

Examen U2: Guaman Jhon

Ejercicio 2: API RESTful para Gestión de Tareas (To-Do)

Este ejercicio se enfoca en la correcta implementación de una API RESTful y la aplicación rigurosa de una arquitectura por capas.

Descripción del Ejercicio

Debes desarrollar una API RESTful completa para gestionar una lista de tareas (To-Do). La API debe permitir a los usuarios registrarse, iniciar sesión y luego realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) sobre sus propias tareas. Un usuario **no** debe poder ver o modificar las tareas de otros usuarios.

Requisitos Técnicos

- **Backend:** Node.js con Express.
- **Base de Datos:** MongoDB con Mongoose.
- **Autenticación:** JWT (JSON Web Tokens).
- **Arquitectura:** Aplicar principios de Arquitectura Limpia (separación de capas). El foco principal es este punto.

Requisitos Funcionales

1. Autenticación de Usuarios:

- Crear un endpoint POST `/api/auth/register` para registrar un nuevo usuario (nombre, email y password). La contraseña debe guardarse hasheada.
- Crear un endpoint POST `/api/auth/login` que valide las credenciales y devuelva un JWT.

2. Operaciones CRUD de Tareas (Endpoints Protegidos):

- POST `/api/tasks`: Crear una nueva tarea. El body debe contener el title y opcionalmente una description. La tarea debe quedar asociada al usuario autenticado.
- GET `/api/tasks`: Obtener **únicamente** las tareas del usuario autenticado.
- PUT `/api/tasks/:id`: Actualizar una tarea existente (ej. cambiar título, descripción o marcar como completada). Solo el dueño de la tarea puede actualizarla.
- DELETE `/api/tasks/:id`: Eliminar una tarea. Solo el dueño de la tarea puede eliminarla.

3. Modelos de Datos (Mongoose):

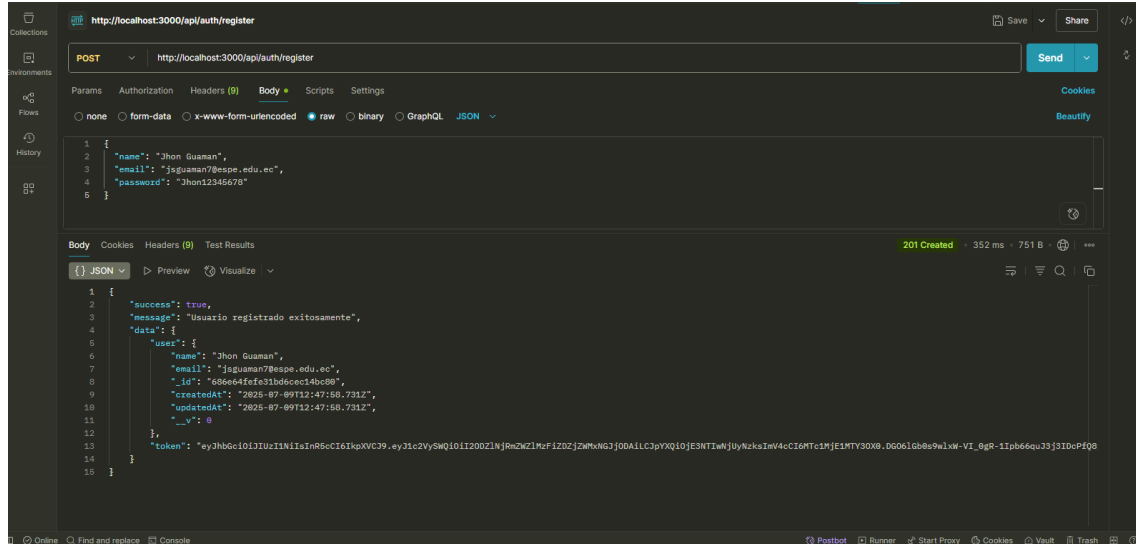
- **User:** name, email, password.
- **Task:** title, description, completed (boolean, default: false), user (referencia al modelo User).

Estructura de Proyecto Sugerida (Arquitectura Limpia)

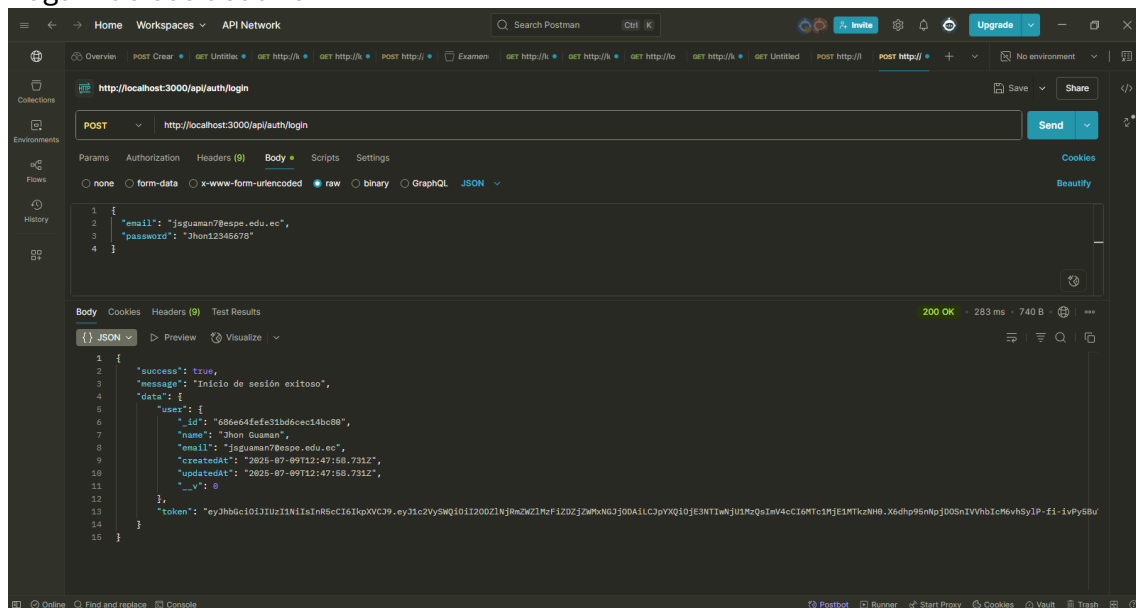
```
/src
|-- /api
|   |-- /routes      # Define las rutas de la API (auth.routes.js, task.routes.js)
|   |-- /controllers  # Controladores (req, res) (auth.controller.js, task.controller.js)
|-- /domain
|   |-- /models      # Esquemas de Mongoose (user.model.js, task.model.js)
|   |-- /use-cases    # Lógica de negocio (create-task.use-case.js, get-user-tasks.use-case.js)
|-- /infrastructure
|   |-- /repositories # Lógica de acceso a datos (user.repository.js, task.repository.js)
|   |-- /middlewares  # Middlewares de Express (auth.middleware.js)
```

```
|-- /config      # Configuración (db, variables de entorno)
|-- app.js       # Archivo principal del servidor Express
```

Registro de un usuario



Login de ese usuario



Visualizacion sin token

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/auth/profile`. The response is a 401 Unauthorized status with a 9 ms response time and 405 B of data. The response body is in JSON format:

```
{
  "success": false,
  "message": "Acceso denegado. No se proporcionó token de autorización"
}
```

The interface includes tabs for Params, Auth, Headers (9), Body, Scripts, and Settings. The Body tab is selected, showing the JSON response.

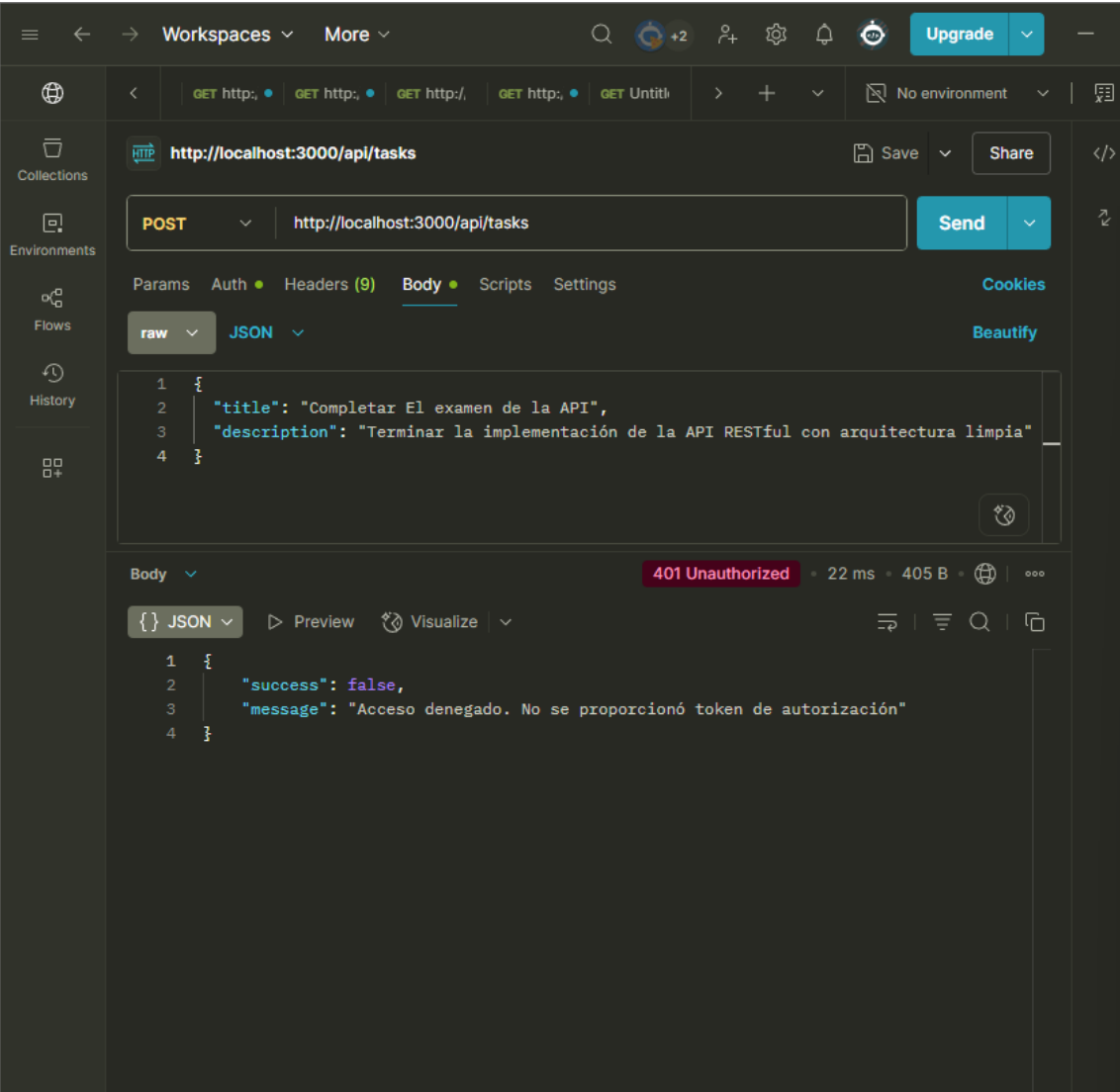
Con el token para permisos

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/auth/profile`. The request is configured with a Bearer token in the Authorization header. The response is a 200 OK status with a 33 ms response time and 555 B of data. The response body is in JSON format:

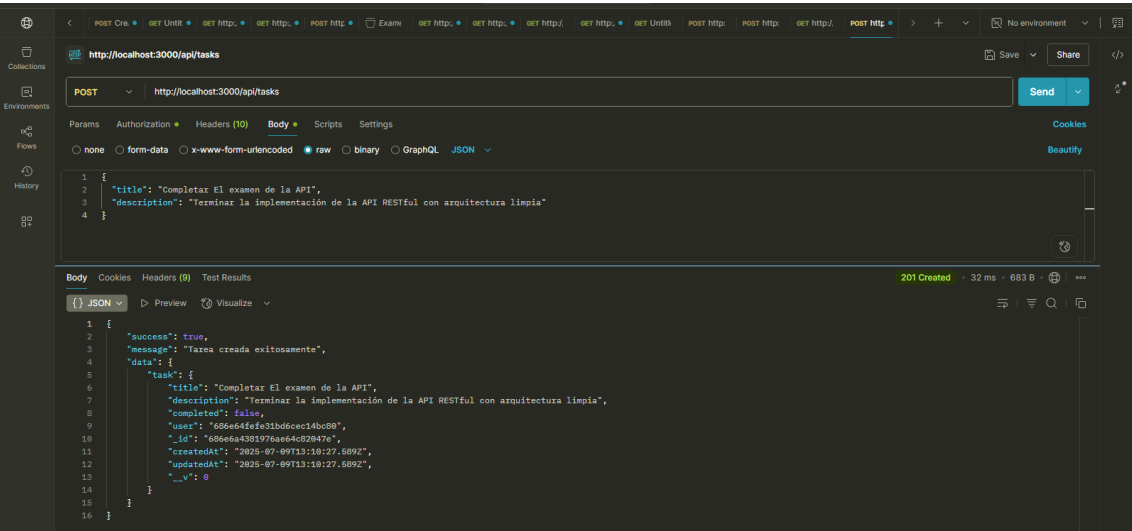
```
{
  "success": true,
  "message": "Perfil obtenido exitosamente",
  "data": {
    "user": {
      "_id": "686e64fe31bd6ceci4bc88",
      "name": "Jhon Guaman",
      "email": "jrguaman78@ape.edu.ec",
      "createdAt": "2025-07-09T12:47:58.731Z",
      "updatedAt": "2025-07-09T12:47:58.731Z",
      "__v": 0
    }
  }
}
```

The interface includes tabs for Params, Authorization, Headers (10), Body, Scripts, and Settings. The Authorization tab is selected, showing the Bearer token. The Body tab is also selected, showing the JSON response.

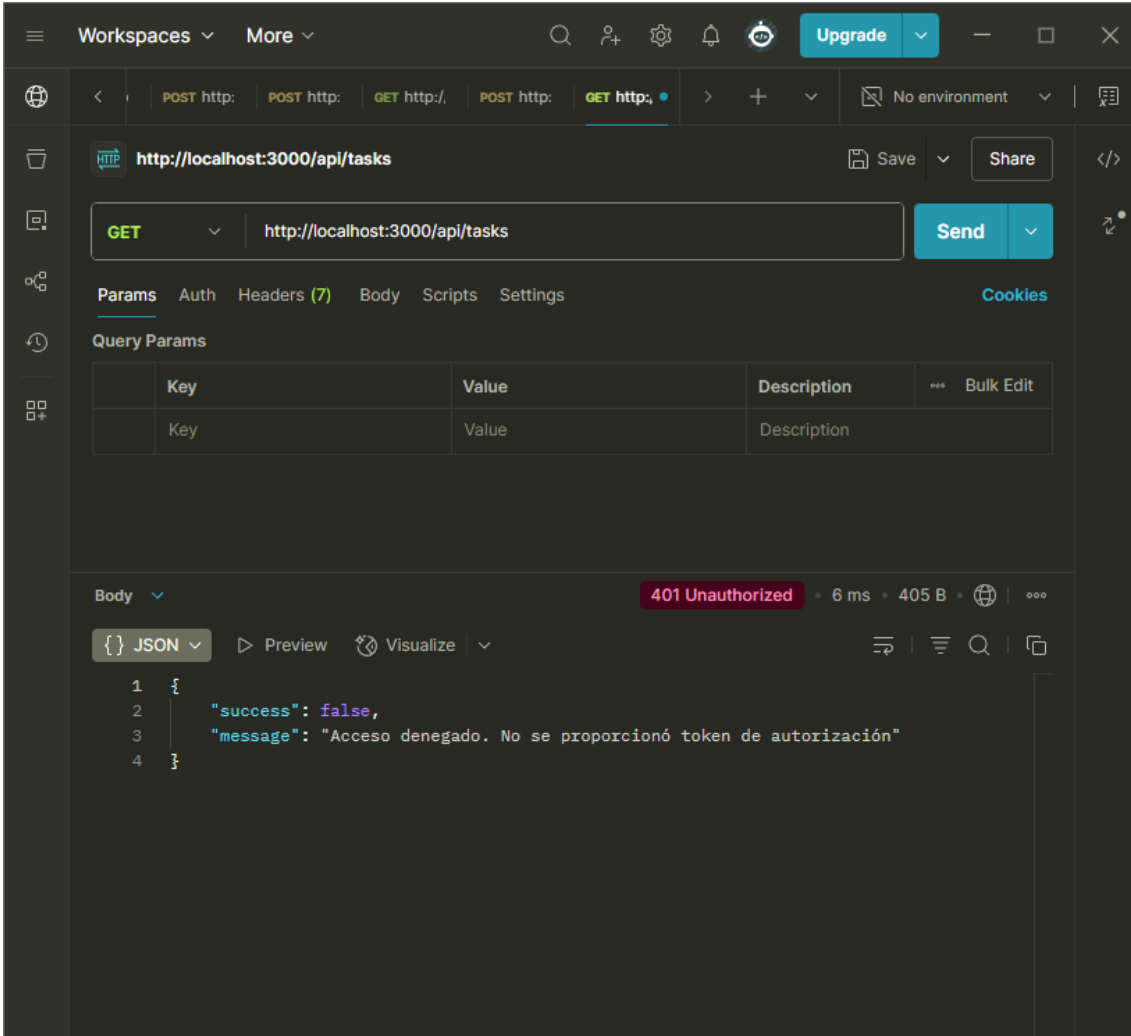
Crear una tarea Sin el token



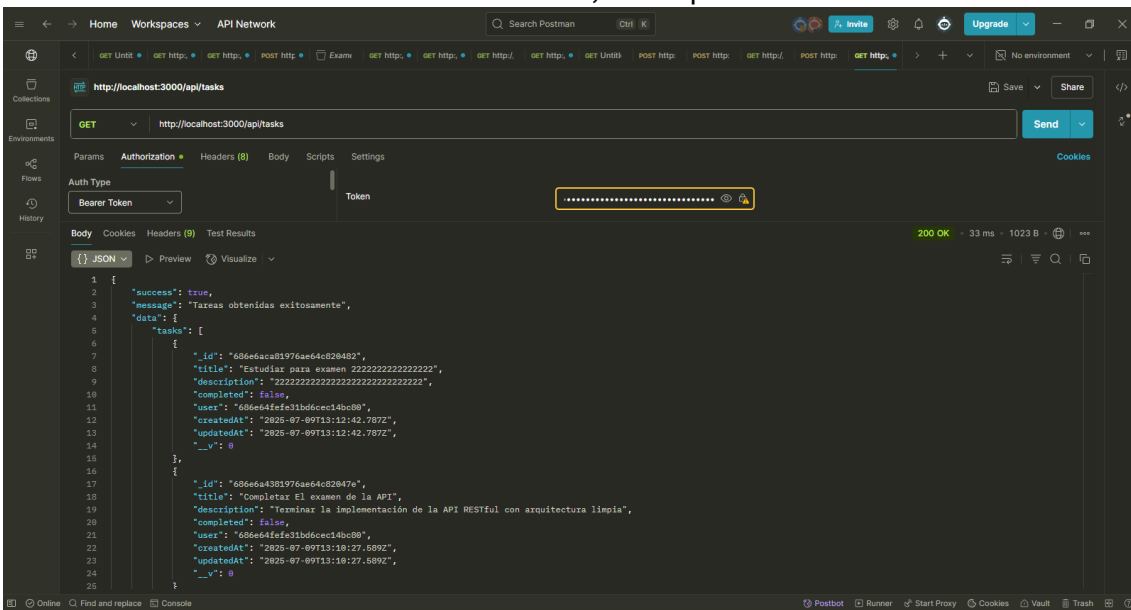
Con el token



Obtener tareas sin el token



Obtener tareas con el token de ese usuario, solo aparecerá las de ese usuario



Obtener Tarea Específica

Workspaces More

GET http://localhost:3000/api/tasks/686e6aca81976ae64c820482

GET http://localhost:3000/api/tasks/686e6aca81976ae64c820482

Params Auth Headers (8) Body Scripts Settings

raw JSON

1 Ctrl+Alt+P for Postbot

Body 200 OK 10 ms 648 B

```
{
  "success": true,
  "message": "Tarea obtenida exitosamente",
  "data": {
    "task": {
      "_id": "686e6aca81976ae64c820482",
      "title": "Estudiar para examen 2222222222222222",
      "description": "22222222222222222222222222222222",
      "completed": false,
      "user": "686e64fefe31bd6cec14bc80",
      "createdAt": "2025-07-09T13:12:42.787Z",
      "updatedAt": "2025-07-09T13:12:42.787Z",
      "__v": 0
    }
  }
}
```

Actualizar Tarea sin token

PUT http://localhost:3000/api/tasks/686e6aca81976ae64c820482

PUT http://localhost:3000/api/tasks/686e6aca81976ae64c820482

Params Auth Headers (9) Body Scripts Settings

raw JSON

```
{
  "title": "EXAMEN ACTUALIZAD000 FINALIZADO",
  "description": "ACTUALIZAD000000",
  "completed": true
}
```

Body 401 Unauthorized 7 ms 405 B

```
{
  "success": false,
  "message": "Acceso denegado. No se proporcionó token de autorización"
}
```

Actualizar Tarea con token (solo el usuario puede actualizar)

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/api/tasks/686e6aca81976ae64c820482`. The request body is a JSON object:

```
1 {
2   "title": "EXAMEN ACTUALIZAD000 FINALIZADO",
3   "description": "ACTUALIZAD000000",
4   "completed": true
5 }
```

The response is a 200 OK status with a response time of 25 ms and a body size of 632 B. The response body is a JSON object:

```
1 {
2   "success": true,
3   "message": "Tarea actualizada exitosamente",
4   "data": {
5     "task": {
6       "_id": "686e6aca81976ae64c820482",
7       "title": "EXAMEN ACTUALIZAD000 FINALIZADO",
8       "description": "ACTUALIZAD000000",
9       "completed": true,
10      "user": "686e64fefe31bd6cec14bc80",
11      "createdAt": "2025-07-09T13:12:42.787Z",
12      "updatedAt": "2025-07-09T13:23:22.198Z",
13      "__v": 0
14    }
15  }
16 }
```

Intento de eliminar tarea sin tener el token

Workspaces More

GET http:// POST http:// GET http:// PUT http:// DEL http://

Save Share

DELETE http://localhost:3000/api/tasks/686e6aca81976ae64c820482 Send

Params Auth Headers (10) Body Scripts Settings Cookies

Query Params

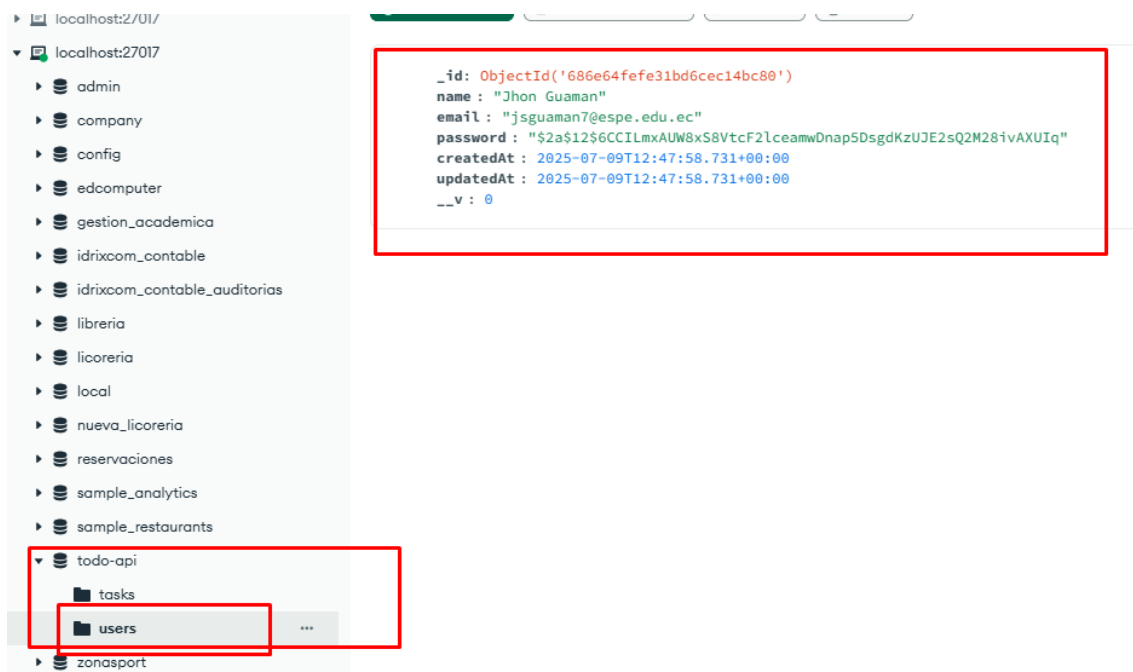
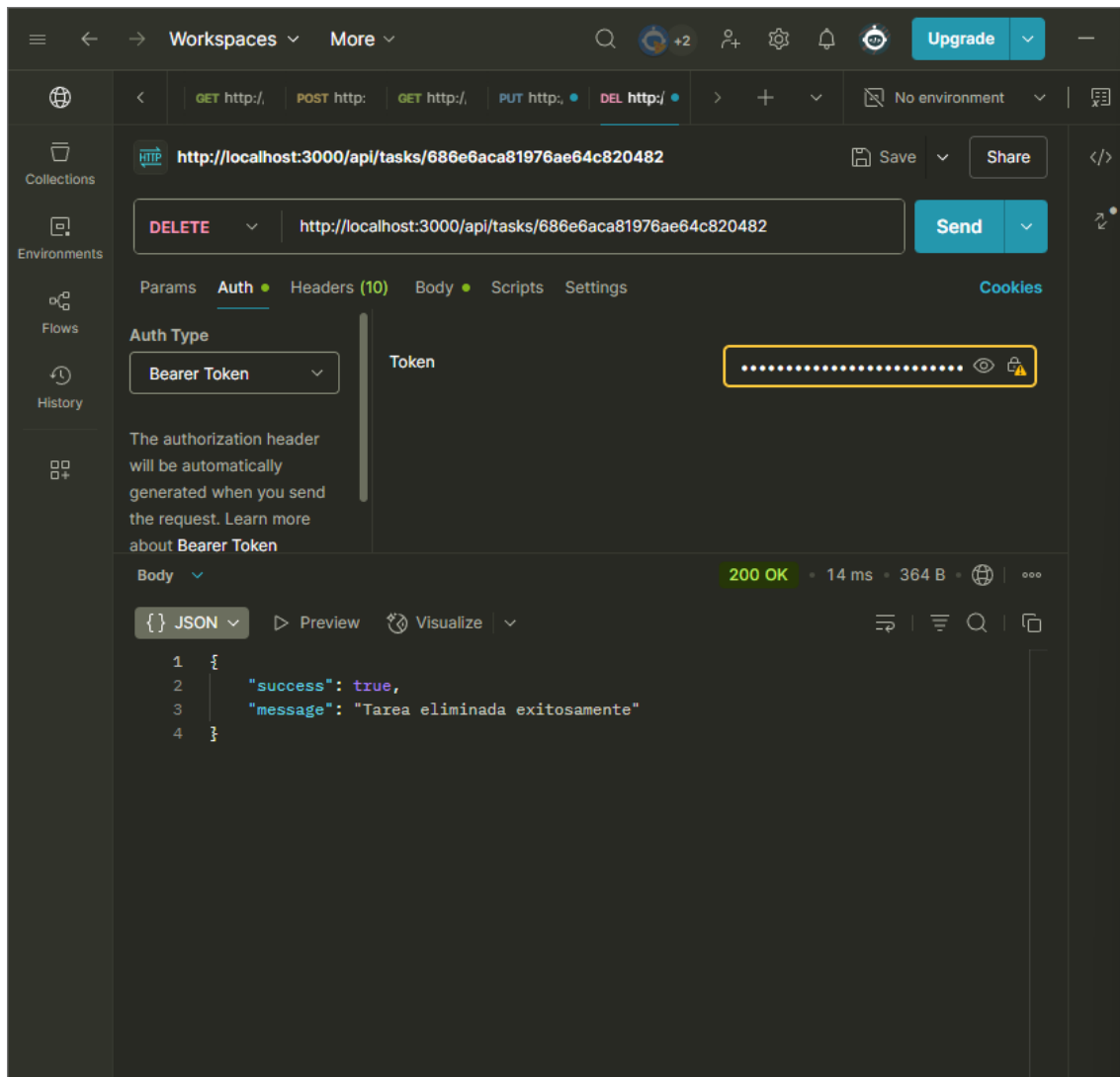
Key	Value	Description	Bulk Edit
Key	Value	Description	

Body 401 Unauthorized 6 ms 388 B

JSON Preview Visualize

```
1 {
2   "success": false,
3   "message": "Token inválido",
4   "error": "Token inválido"
5 }
```

Eliminar tarea con el token(solo el usuario puede eliminarla)



localhost:27017

- admin
- company
- config
- edocomputer
- gestion_academica
- idrixcom_contable
- idrixcom_contable_auditorias
- libreria
- licoreria
- local
- nueva_licoreria
- reservaciones
- sample_analytics
- sample_restaurants
- todo-api
- tasks**

```
_id: ObjectId('686e6a4381976ae64c82047e')
title: "Completar El examen de la API"
description: "Terminar la implementación de la API RESTful con arquitectura limpia"
completed: false
user: ObjectId('686e64fefe31bd6cec14bc80')
createdAt: 2025-07-09T13:10:27.589+00:00
updatedAt: 2025-07-09T13:10:27.589+00:00
__v: 0
```