# Documentación de la API - Backend Restaurante

## URL DEFAULT de la API:

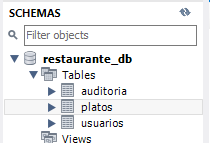
http://localhost:3001/api/

## Configuración inicial 🡪

Para poder empezar a correr nuestro programa, primero debemos setear una base de datos local de MySQL (instalar **server y workbench** utilizando el **MySQL Installer - Community**), y luego, crear una db desde **workbench** de nombre exacto “restaurante\_db”.

Dentro, crearemos las tablas de “platos”, “usuarios” y “auditoria” (Los scripts serán provistos).

Desde el visualizador de esquemas debería verse así:



Luego, para poder visualizar contenido inicial en la página, también se proveerá un script de “bulk insert” para los inserts iniciales.

Dispondremos de un **usuario “inicial”** en caso de no haber ningún otro creado llamado “admin”, con contraseña “admin123”.

\*También se proveerá una colección de postman para probar la API. Los scripts y la colección están en la carpeta “Scripts Iniciales” en el repositorio/proyecto.

## Nota: En el archivo “.env” en la carpeta raíz del back end, están los valores de acceso a la db local (host, usuario, password y nombre de la db)

## Autenticación 🡪

Todas las rutas (excepto /login) requieren un token JWT.

Primero debemos logearnos mediante el endpoint de login antes de poder utilizar los otros.

El token debe enviarse en los headers como:

Authorization: Bearer <token>

## Endpoints disponibles 🡪

### Login:

POST /login

**Body:**

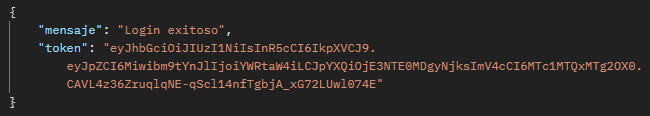
{

  "nombre": "admin",

  "password": "admin123"

}

**Respuesta:**

****

### Administracion de Platos:

**Listar platos:**

GET /platos

Query opcional: nombre, precioMin, precioMax, disponible, categoria, subcategoria.

Respuesta: Array de plato/s.

**Nota**: Por defecto se listarán TODOS los platos, para aplicar filtros, debemos utilizar query strings. Como por ejemplo “/platos?disponible=true”

**Dar de alta un plato:**

POST /platos

**Body:**

{

  "nombre": "Un plato",

  "descripcion": "Una descripcion",

  "alergenos": "Un alergeno",

  "precio": 1651,

  "disponible": 1,

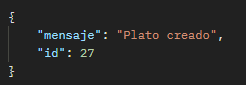
  "categoria": "Al Plato",

  "subcategoria": "Carne blanca"

}

**Nota**: Los campos nombre, precio y categoría son obligatorios. El valor por defecto de disponible será 1 y el de subcategoria “General”.

**Respuesta:**



**Modificar un plato:**

PUT /platos/:id

**Body parcial** (excepto id): puede incluir cualquier campo modificable. Ej.

{

  "nombre": "Papas Fritas",

  "alergenos": "Tomate",

}

**Respuesta:**

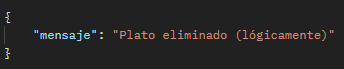


**Eliminar un usuario:**

DELETE /platos/:id

Elimina lógicamente el plato (disponible = false).

**Respuesta:**



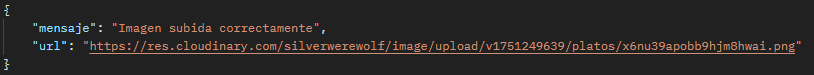
**Dar de alta o modificar la imagen de un plato:**

POST /platos/:id/imagen

Sube una imagen (form-data con campo "foto") a Cloudinary.

**Nota:** Si se utiliza postman debe enviarse un form-data en body con key “foto”, y enviar la imagen a subir.

**Respuesta:**



### Administracion de Usuarios 🡪

**Listar usuarios:**

GET /usuarios

Query opcional: nombre, activo.

Respuesta: Array de usuario/s.

Nota: Por defecto se listarán TODOS los usuarios, para aplicar filtros, debemos utilizar query strings. Como por ejemplo “/usuarios?activo=true”

**Dar de alta un usuario:**

POST /usuarios

**Body:**

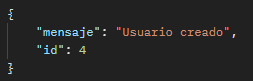
{

  "nombre": "admin",

  "password": "123456"

}

**Respuesta**:



**Modificar un usuario:**

PUT /usuarios/:id

**Body**: Modifica parcialmente nombre o password (está se vuelve a rehashear). Ej.

{

  "nombre": "holaa"

}

**Respuesta:**



**Eliminar un usuario:**

DELETE /usuarios/:id

Elimina lógicamente el usuario (activo = false).

**Respuesta:**



### Auditoría 🡪

Las acciones clave se registran en la tabla auditoria. Se pueden visualizar los registros desde la base de datos luego de realizar alguna acción.

Los campos de importancia son: usuario\_id, acción, entidad, entidad\_id, fecha. Estos corresponden respectivamente al ID del usuario que causó dicha acción, la entidad afectada (un plato u otro usuario, por ejemplo), el id de esa entidad y la fecha de la acción.

Ejemplo 🡪

