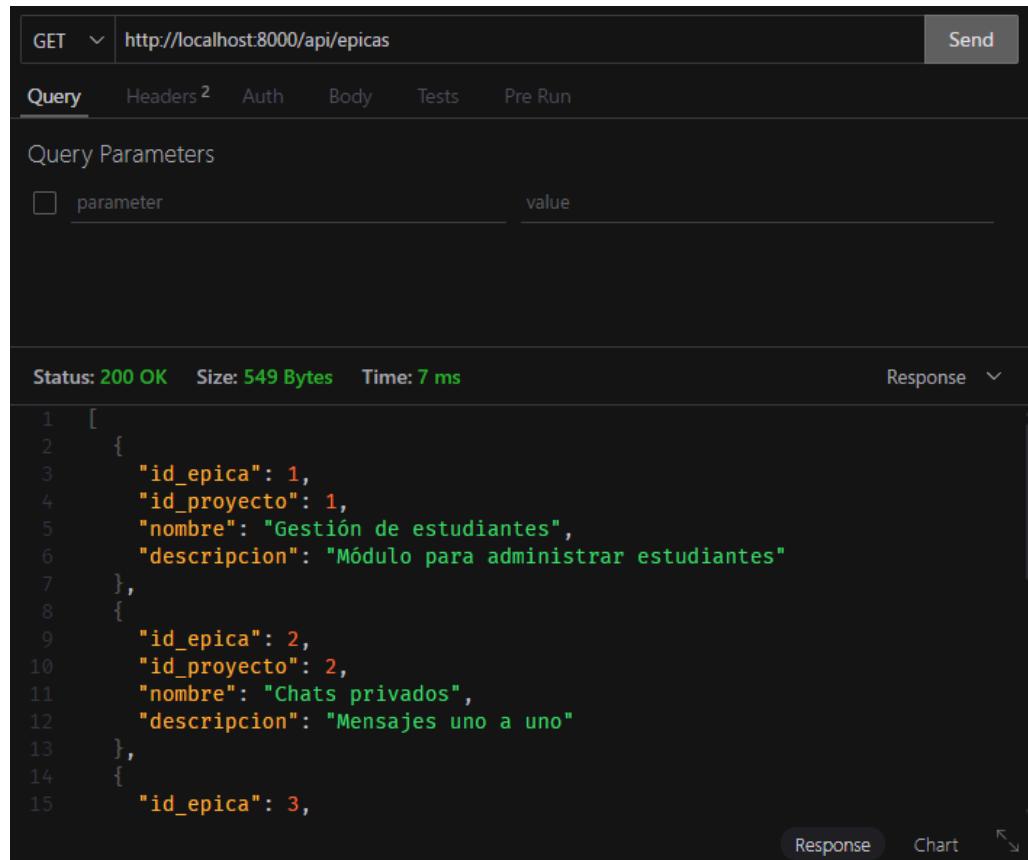


# Mariana / Épicas

## 1. Obtener todas las épicas

**GET** ✓

/api/epicas



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

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

Query Parameters

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

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

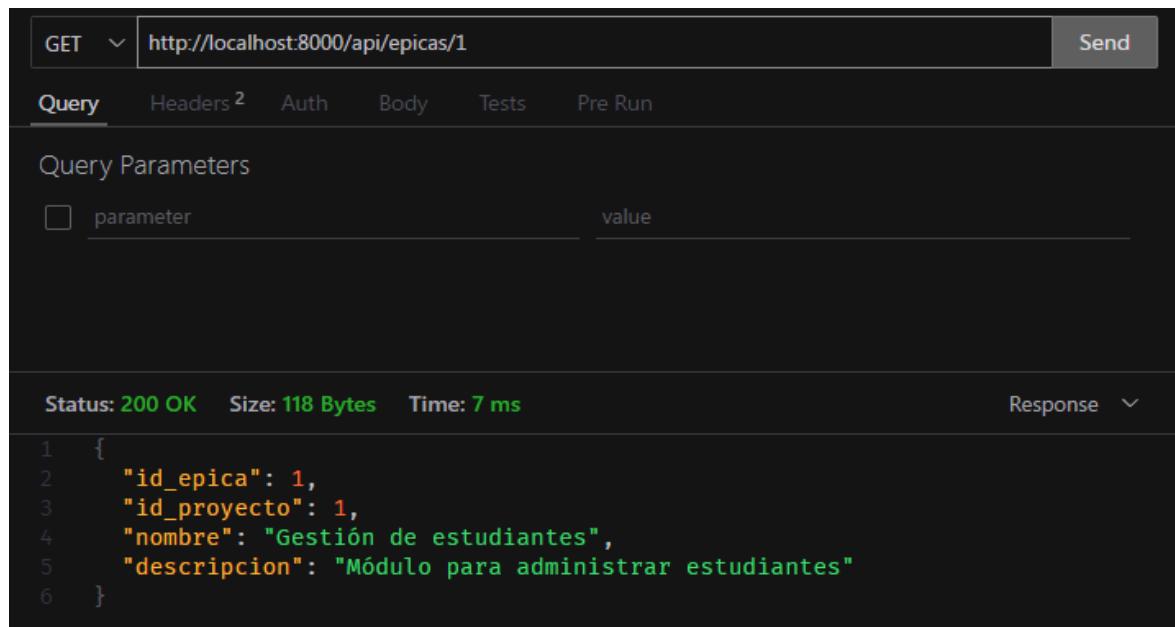
```
1 [  
2 {  
3   "id_epica": 1,  
4   "id_proyecto": 1,  
5   "nombre": "Gestión de estudiantes",  
6   "descripcion": "Módulo para administrar estudiantes"  
7 },  
8 {  
9   "id_epica": 2,  
10  "id_proyecto": 2,  
11  "nombre": "Chats privados",  
12  "descripcion": "Mensajes uno a uno"  
13 },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

Response Chart ↴

## 2. Obtener una épica por ID

**GET** ✓

/api/epicas/{id}



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

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

Query Parameters

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

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

```
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"  
}
```

The screenshot shows the Postman interface. At the top, it says 'POST' and 'http://localhost:8000/api/epicas'. Below that is a toolbar with 'Query', 'Headers 2', 'Auth', 'Body 1' (which is selected), 'Tests', and 'Pre Run'. Under 'Body', there are tabs for 'JSON', 'XML', 'Text', 'Form', 'Form-encode', 'GraphQL', and 'Binary'. The 'JSON' tab is selected, displaying the JSON body from above. Below the body editor, the status bar shows 'Status: 201 Created', 'Size: 96 Bytes', and 'Time: 203 ms'. The response section below contains the same JSON data as the body.

```
1 {  
2   "id_proyecto": 1,  
3   "nombre": "Nueva épica",  
4   "descripcion": "Descripción de la épica"  
5 }
```

```
1 {  
2   "id_epica": 6,  
3   "id_proyecto": 1,  
4   "nombre": "Nueva épica",  
5   "descripcion": "Descripción de la épica"  
6 }
```

### 4. Actualizar una épica existente

**PUT** ✓

/api/epicas/{id}

**Body (JSON):**

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

PUT  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  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 ▾

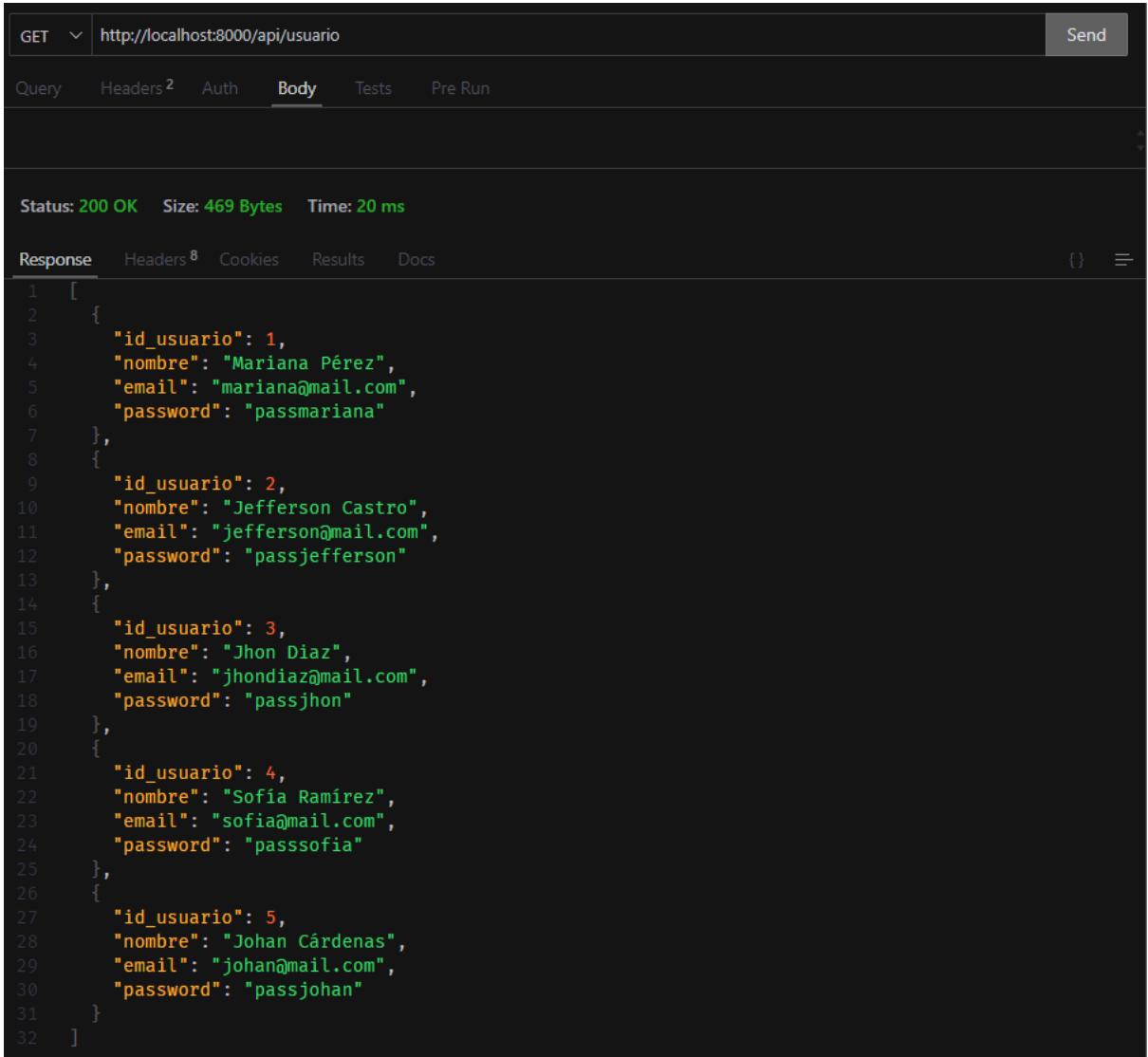
```
1 {
2     "message": "Épica eliminada"
3 }
```

## Sofia / Usuario

1. Listar todos los usuarios

GET 

/api/usuario



The screenshot shows a Postman request to `http://localhost:8000/api/usuario`. The response status is `200 OK`, size is `469 Bytes`, and time is `20 ms`. The response body is a JSON array containing five user objects, each with `id_usuario`, `nombre`, `email`, and `password` fields.

```
[{"id_usuario": 1, "nombre": "Mariana Pérez", "email": "mariana@mail.com", "password": "passmariana"}, {"id_usuario": 2, "nombre": "Jefferson Castro", "email": "jefferson@mail.com", "password": "passjefferson"}, {"id_usuario": 3, "nombre": "Jhon Diaz", "email": "jhondiaz@mail.com", "password": "passjhon"}, {"id_usuario": 4, "nombre": "Sofia Ramirez", "email": "sofia@mail.com", "password": "passsofia"}, {"id_usuario": 5, "nombre": "Johan Cárdenas", "email": "johan@mail.com", "password": "passjohan"}]
```

## 2. Obtener un usuario por ID

**GET** ✓

/api/usuario/4

The screenshot shows a Postman interface with a 'GET' request to 'http://localhost:8000/api/usuario/4'. The 'Body' tab is selected. The response status is '200 OK', size is '91 Bytes', and time is '6 ms'. The response body is a JSON object:

```
1 {  
2   "id_usuario": 4,  
3   "nombre": "Sofía Ramírez",  
4   "email": "sofia@mail.com",  
5   "password": "passssofia"  
6 }
```

## 3. Crear un usuario nuevo

**POST** ✓

/api/usuario

**Body (JSON):**

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

The screenshot shows a Postman interface with a 'POST' request to 'http://localhost:8000/api/usuario'. The 'Body' tab is selected and set to 'JSON'. The response status is '201 Created', size is '66 Bytes', and time is '12 ms'. The response body is a JSON object:

```
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"  
}
```

The screenshot shows the Postman interface with a successful API call. The request method is PUT, the URL is `http://localhost:8000/api/usuario/4`, and the response status is 200 OK. The JSON response body is identical to the request body provided above.

#### 5. Eliminar un usuario

**DELETE** ✓

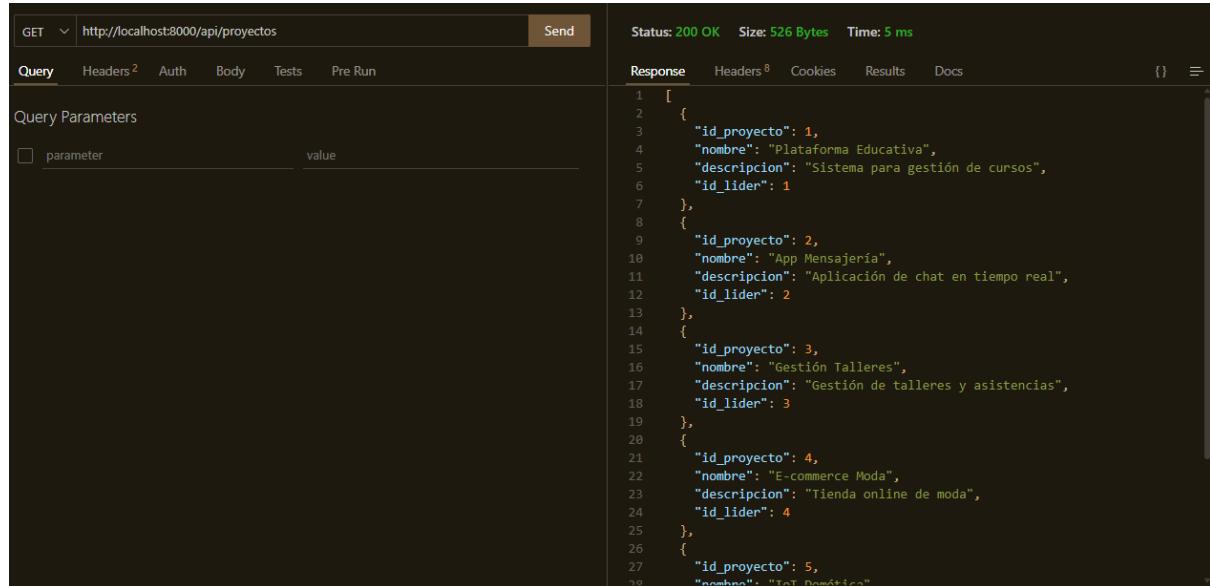
/api/usuario/6

The screenshot shows the Postman interface with a successful API call. The request method is DELETE, the URL is `http://localhost:8000/api/usuario/6`, and the response status is 200 OK. The JSON response body contains a single key "message" with the value "Usuario eliminado".

# Jefferson / Proyecto

## 1. Listar todos proyectos: GET:✓

/api/proyectos



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

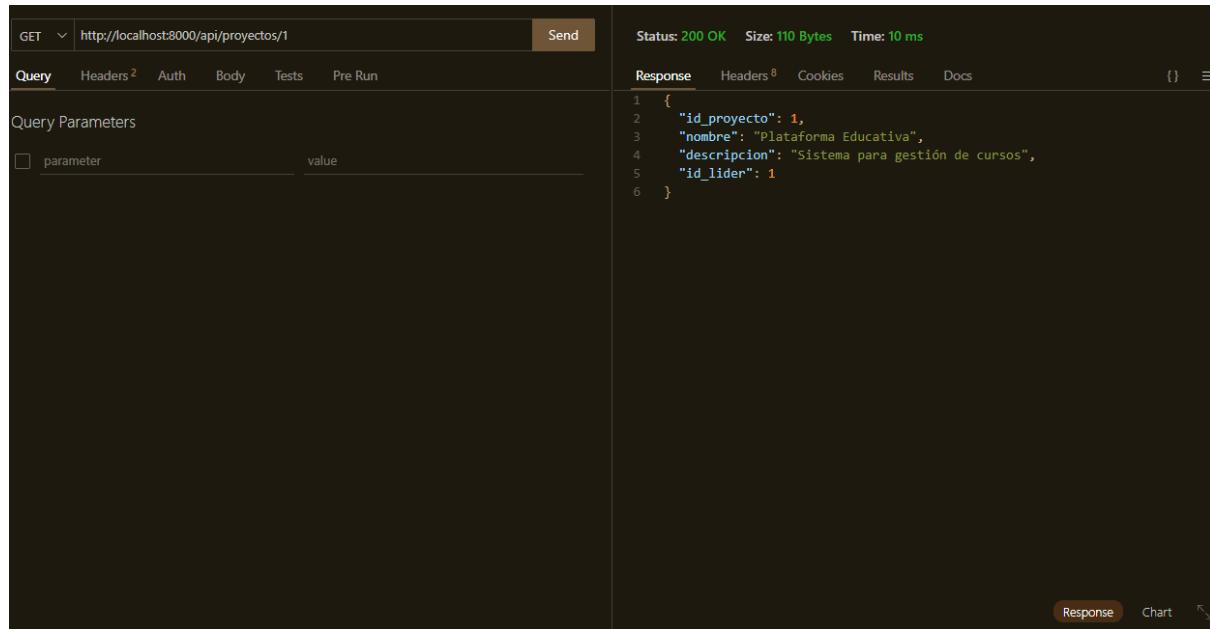
Status: 200 OK Size: 526 Bytes Time: 5 ms

Response Headers Cookies Results Docs

```
[{"id_proyecto": 1, "nombre": "Plataforma Educativa", "descripcion": "Sistema para gestión de cursos", "id_lider": 1}, {"id_proyecto": 2, "nombre": "App Mensajería", "descripcion": "Aplicación de chat en tiempo real", "id_lider": 2}, {"id_proyecto": 3, "nombre": "Gestión Talleres", "descripcion": "Gestión de talleres y asistencias", "id_lider": 3}, {"id_proyecto": 4, "nombre": "E-commerce Moda", "descripcion": "Tienda online de moda", "id_lider": 4}, {"id_proyecto": 5, "nombre": "TecT Domésticas", "descripcion": "Plataforma para el hogar", "id_lider": 5}]
```

## 2. Obtener por id los proyectos: GET✓

api/proyectos/1



GET http://localhost:8000/api/proyectos/1 Send

Status: 200 OK Size: 110 Bytes Time: 10 ms

Response Headers Cookies Results Docs

```
{ "id_proyecto": 1, "nombre": "Plataforma Educativa", "descripcion": "Sistema para gestión de cursos", "id_lider": 1}
```

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

(ejemplo para id\_lider=1)

/api/proyectos/leader/1

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8000/api/proyectos/leader/1`. The response status is **200 OK**, size is **112 Bytes**, and time is **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 the Postman interface with a successful POST request to `http://localhost:8000/api/proyectos`. The response status is **201 Created**, size is **98 Bytes**, and time is **28 ms**. The response body is a JSON object:

```
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

The screenshot shows a POSTMAN interface. The URL is `http://localhost:8000/api/proyectos/6`. The method is `PUT`. The `Body` tab is selected, showing a JSON payload:

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

The response status is `200 OK`, size is `103 Bytes`, and time is `9 ms`. The response body is identical to the request body.

## 6. Eliminar proyecto: DELETE✓

/api/proyectos/6

The screenshot shows a POSTMAN interface. The URL is `http://localhost:8000/api/proyectos/6`. The method is `DELETE`. The `Body` tab is selected, showing a JSON payload:

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

The response status is `200 OK`, size is `103 Bytes`, and time is `9 ms`. The response body is identical to the request body.

# Johan / historia\_usuario & tarea

## 1. Crear nueva historia:POST✓

/api/historias

**Body (JSON):**

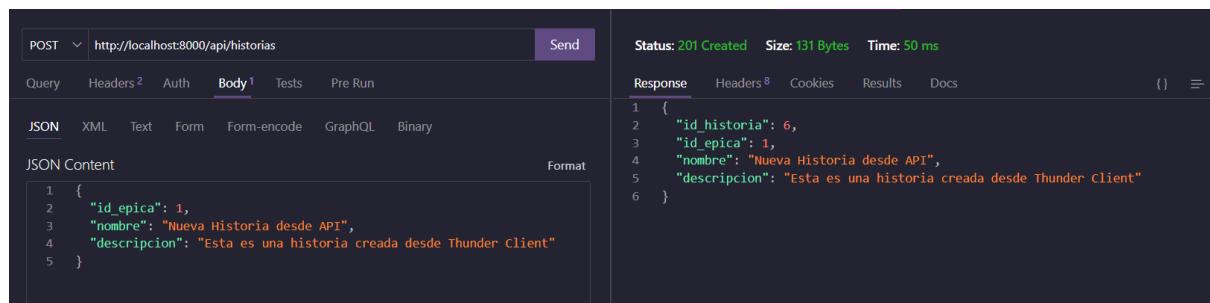
{

"id\_epica": 1,

"nombre": "Nueva Historia desde API",

"descripcion": "Esta es una historia creada desde Thunder Client"

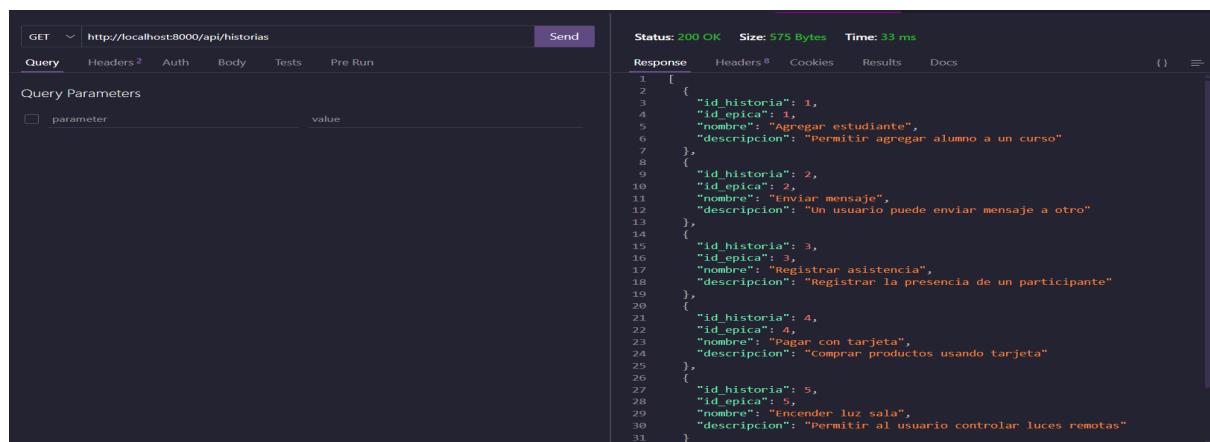
}



```
POST http://localhost:8000/api/historias
Status: 201 Created Size: 131 Bytes Time: 50 ms
Response Headers 8 Cookies Results Docs
1 {
2   "id_historia": 6,
3   "id_epica": 1,
4   "nombre": "Nueva Historia desde API",
5   "descripcion": "Esta es una historia creada desde Thunder Client"
6 }
```

## 2. Listar todas las historias:GET✓

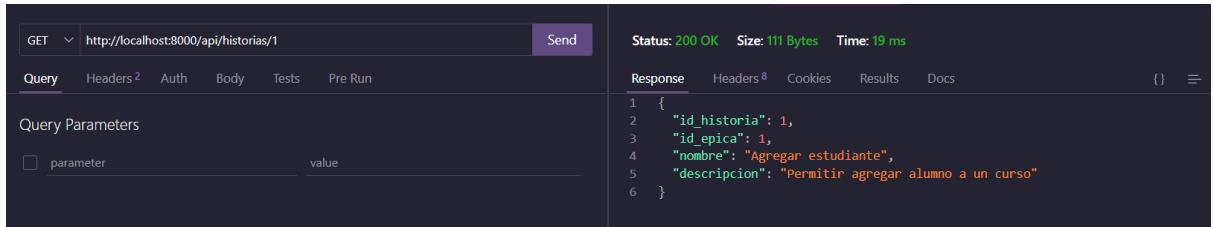
/api/historias



```
GET http://localhost:8000/api/historias
Status: 200 OK Size: 575 Bytes Time: 33 ms
Response Headers 8 Cookies Results Docs
1 [
2   {
3     "id_historia": 1,
4     "id_epica": 1,
5     "nombre": "Agregar estudiante",
6     "descripcion": "Permitir agregar alumno a un curso"
7   },
8   {
9     "id_historia": 2,
10    "id_epica": 2,
11    "nombre": "Enviar mensaje",
12    "descripcion": "Un usuario puede enviar mensaje a otro"
13  },
14  {
15    "id_historia": 3,
16    "id_epica": 3,
17    "nombre": "Registrar asistencia",
18    "descripcion": "Registrar la presencia de un participante"
19  },
20  {
21    "id_historia": 4,
22    "id_epica": 4,
23    "nombre": "Pagar con tarjeta",
24    "descripcion": "Comprar productos usando tarjeta"
25  },
26  {
27    "id_historia": 5,
28    "id_epica": 5,
29    "nombre": "Encender luz sala",
30    "descripcion": "Permitir al usuario controlar luces remotas"
31 }
```

### 3. Obtener historia por id:GET

/api/historias/{id}



GET <http://localhost:8000/api/historias/1> Send

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

Query Parameters

parameter value

Status: 200 OK Size: 111 Bytes Time: 19 ms

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

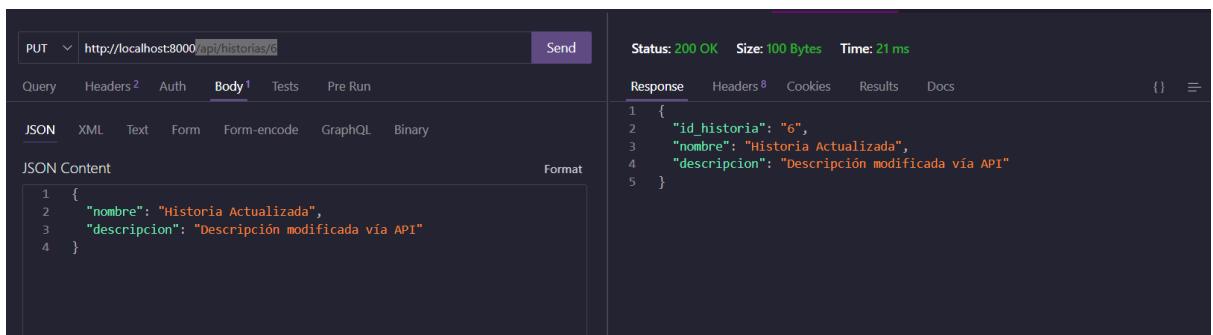
```
1 {  
2   "id_historia": 1,  
3   "id_epica": 1,  
4   "nombre": "Agregar estudiante",  
5   "descripcion": "Permitir agregar alumno a un curso"  
6 }
```

### 4. Actualizar historia:PUT

/api/historias/{id}

**Body (JSON):**

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



PUT <http://localhost:8000/api/historias/6> 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

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

Status: 200 OK Size: 100 Bytes Time: 21 ms

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

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

## 5. Eliminar historia:DELETE

/api/historias/{id}

Status: 200 OK Size: 43 Bytes Time: 16 ms

Response Headers 8 Cookies Results Docs

```
1 {  
2   "message": "Historia de usuario eliminada"  
3 }
```

## 1. Crear nueva tarea:POST

/api/tareas

**Body (JSON):**

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

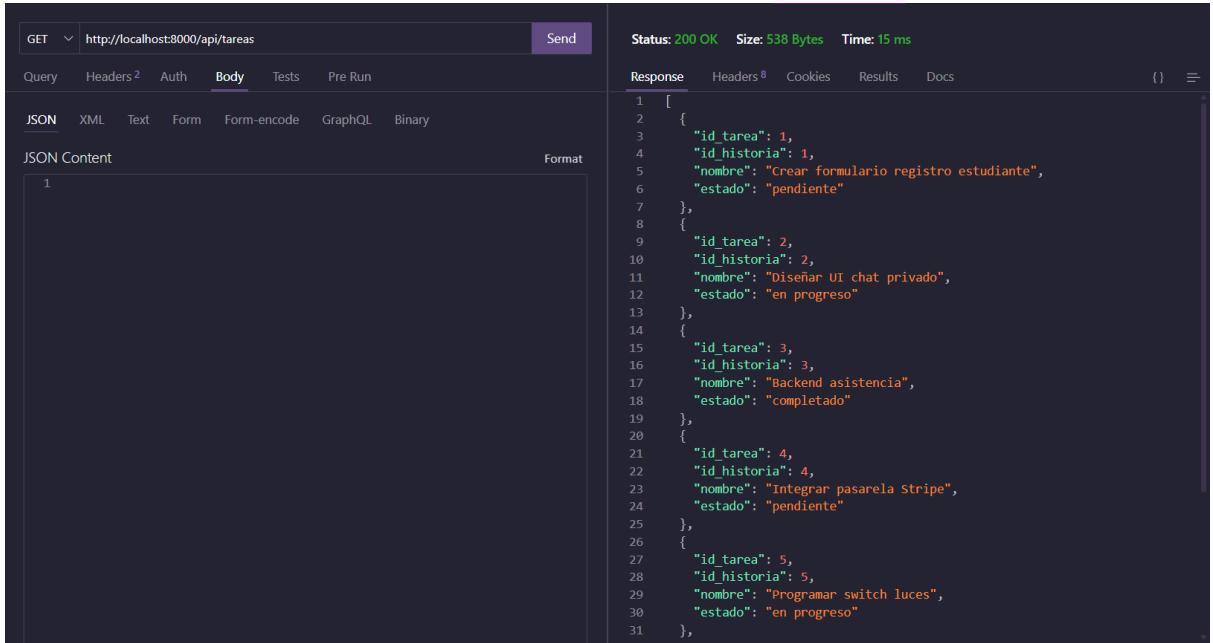
Status: 201 Created Size: 84 Bytes Time: 14 ms

Response Headers 8 Cookies Results Docs

```
1 {  
2   "id_tarea": 7,  
3   "id_historia": 1,  
4   "nombre": "Nueva Tarea de Prueba",  
5   "estado": "pendiente"  
6 }
```

## 2. Listar todas las tareas:GET✓

/api/tareas

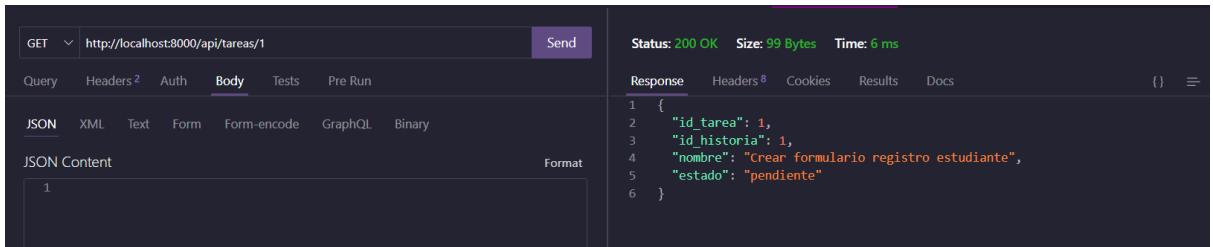


The screenshot shows a Postman request for a GET request to `http://localhost:8000/api/tareas`. The response status is 200 OK, size is 538 Bytes, and time is 15 ms. The response body is a JSON array containing five task objects:

```
[{"id_tarea": 1, "id_historia": 1, "nombre": "Crear formulario registro estudiante", "estado": "pendiente"}, {"id_tarea": 2, "id_historia": 2, "nombre": "Diseñar UI chat privado", "estado": "en progreso"}, {"id_tarea": 3, "id_historia": 3, "nombre": "Backend asistencia", "estado": "completado"}, {"id_tarea": 4, "id_historia": 4, "nombre": "Integrar pasarela Stripe", "estado": "pendiente"}, {"id_tarea": 5, "id_historia": 5, "nombre": "Programar switch luces", "estado": "en progreso"}]
```

## 3. Obtener una tarea por ID:GET✓

/api/tareas/{id}



The screenshot shows a Postman request for a GET request to `http://localhost:8000/api/tareas/1`. The response status is 200 OK, size is 99 Bytes, and time is 6 ms. The response body is a JSON object representing the first task:

```
{ "id_tarea": 1, "id_historia": 1, "nombre": "Crear formulario registro estudiante", "estado": "pendiente" }
```

#### 4. Actualizar tarea:PUT✓

/api/tareas/{id}

**Body (JSON):**

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

The screenshot shows a Postman request for a PUT operation at `http://localhost:8000/api/tareas/7`. The request body is a JSON object with three fields: `nombre`, `estado`, and `idHistoria`. The response status is 200 OK, size is 83 Bytes, and time is 14 ms. The response body is identical to the request body.

```
PUT http://localhost:8000/api/tareas/7  
Content-Type: application/json  
  
{"id_tarea": "7", "nombre": "Tarea Modificada", "estado": "en progreso", "id_historia": 1}  
  
Status: 200 OK Size: 83 Bytes Time: 14 ms  
Response Headers: Content-Type: application/json  
Content: {"id_tarea": "7", "nombre": "Tarea Modificada", "estado": "en progreso", "id_historia": 1}
```

#### 5. Eliminar tarea:DELETE✓

/api/tareas/{id}

The screenshot shows a Postman request for a DELETE operation at `http://localhost:8000/api/tareas/7`. The request body is a JSON object with one field: `message`. The response status is 200 OK, size is 29 Bytes, and time is 12 ms. The response body contains the message "Tarea eliminada".

```
DELETE http://localhost:8000/api/tareas/7  
Content-Type: application/json  
  
{"message": "Tarea eliminada"}  
  
Status: 200 OK Size: 29 Bytes Time: 12 ms  
Response Headers: Content-Type: application/json  
Content: {"message": "Tarea eliminada"}
```

## Jhon / usuario\_proyecto & usuario\_tarea

usuario\_proyecto

### 1. Asignar usuario a proyecto: POST

api/usuario-proyectos

Status: 201 Created Size: 54 Bytes Time: 32 ms

Response Headers 8 Cookies Results Docs

```
1 {  
2   "id_usuario": 6,  
3   "id_proyecto": 6,  
4   "rol": "desarrollador"  
5 }
```

### 2. Quitar usuario de proyecto: DELETE

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

Ejemplo:

/api/usuario-proyectos/6/6

Status: 200 OK Size: 43 Bytes Time: 20 ms

Response Headers 8 Cookies Results Docs

```
1 {  
2   "message": "Usuario removido del proyecto"  
3 }
```

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

api/usuario-proyectos/project/2

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

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

Response Headers: Content-Type: application/json; charset=utf-8

```
[{"id_usuario": 1, "nombre": "Mariana Pérez", "email": "mariana@mail.com", "password": "passmariana"}, {"id_usuario": 2, "nombre": "Jefferson Castro", "email": "jefferson@mail.com", "password": "passjefferson"}, {"id_usuario": 3, "nombre": "Jhon Diaz", "email": "jhondiaz@mail.com", "password": "passjhon"}, {"id_usuario": 4, "nombre": "Sofía Ramírez", "email": "sofia@mail.com", "password": "passsofia"}, {"id_usuario": 5, "nombre": "Tatiana Cárdenas"}]
```

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

/api/usuario-proyectos/user/1

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

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

Response Headers: Content-Type: application/json; charset=utf-8

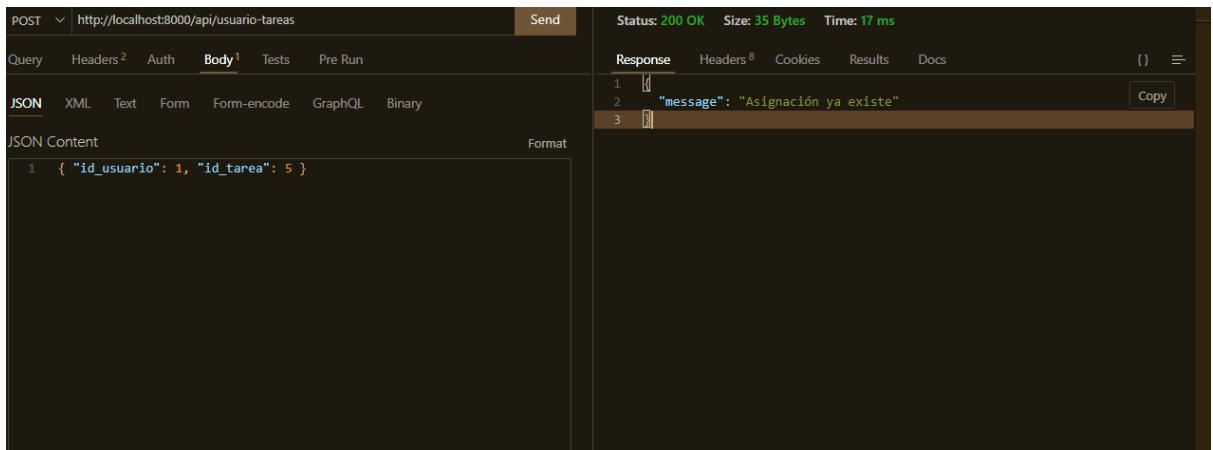
```
[{"id_proyecto": 1, "nombre": "Plataforma Educativa", "descripcion": "Sistema para gestión de cursos", "id_lider": 1}, {"id_proyecto": 2, "nombre": "App Mensajería", "descripcion": "Aplicación de chat en tiempo real", "id_lider": 2}, {"id_proyecto": 3, "nombre": "Gestión Talleres", "descripcion": "Gestión de talleres y asistencias", "id_lider": 3}, {"id_proyecto": 4, "nombre": "E-commerce Moda", "descripcion": "Tienda online de moda", "id_lider": 4}, {"id_proyecto": 5, "nombre": "TecT Domótica"}]
```

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 }



POST <http://localhost:8000/api/usuario-tareas> 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

```
1 { "id_usuario": 1, "id_tarea": 5 }
```

Status: 200 OK Size: 35 Bytes Time: 17 ms

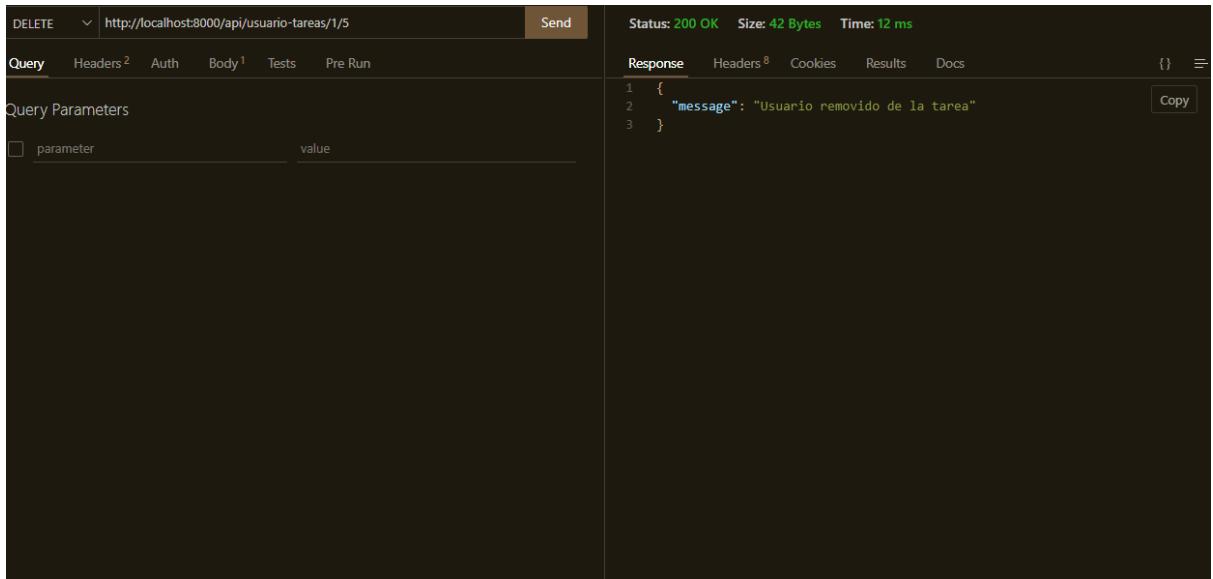
Response Headers<sup>8</sup> Cookies Results Docs

```
1 {}
2   "message": "Asignación ya existe"
3 []
```

Copy

## 2. Quitar usuario de tarea :DELETE

/api/usuario-tareas/1/5



DELETE <http://localhost:8000/api/usuario-tareas/1/5> Send

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

Query Parameters

parameter value

Status: 200 OK Size: 42 Bytes Time: 12 ms

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

```
1 {}
2   "message": "Usuario removido de la tarea"
3 }
```

Copy

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

/api/usuario-tareas/task/5

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

Status: 200 OK Size: 93 Bytes Time: 11 ms

Response Headers: Content-Type: application/json

```
[{"id_usuario": 5, "nombre": "Johan C\u00f3rdenas", "email": "johan@mail.com", "password": "passjohan"}]
```

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

/api/usuario-tareas/user/1

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

Status: 200 OK Size: 101 Bytes Time: 5 ms

Response Headers: Content-Type: application/json

```
[{"id_tarea": 1, "id_historia": 1, "nombre": "Crear formulario registro estudiante", "estado": "pendiente"}]
```