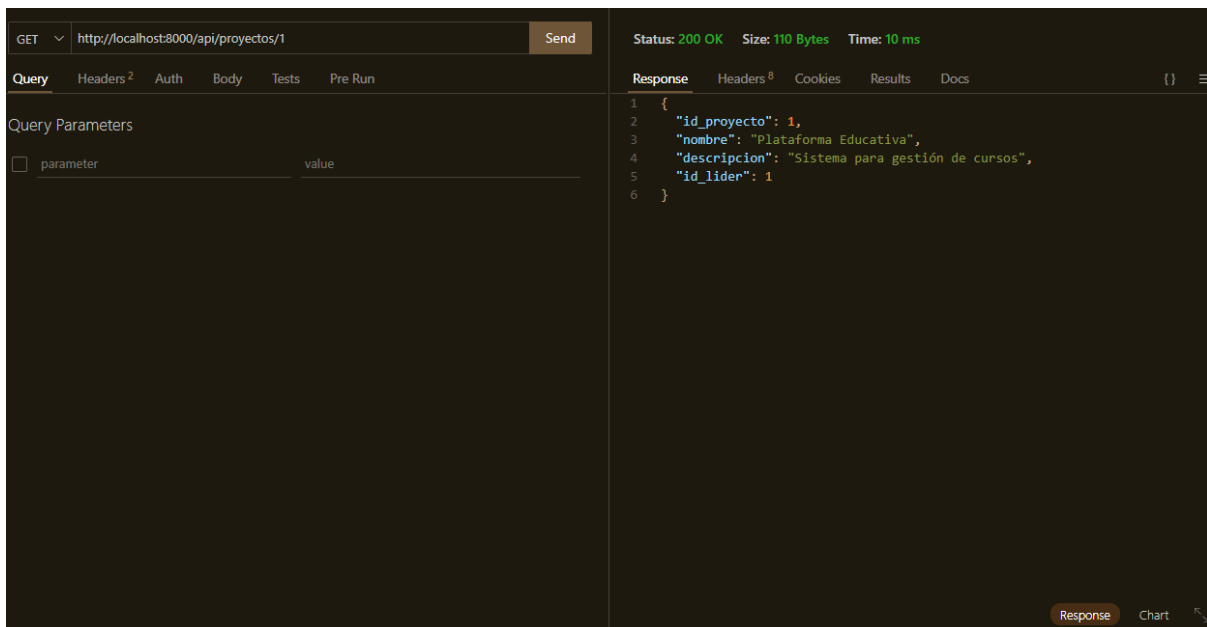


The screenshot shows a REST client interface. The top bar displays the method 'GET' and the URL 'http://localhost:8000/api/proyectos'. The 'Send' button is visible. Below the URL bar, the 'Query' tab is selected, showing 'Query Parameters' with a table with columns 'parameter' and 'value'. The 'Response' tab is also visible, showing the status '200 OK', size '526 Bytes', and time '5 ms'. The response body is a JSON array of five project objects.

```
1  [
2  {
3    "id_proyecto": 1,
4    "nombre": "Plataforma Educativa",
5    "descripcion": "Sistema para gestión de cursos",
6    "id_lider": 1
7  },
8  {
9    "id_proyecto": 2,
10   "nombre": "App Mensajería",
11   "descripcion": "Aplicación de chat en tiempo real",
12   "id_lider": 2
13 },
14 {
15   "id_proyecto": 3,
16   "nombre": "Gestión Talleres",
17   "descripcion": "Gestión de talleres y asistencias",
18   "id_lider": 3
19 },
20 {
21   "id_proyecto": 4,
22   "nombre": "E-commerce Moda",
23   "descripcion": "Tienda online de moda",
24   "id_lider": 4
25 },
26 {
27   "id_proyecto": 5,
28   "nombre": "IoT Domótica"
29 }
```

## 2. Obtener por id los proyectos: GET

api/proyectos/1



The screenshot shows a REST client interface. The top bar displays the method 'GET' and the URL 'http://localhost:8000/api/proyectos/1'. The 'Send' button is visible. Below the URL bar, the 'Query' tab is selected, showing 'Query Parameters' with a table with columns 'parameter' and 'value'. The 'Response' tab is also visible, showing the status '200 OK', size '110 Bytes', and time '10 ms'. The response body is a JSON object representing a single project.

```
1  {
2    "id_proyecto": 1,
3    "nombre": "Plataforma Educativa",
4    "descripcion": "Sistema para gestión de cursos",
5    "id_lider": 1
6  }
```

### 3. Listar por líder de proyecto: GET: ✓

(ejemplo para id\_lider=1)

/api/proyectos/leader/1

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/proyectos/leader/1`. The response status is 200 OK, with a size of 112 Bytes and a time of 6 ms. The response body is a JSON array containing one object.

```
1 [
2   {
3     "id_proyecto": 1,
4     "nombre": "Plataforma Educativa",
5     "descripcion": "Sistema para gestión de cursos",
6     "id_lider": 1
7   }
8 ]
```

### 4. Crear proyecto: POST ✓

/api/proyectos

The screenshot shows a REST client interface with a POST request to `http://localhost:8000/api/proyectos`. The request body is a JSON object. The response status is 201 Created, with a size of 98 Bytes and a time of 28 ms. The response body is a JSON object.

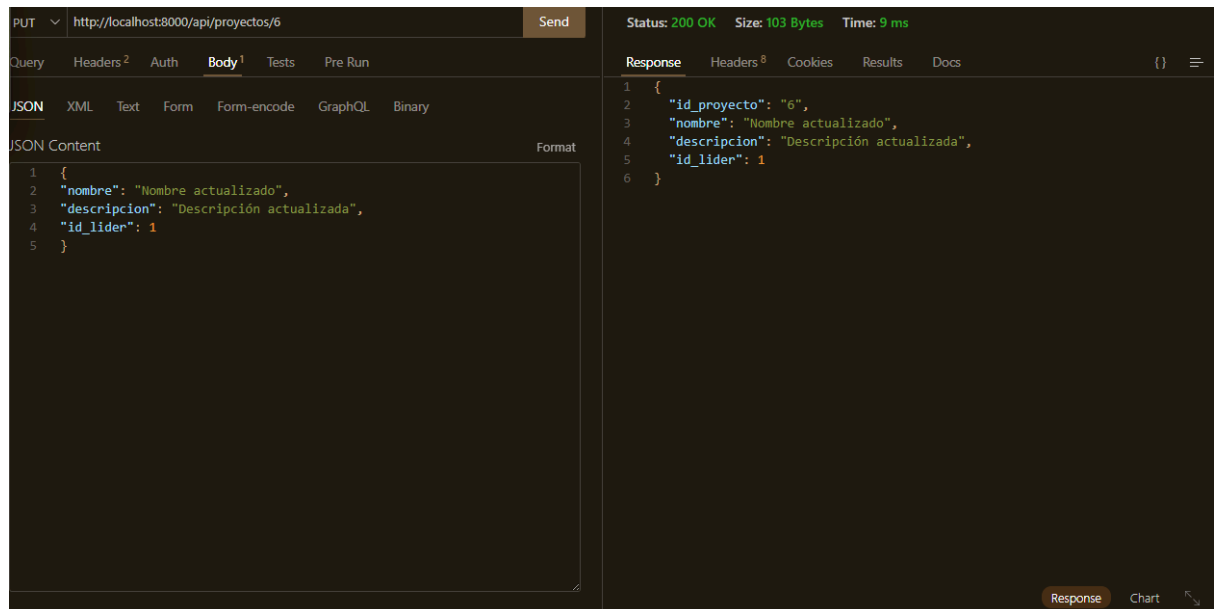
```
1 {
2   "nombre": "Nuevo proyecto",
3   "descripcion": "Descripción del proyecto",
4   "id_lider": 1
5 }
```

```
1 {
2   "id_proyecto": 6,
3   "nombre": "Nuevo proyecto",
4   "descripcion": "Descripción del proyecto",
5   "id_lider": 1
6 }
```



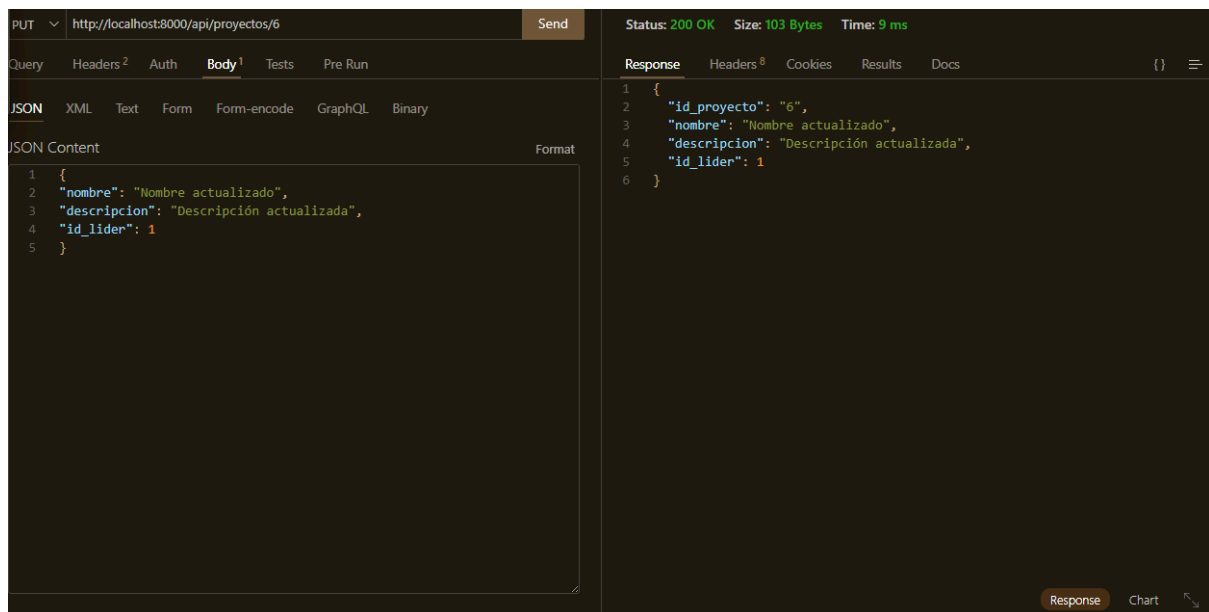
## 5. Actualizar Proyecto: PUT: ✓

/api/proyectos/6



## 6. Eliminar proyecto: DELETE ✓

/api/proyectos/6



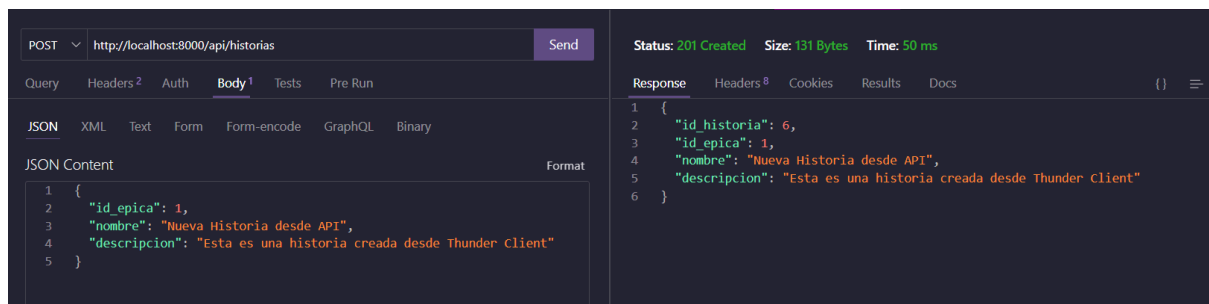
# Johan / historia\_usuario & tarea

## 1. Crear nueva historia:POST

/api/historias

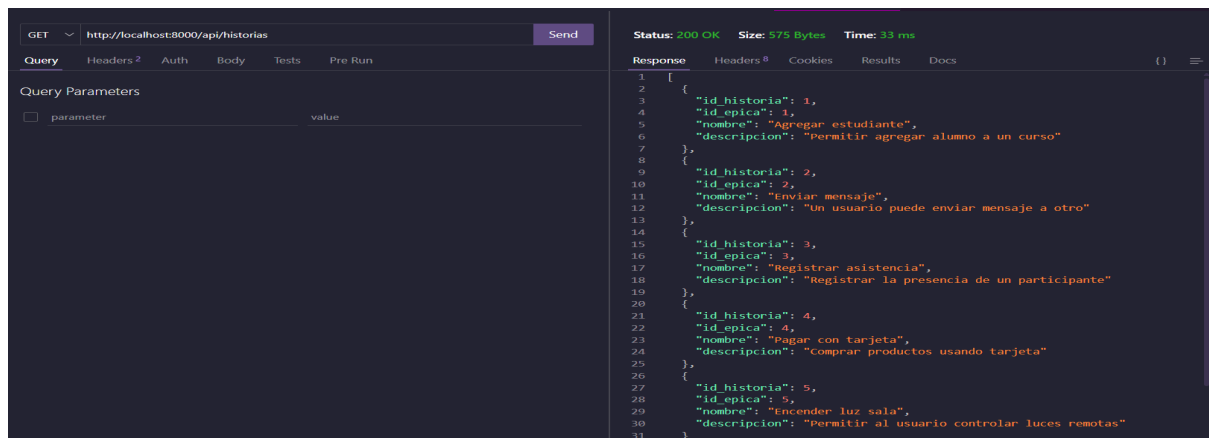
Body (JSON):

```
{  
  
  "id_epica": 1,  
  
  "nombre": "Nueva Historia desde API",  
  
  "descripcion": "Esta es una historia creada desde Thunder Client"  
}
```



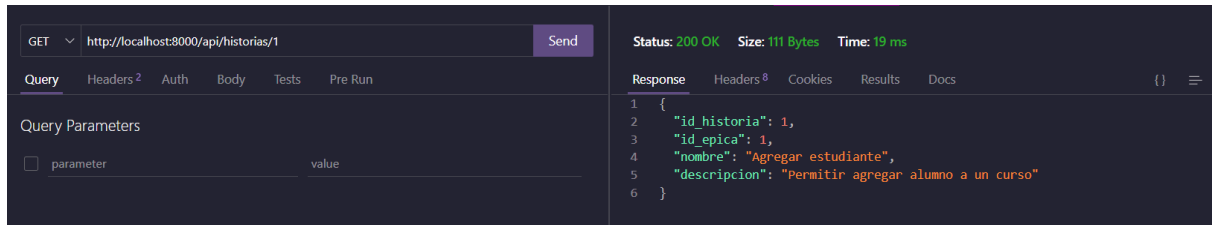
## 2. Listar todas las historias:GET

/api/historias



### 3. Obtener historia por id:GET

/api/historias/{id}



The screenshot shows a REST client interface. The request is a GET to `http://localhost:8000/api/historias/1`. The response status is 200 OK, with a size of 111 Bytes and a time of 19 ms. The response body is a JSON object:

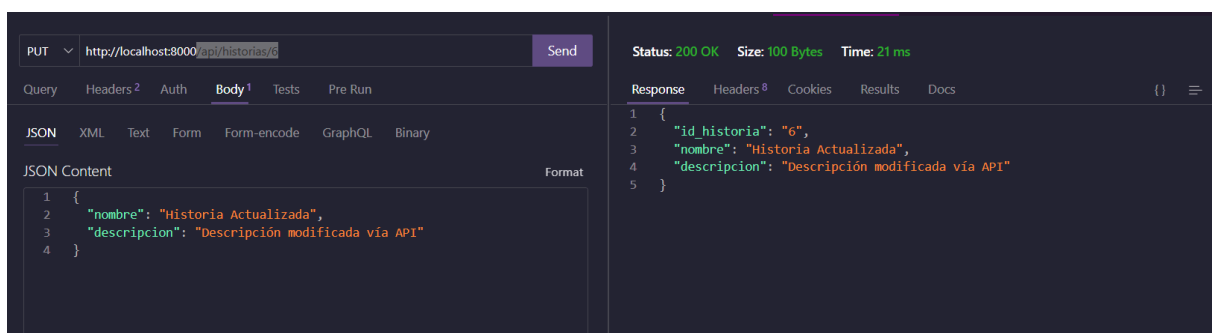
```
1 {
2   "id_historia": 1,
3   "id_epica": 1,
4   "nombre": "Agregar estudiante",
5   "descripcion": "Permitir agregar alumno a un curso"
6 }
```

### 4. Actualizar historia:PUT

/api/historias/{id}

**Body (JSON):**

```
{
  "nombre": "Historia Actualizada",
  "descripcion": "Descripción modificada vía API"
}
```



The screenshot shows a REST client interface. The request is a PUT to `http://localhost:8000/api/historias/6`. The request body is a JSON object: 

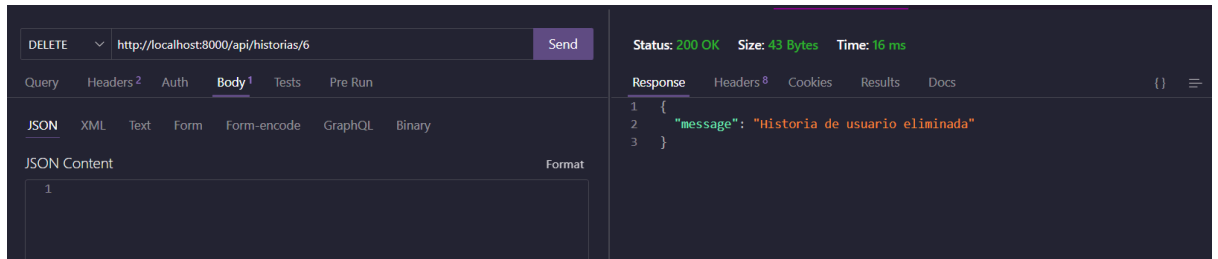
```
{ "nombre": "Historia Actualizada", "descripcion": "Descripción modificada vía API" }
```

. The response status is 200 OK, with a size of 100 Bytes and a time of 21 ms. The response body is a JSON object:

```
1 {
2   "id_historia": "6",
3   "nombre": "Historia Actualizada",
4   "descripcion": "Descripción modificada vía API"
5 }
```

## 5. Eliminar historia:DELETE✔

/api/historias/{id}

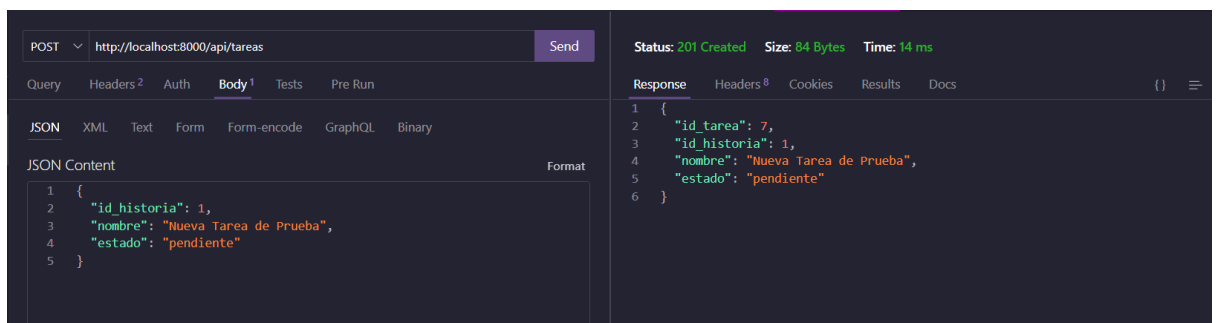


## 1. Crear nueva tarea:POST✔

/api/tareas

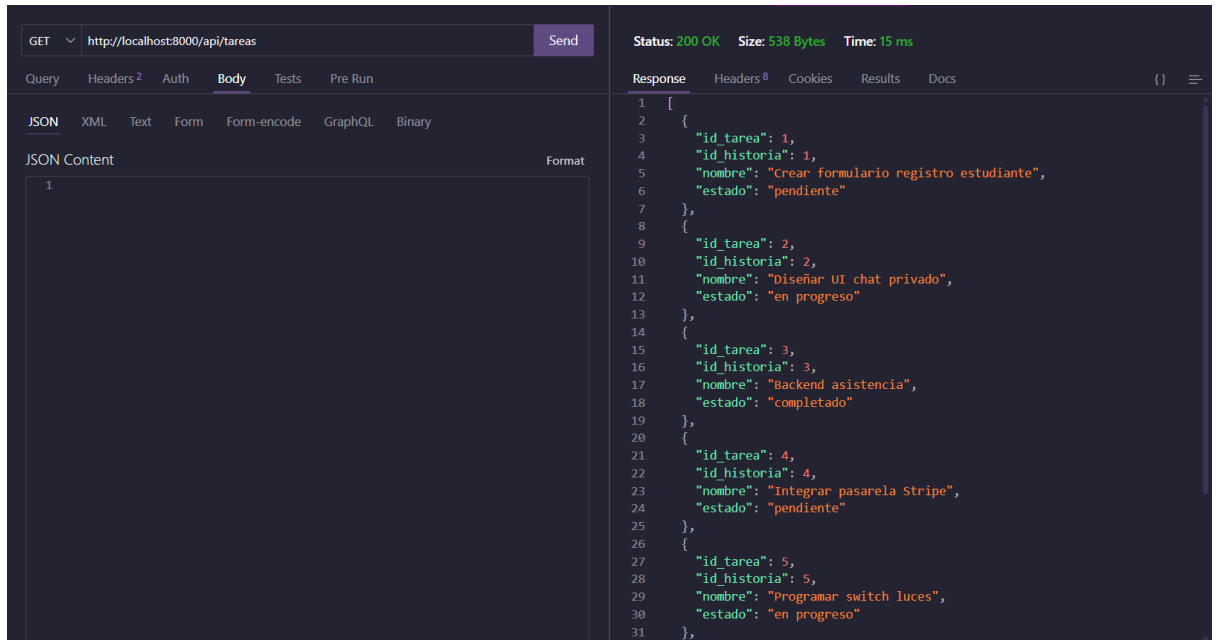
Body (JSON):

```
{  
  "id_historia": 1,  
  "nombre": "Nueva Tarea de Prueba",  
  "estado": "pendiente"  
}
```



## 2. Listar todas las tareas: GET

/api/tareas

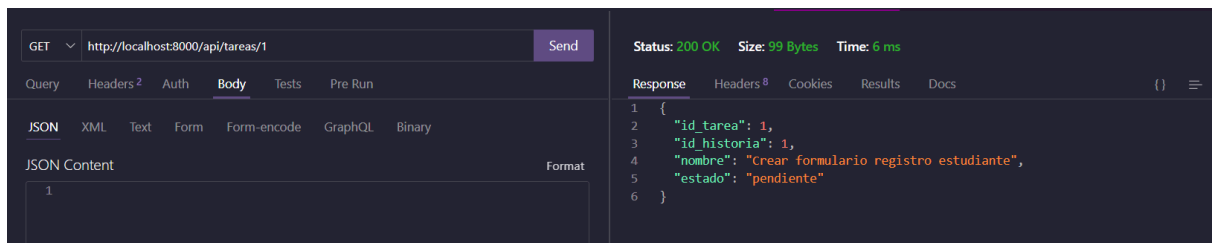


The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/tareas`. The response is a JSON array of 5 task objects. The left pane shows the JSON content, and the right pane shows the raw response text.

```
1 [
2   {
3     "id_tarea": 1,
4     "id_historia": 1,
5     "nombre": "Crear formulario registro estudiante",
6     "estado": "pendiente"
7   },
8   {
9     "id_tarea": 2,
10    "id_historia": 2,
11    "nombre": "Diseñar UI chat privado",
12    "estado": "en progreso"
13  },
14  {
15    "id_tarea": 3,
16    "id_historia": 3,
17    "nombre": "Backend asistencia",
18    "estado": "completado"
19  },
20  {
21    "id_tarea": 4,
22    "id_historia": 4,
23    "nombre": "Integrar pasarela Stripe",
24    "estado": "pendiente"
25  },
26  {
27    "id_tarea": 5,
28    "id_historia": 5,
29    "nombre": "Programar switch luces",
30    "estado": "en progreso"
31  },
32 ]
```

## 3. Obtener una tarea por ID: GET

/api/tareas/{id}



The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/tareas/1`. The response is a single JSON object representing the first task. The left pane shows the JSON content, and the right pane shows the raw response text.

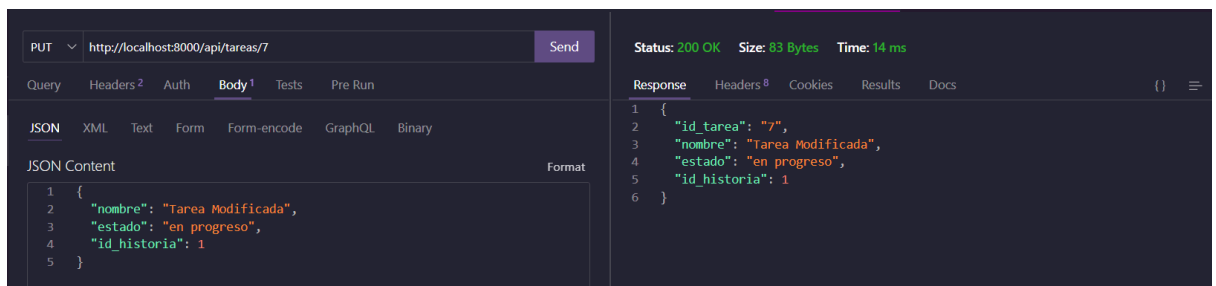
```
1 {
2   "id_tarea": 1,
3   "id_historia": 1,
4   "nombre": "Crear formulario registro estudiante",
5   "estado": "pendiente"
6 }
```

#### 4. Actualizar tarea:PUT

/api/tareas/{id}

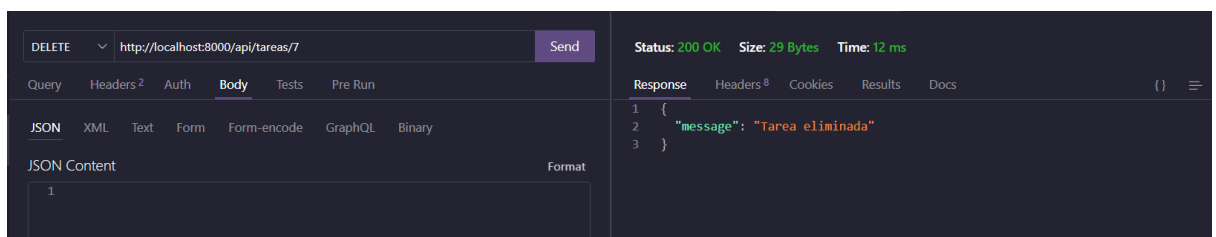
Body (JSON):

```
{  
  
  "nombre": "Tarea Modificada",  
  
  "estado": "en progreso",  
  
  "id_historia": 1  
}
```



#### 5. Eliminar tarea:DELETE

/api/tareas/{id}

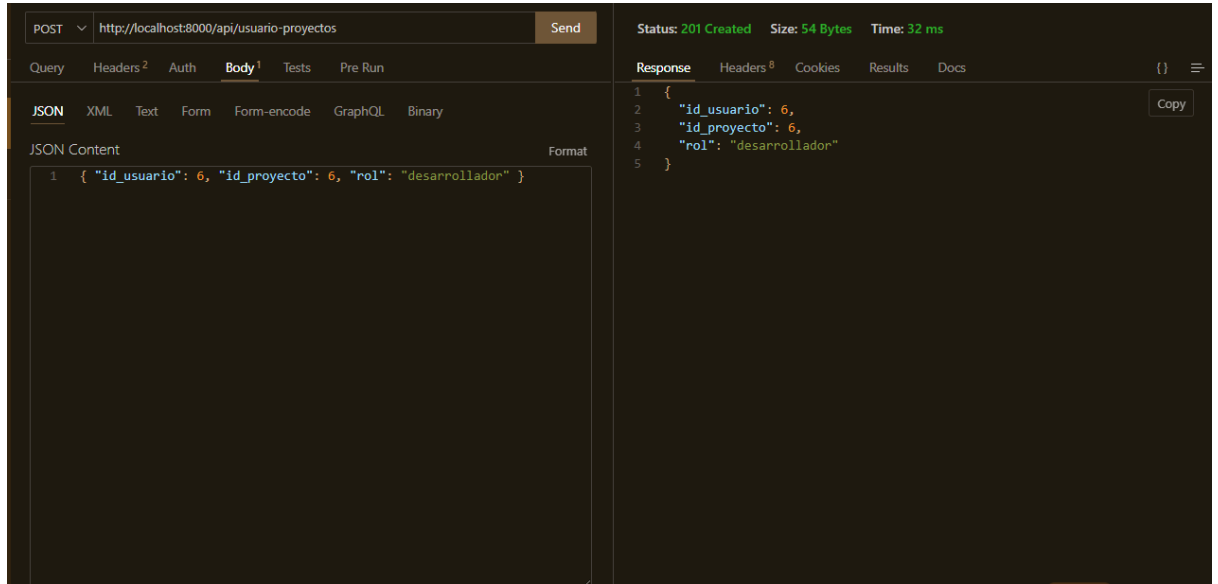


## Jhon / usuario\_proyecto & usuario\_tarea

usuario\_proyecto

### 1. Asignar usuario a proyecto: POST ✓

api/usuario-proyectos

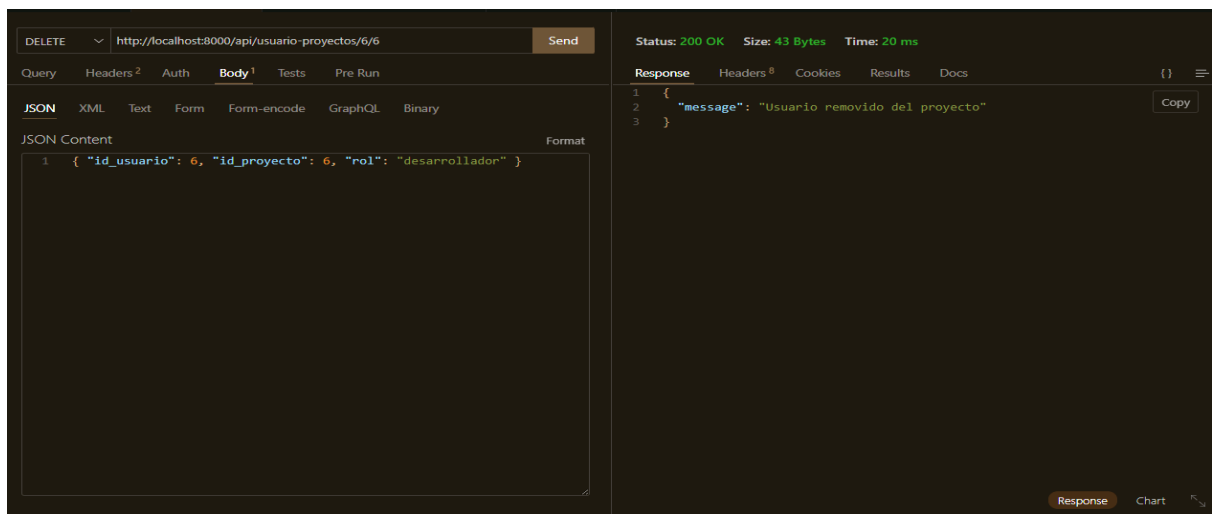


### 2. Quitar usuario de proyecto:DELETE ✓

/api/usuario-proyectos/:id\_usuario/:id\_proyecto

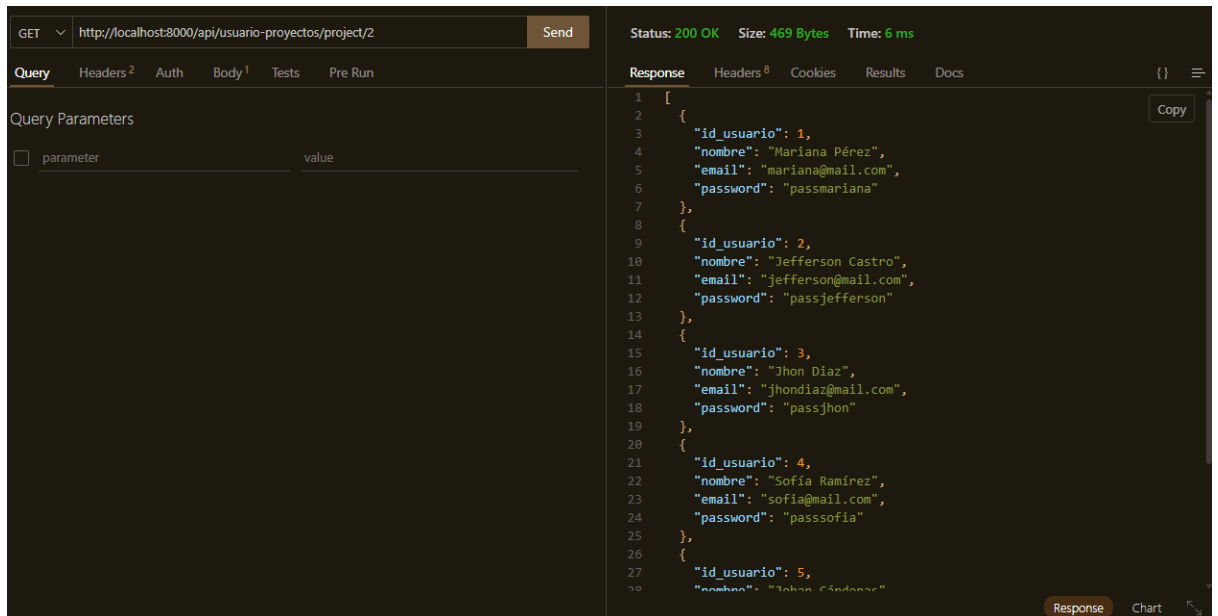
Ejemplo:

/api/usuario-proyectos/6/6



### 3. Listar usuarios de un proyecto: GET ✓

api/usuario-proyectos/project/2



GET ☐ http://localhost:8000/api/usuario-proyectos/project/2

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

Query Parameters

parameter	value
-----------	-------

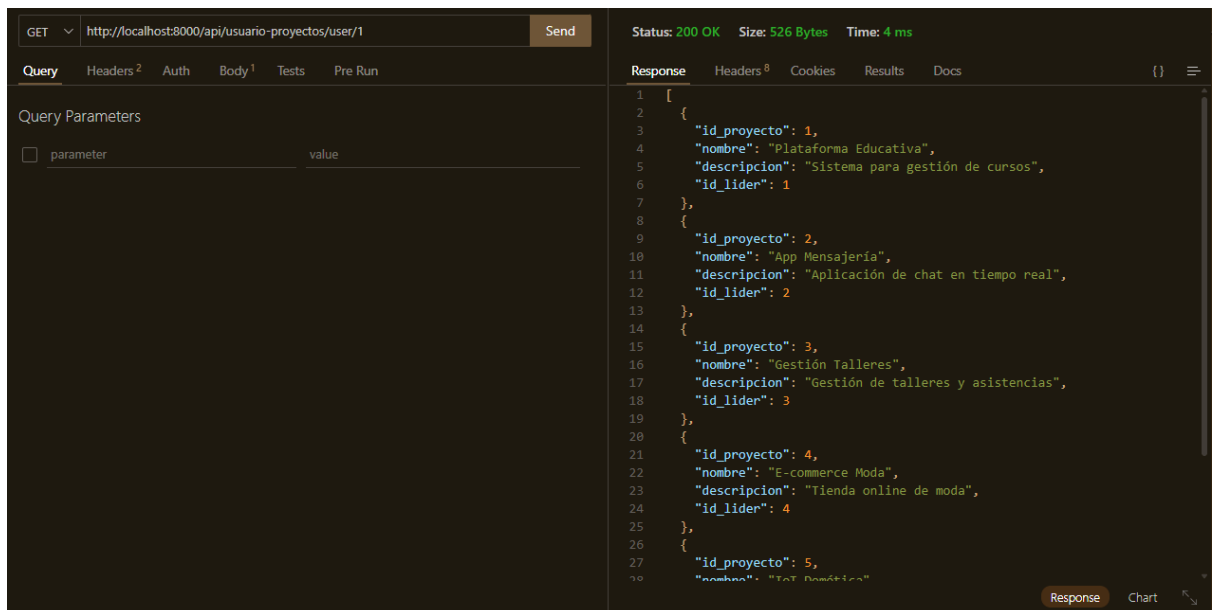
Status: 200 OK Size: 469 Bytes Time: 6 ms

Response Headers<sup>8</sup> Cookies Results Docs

```
1 [
2 {
3   "id_usuario": 1,
4   "nombre": "Mariana Pérez",
5   "email": "mariana@mail.com",
6   "password": "passmariana"
7 },
8 {
9   "id_usuario": 2,
10  "nombre": "Jefferson Castro",
11  "email": "jefferson@mail.com",
12  "password": "passjefferson"
13 },
14 {
15   "id_usuario": 3,
16   "nombre": "Jhon Diaz",
17   "email": "jhondiaz@mail.com",
18   "password": "passjhon"
19 },
20 {
21   "id_usuario": 4,
22   "nombre": "Sofia Ramirez",
23   "email": "sofia@mail.com",
24   "password": "passsofia"
25 },
26 {
27   "id_usuario": 5,
28   "nombre": "Johan Candoso"
29 }
```

### 4. Listar proyectos de un usuario — GET ✓

/api/usuario-proyectos/user/1



GET ☐ http://localhost:8000/api/usuario-proyectos/user/1

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 526 Bytes Time: 4 ms

Response Headers<sup>8</sup> Cookies Results Docs

```
1 [
2 {
3   "id_proyecto": 1,
4   "nombre": "Plataforma Educativa",
5   "descripcion": "Sistema para gestión de cursos",
6   "id_lider": 1
7 },
8 {
9   "id_proyecto": 2,
10  "nombre": "App Mensajería",
11  "descripcion": "Aplicación de chat en tiempo real",
12  "id_lider": 2
13 },
14 {
15   "id_proyecto": 3,
16   "nombre": "Gestión Talleres",
17   "descripcion": "Gestión de talleres y asistencias",
18   "id_lider": 3
19 },
20 {
21   "id_proyecto": 4,
22   "nombre": "E-commerce Moda",
23   "descripcion": "Tienda online de moda",
24   "id_lider": 4
25 },
26 {
27   "id_proyecto": 5,
28   "nombre": "E-commerce Moda"
29 }
```

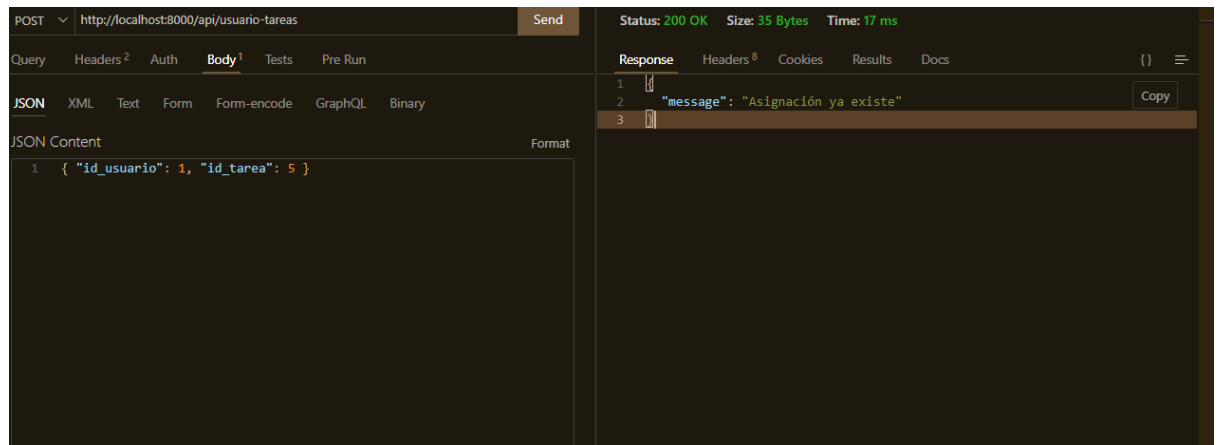


usuario\_tarea

## 1. Asignar usuario a tarea :POST

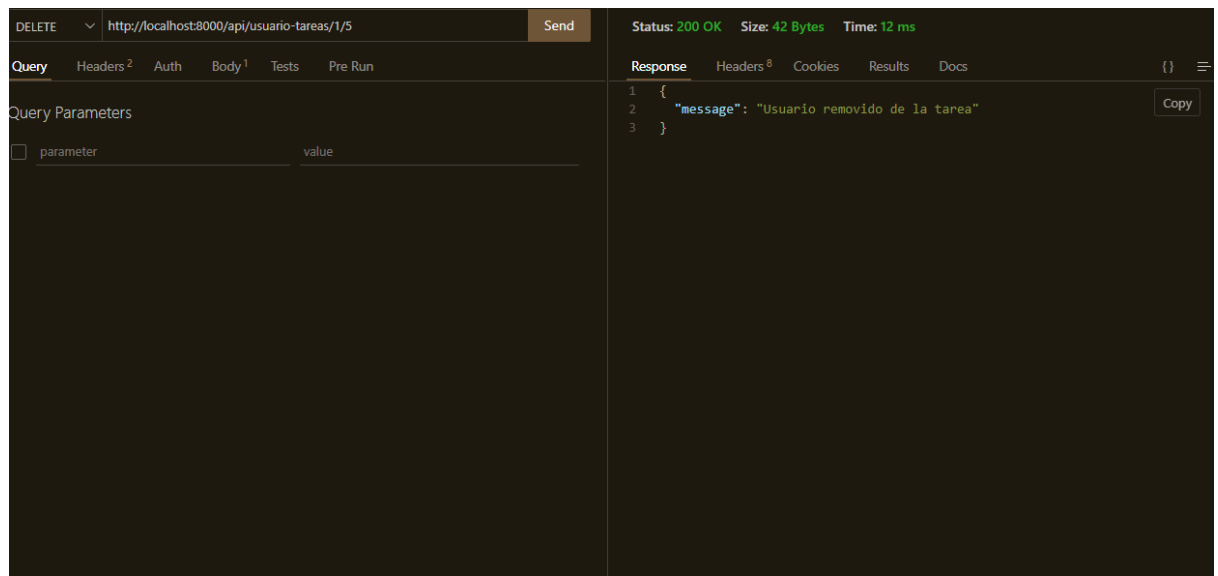
api/usuario-tareas

- Si la asignación ya existe → responde 200 { message: "Asignación ya existe" }
- Si falta usuario o tarea → responde 400 con mensaje explicando que faltan registros padre
- Si se crea correctamente → 201 con { id\_usuario, id\_tarea }



## 2. Quitar usuario de tarea :DELETE

/api/usuario-tareas/1/5



### 3. Listar usuarios de una tarea :GET ✓

/api/usuario-tareas/task/5

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/usuario-tareas/task/5`. The response status is 200 OK, with a size of 93 Bytes and a time of 11 ms. The response body is a JSON array containing one object for user ID 5.

```
GET http://localhost:8000/api/usuario-tareas/task/5
Send

Query Parameters
parameter value

Response Headers Cookies Results Docs
1 [
2 {
3   "id_usuario": 5,
4   "nombre": "Johan Cárdenas",
5   "email": "johan@mail.com",
6   "password": "passjohan"
7 }
8 ]
```

### 4. Listar tareas de un usuario: GET ✓

/api/usuario-tareas/user/1

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/usuario-tareas/user/1`. The response status is 200 OK, with a size of 101 Bytes and a time of 5 ms. The response body is a JSON array containing one object for task ID 1.

```
GET http://localhost:8000/api/usuario-tareas/user/1
Send

Query Parameters
parameter value

Response Headers Cookies Results Docs
1 [
2 {
3   "id_tarea": 1,
4   "id_historia": 1,
5   "nombre": "Crear formulario registro estudiante",
6   "estado": "pendiente"
7 }
8 ]
```