

# **Dockerización de API y BBDD**

Programación de Servicios y Procesos

**Adrián Condines Celada**

Aula Estudio

2º Ciclo Superior - Desarrollo de Aplicaciones Multiplataforma

Curso 2025 - 2026

# Índice

<b>1. INTRODUCCIÓN Y OBJETIVOS</b>	<b>2</b>
<b>2. API</b>	<b>2</b>
2.1. Listar todos los usuarios . . . . .	2
2.2. Listar un usuario en concreto . . . . .	3
2.3. Añadir un usuario . . . . .	4
2.4. Eliminar un usuario . . . . .	5
2.5. Cambiar el nombre de un usuario . . . . .	6
2.6. Listar todos los grupos . . . . .	7
2.7. Listar un grupo en concreto . . . . .	8
2.8. Añadir un grupo . . . . .	9
2.9. Eliminar un grupo . . . . .	10
2.10. Añadir un usuario a un grupo . . . . .	11
2.11. Eliminar un usuario a un grupo . . . . .	12
<b>3. ESTRUCTURA DEL PROYECTO</b>	<b>12</b>
<b>4. PROCESO DE DOCKERIZACIÓN</b>	<b>13</b>
<b>5. CI/CD CON GITHUB ACTIONS</b>	<b>13</b>

# 1. INTRODUCCIÓN Y OBJETIVOS

## 2. API

Esta sección detalla los endpoints disponibles en la API para gestionar los usuarios y los grupos.

### 2.1. Listar todos los usuarios

#### 2.1.1. Descripción

Obtiene una lista de todos los registros almacenados en la tabla `users`.

- **Método:** GET
- **URL:** /api/usuarios

#### 2.1.2. Ejemplo de Input

No requiere ningún input (ni parámetros en la URL ni body).

#### 2.1.3. Ejemplo de Output (Éxito)

Responde con un array de objetos JSON, cada uno representando un rapero.

```
[  
  {  
    "id": 1,  
    "name": "Bruno"  
  },  
  {  
    "id": 2,  
    "name": "Adriano"  
  }  
]
```

#### **2.1.4. Ejemplo de Output (Error)**

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al devolver los datos de los usuarios: (error)"  
}
```

### **2.2. Listar un usuario en concreto**

#### **2.2.1. Descripción**

Obtiene un solo registro en la tabla users.

- **Método:** GET
- **URL:** /api/usuarios/:id

#### **2.2.2. Ejemplo de Input**

Parámetro id en la URL.

#### **2.2.3. Ejemplo de Output (Éxito)**

Responde con un JSON en el que se incluye el usuario a buscar.

```
[  
  {  
    "id": 1,  
    "name": "Bruno"  
  }  
]
```

## 2.2.4. Ejemplo de Output (Error)

```
\justify
Responde con un JSON incluyendo un mensaje de error.

{
  "error": "Error al devolver los datos del usuario: (error)"
}
```

## 2.3. Añadir un usuario

### 2.3.1. Descripción

Añade un registro a la tabla users.

- **Método:** POST
- **URL:** /api/usuarios

### 2.3.2. Ejemplo de Input

Un objeto JSON en el body de la petición con el nombre del nuevo usuario.

```
{
  "name": "Adriano"
}
```

### 2.3.3. Ejemplo de Output (Éxito)

Responde con un mensaje de confirmación y un estado 201.

```
{
  "success": "Los datos se han insertado correctamente"
}
```

#### **2.3.4. Ejemplo de Output (Error)**

```
\justify
Responde con un JSON incluyendo un mensaje de error.

{
  "error": "Error al añadir el usuario: (error)"
}
```

### **2.4. Eliminar un usuario**

#### **2.4.1. Descripción**

Elimina un registro de la tabla users.

- **Método:** DELETE
- **URL:** /api/usuarios/:id

#### **2.4.2. Ejemplo de Input**

Parámetro `id` en la URL.

#### **2.4.3. Ejemplo de Output (Éxito)**

Responde con un mensaje de confirmación y un estado 201.

```
{
  "success": "Los datos se han insertado correctamente"
}
```

#### **2.4.4. Ejemplo de Output (Error)**

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al añadir el usuario: (error)"  
}
```

## 2.5. Cambiar el nombre de un usuario

### 2.5.1. Descripción

Modifica el campo **name** de un registro de la tabla `users`.

- **Método:** PUT
- **URL:** /api/usuarios/:id

### 2.5.2. Ejemplo de Input

Parámetro `id` en la URL y un objeto JSON en el body de la petición con el nombre del nuevo usuario.

```
{  
  "name": "Adriano"  
}
```

### 2.5.3. Ejemplo de Output (Éxito)

Responde con un mensaje de confirmación y un estado 200.

```
{  
  "success": "Se ha modificado registro correctamente"  
}
```

#### **2.5.4. Ejemplo de Output (Error)**

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al modificar el nombre del usuario: (error)"  
}
```

### **2.6. Listar todos los grupos**

#### **2.6.1. Descripción**

Obtiene una lista de todos los registros almacenados en la tabla groups.

- **Método:** GET
- **URL:** /api/grupos

#### **2.6.2. Ejemplo de Input**

No requiere ningún input (ni parámetros en la URL ni body).

#### **2.6.3. Ejemplo de Output (Éxito)**

Responde con un array de objetos JSON, cada uno representando un rapero.

```
[  
  {  
    "id": 1,  
    "name": "Aulaestudienses"  
  },  
  {  
    "id": 2,  
    "name": "Antijavanianos"  
  }  
]
```

## 2.6.4. Ejemplo de Output (Error)

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al devolver los datos de los grupos: (error)"  
}
```

## 2.7. Listar un grupo en concreto

### 2.7.1. Descripción

Obtiene un solo registro en la tabla groups.

- **Método:** GET
- **URL:** /api/grupos/:id

### 2.7.2. Ejemplo de Input

Parámetro id en la URL.

### 2.7.3. Ejemplo de Output (Éxito)

Responde con un JSON en el que se incluye el grupo a buscar.

```
[  
  {  
    "id": 1,  
    "name": "Aulaestudienses"  
  }  
]
```

## 2.7.4. Ejemplo de Output (Error)

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al devolver los datos del usuario: (error)"  
}
```

## 2.8. Añadir un grupo

### 2.8.1. Descripción

Añade un registro a la tabla users.

- **Método:** POST
- **URL:** /api/grupos

### 2.8.2. Ejemplo de Input

Un objeto JSON en el body de la petición con el nombre del nuevo grupo.

```
{  
  "name": "Aulaestudienses"  
}
```

### 2.8.3. Ejemplo de Output (Éxito)

Responde con un mensaje de confirmación y un estado 200.

```
{  
  "success": "Los datos se han insertado correctamente"  
}
```

#### **2.8.4. Ejemplo de Output (Error)**

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al añadir el usuario: (error)"  
}
```

### **2.9. Eliminar un grupo**

#### **2.9.1. Descripción**

Elimina un registro de la tabla groups.

- **Método:** DELETE
- **URL:** /api/grupos/:id

#### **2.9.2. Ejemplo de Input**

Parámetro id en la URL.

#### **2.9.3. Ejemplo de Output (Éxito)**

Responde con un mensaje de confirmación y un estado 201.

```
{  
  "success": "Grupo eliminado correctamente"  
}
```

#### **2.9.4. Ejemplo de Output (Error)**

Responde con un JSON incluyendo un mensaje de error.

```
{  
  "error": "Error al eliminar el grupo: (error)"
```

```
}
```

## 2.10. Añadir un usuario a un grupo

### 2.10.1. Descripción

Añade un registro a la tabla users\_groups.

- **Método:** POST
- **URL:** /api/grupos/:id\_grupo/:id\_usuario

### 2.10.2. Ejemplo de Input

Parámetros id\_grupo y id\_usuario en la URL.

### 2.10.3. Ejemplo de Output (Éxito)

Responde con un mensaje de confirmación y un estado 200.

```
{
  "success": "Los datos se han insertado correctamente"
}
```

### 2.10.4. Ejemplo de Output (Error)

Responde con un JSON incluyendo un mensaje de error.

```
{
  "error": "Error al añadir el usuario al grupo: (error)"
}
```

## 2.11. Eliminar un usuario a un grupo

### 2.11.1. Descripción

Elimina un registro de la tabla users\_groups.

- **Método:** DELETE
- **URL:** /api/grupos/:id\_grupo/:id\_usuario

### 2.11.2. Ejemplo de Input

Parámetros id\_grupo y id\_usuario en la URL.

### 2.11.3. Ejemplo de Output (Éxito)

Responde con un mensaje de confirmación y un estado 200.

```
{  
    "success": "Usuario eliminado del grupo correctamente"  
}
```

### 2.11.4. Ejemplo de Output (Error)

Responde con un JSON incluyendo un mensaje de error.

```
{  
    "error": "Error al añadir el usuario al grupo: (error)"  
}
```

## 3. ESTRUCTURA DEL PROYECTO

En esta sección se define la estructura del proyecto:

```
api_usuarios_grupos/
|-- db/
|-- DOCUMENTACIÓN/
|   |-- images/
|   |-- DOCUMENTACIÓN API Y BBDD - ADRIANO.tex
|   |-- DOCUMENTACIÓN API Y BBDD - ADRIANO.pdf
|-- .dockerignore
|-- .env
|-- docker-compose.yaml
|-- Dockerfile
|-- index.js
|-- package-lock.json
|-- package.json
```

## 4. PROCESO DE DOCKERIZACIÓN

## 5. CI/CD CON GITHUB ACTIONS