19-8-2025

Angel David Reyes Tellez, Luis Ivan marquez Azuara, Brayn kalid reyes silva Aldo Tolentino Domingo

Universidad tecnologica de xicotepec de juarez

Documento de pruebas y evidencia de la API

Fase H – “Hacer” del ciclo PHVA

**Prueba – Crear usuario completo con credenciales**

**Endpoint:** POST /api/usuarios/completo  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/post_api_usuarios_completo>  
**Propósito:** Crear un usuario del sistema y sus credenciales de acceso en una sola operación.

**1) Objetivo de la prueba**

Verificar que el sistema permita registrar un usuario y sus credenciales (correo/contraseña) de forma atómica, devolviendo los IDs creados.

**2) Precondiciones**

* API en ejecución y conexión a BD activa.

## 3) Datos de entrada

**Método:** POST  
**URL:** http://localhost:3000/api/usuarios/completo  
**Headers:**

* Content-Type: application/json

**Body (JSON) usado en la prueba:**

### *{*

### *"identificador": "ADM-018",*

### *"nombre": "Angel",*

### *"apellido": "Reyes",*

### *"teléfono": "+52-81-5555-1235",*

### *"rol": "admin",*

### *"correo": "angel.lopez@eurbana.com",*

### *"password": "Admin123!"*

### *}*

## 4) Solicitud de ejemplo (curl)

### *curl -X POST "http://localhost:3000/api/usuarios/completo" \*

### *-H "Content-Type: application/json" \*

### *-d '{*

### *"identificador": "ADM-018",*

### *"nombre": "Angel",*

### *"apellido": "Reyes",*

### *"telefono": "+52-81-5555-1235",*

### *"rol": "admin",*

### *"correo": "angel.lopez@eurbana.com",*

### *"password": "Admin123!"*

### *}'*

## 5) Salida esperada

**Código HTTP:** 201 Created  
**Respuesta (JSON):**

### *{*

### *"success": true,*

### *"message": "Usuario y credenciales creados exitosamente",*

### *"usuario\_id": "68a216b78442c2285baa94d0",*

### *"auth\_id": "68a216b78442c2285baa94d1"*

### *}*

## 6) Criterios de validación

* Se recibe **201** y success: true.
* usuario\_id y auth\_id tienen **formato de ObjectId**
* El usuario queda **consultable** en GET /api/usuarios y/o GET /api/usuarios/{id}.

## 7) Casos negativos probados / esperados

* **400 Bad Request:** Campos faltantes o formato inválido (ej. password débil o sin correo).
* **409 Conflict:** Correo ya registrado / identificador duplicado.
* **401/403:** Sin token o sin permisos
* **500:** Error interno

## 8) Evidencia

## 

## 9) Trazabilidad a requisitos

* **RF-10** Gestión de usuarios y roles.
* **RNF-06** Control de acceso
* **RNF-18** OWASP (validación de contraseña y datos).

## 10) Observaciones

* La combinación de creación de usuario + credenciales en un solo endpoint **reduce pasos** y evita inconsistencias.

# Prueba – Login de usuario

**Endpoint:** POST /api/auth/login  
**Ruta en Swagger:** [http://localhost:3000/api-docs/#/Autenticación/post\_api\_auth\_login](http://localhost:3000/api-docs/#/Autenticaci%C3%B3n/post_api_auth_login)  
**Propósito:** Validar que un usuario existente pueda iniciar sesión correctamente y obtener un token JWT válido junto con su información básica.

## 1) Objetivo de la prueba

Verificar que el sistema permita a un usuario autenticarse con correo y contraseña válidos, devolviendo un token para acceder a rutas protegidas y datos básicos del perfil.

## 2) Precondiciones

* API en ejecución y conexión a BD activa.
* Usuario previamente registrado con correo y contraseña definidas.

## 3) Datos de entrada

**Método:** POST  
**URL:** http://localhost:3000/api/auth/login  
**Headers:**

* Content-Type: application/json
* Accept: application/json

**Body (JSON) usado en la prueba:**

### *{*

### *"correo": "angel.lopez@eurbana.com",*

### *"password": "Admin123!"*

### *}*

## 4) Solicitud de ejemplo (curl)

### *curl -X POST "http://localhost:3000/api/auth/login" \*

### *-H "Content-Type: application/json" \*

### *-H "accept: application/json" \*

### *-d '{*

### *"correo": "angel.lopez@eurbana.com",*

### *"password": "Admin123!"*

### *}'*

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

### *{*

### *"success": true,*

### *"token": "tokem generado",*

### *"usuario\_id": "68a216b78442c2285baa94d0",*

### *"correo": "angel.lopez@eurbana.com",*

### *"rol": "admin",*

### *"nombre": "Angel",*

### *"apellido": "Reyes"*

### *}*

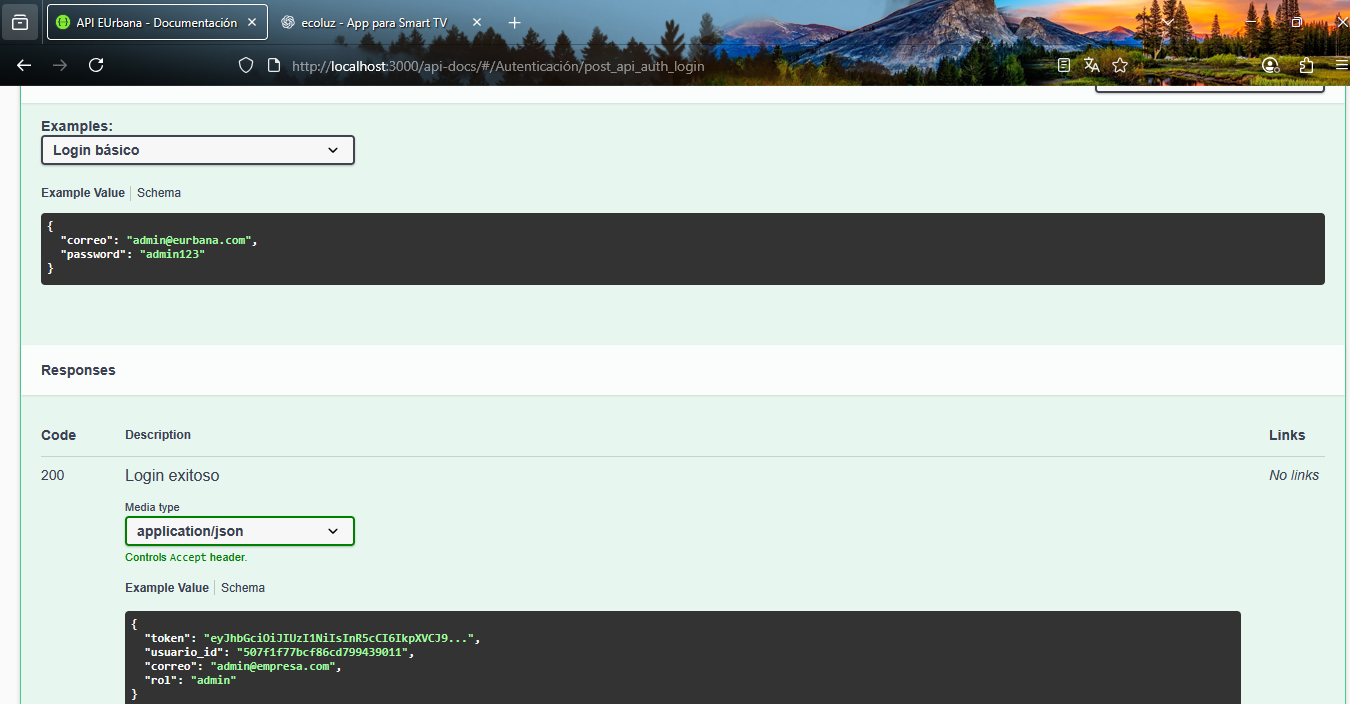
## 6) Criterios de validación

* Se recibe 200 OK y "success": true.
* El usuario\_id corresponde a un ObjectId válido.
* El rol, nombre, apellido y correo coinciden con el usuario autenticado.
* Con el token devuelto es posible acceder a endpoints protegidos.

## 7) Casos negativos probados / esperados

* **400 Bad Request:** Campos faltantes en el body (correo o password).
* **401 Unauthorized:** Credenciales incorrectas (correo no registrado o password inválido).
* **500 Internal Server Error:** Error inesperado (validar logs del servidor).

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-11 Autenticación de usuarios.**
* **RNF-06 Control de acceso.**

## 10) Observaciones

* El token tiene caducidad definida en la configuración del sistema.
* Para todas las rutas protegidas se debe incluir en el header:  
  Authorization: Bearer <TOKEN>

# Prueba – Obtener todos los usuarios

**Endpoint:** GET /api/usuarios  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/get_api_usuarios>  
**Propósito:** Permitir consultar la lista de usuarios registrados en el sistema, activos por defecto y opcionalmente incluir inactivos mediante query params.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve correctamente todos los usuarios registrados, respetando filtros y asegurando que la respuesta contiene el total de registros.

## 2) Precondiciones

* API en ejecución y conexión a BD activa.
* Al menos un usuario previamente creado en el sistema.
* Usuario autenticado con un token válido de administrador u operador autorizado.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/usuarios?incluirInactivos=false  
**Headers:**

* accept: application/json
* Authorization: Bearer <TOKEN\_VALIDO>

## 4) Solicitud de ejemplo (curl)

### *curl -X 'GET' \ 'http://localhost:3000/api/usuarios?incluirInactivos=false' \ -H 'accept: application/json' \ -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'*

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

*{*

*"success": true,*

*"usuarios": [*

*{*

*"\_id": "68815a24c95c89c08ef00a48",*

*"identificador": "ADM-001",*

*"nombre": "Carlos",*

*"apellido": "López",*

*"telefono": "+52-81-5555-1234",*

*"correo": null*

*},*

*{*

*"\_id": "688160ed09ff4c73e63c84f4",*

*"identificador": "ADM-002",*

*"nombre": "Oscar",*

*"apellido": "López",*

*"telefono": "+52-81-5555-1234",*

*"correo": null*

*},*

*{*

*"\_id": "6881662fd65cc9b574d2a389",*

*"identificador": "ADM-009",*

*"nombre": "Luis",*

*"apellido": "Marquez",*

*"telefono": "+52-81-2909-3876",*

*"correo": "admin"*

*},*

*{*

*"\_id": "689f5971a81ec5d05bc66ef7",*

*"identificador": "ADM-001",*

*"nombre": "Carlos",*

*"apellido": "López",*

*"telefono": "+52-81-5555-1234",*

*"correo": "admin"*

*},*

*{*

*"\_id": "68a216b78442c2285baa94d0",*

*"identificador": "ADM-018",*

*"nombre": "Angel",*

*"apellido": "Reyes",*

*"telefono": "+52-81-5555-1235",*

*"correo": "admin"*

*}*

*],*

*"total": 5*

}

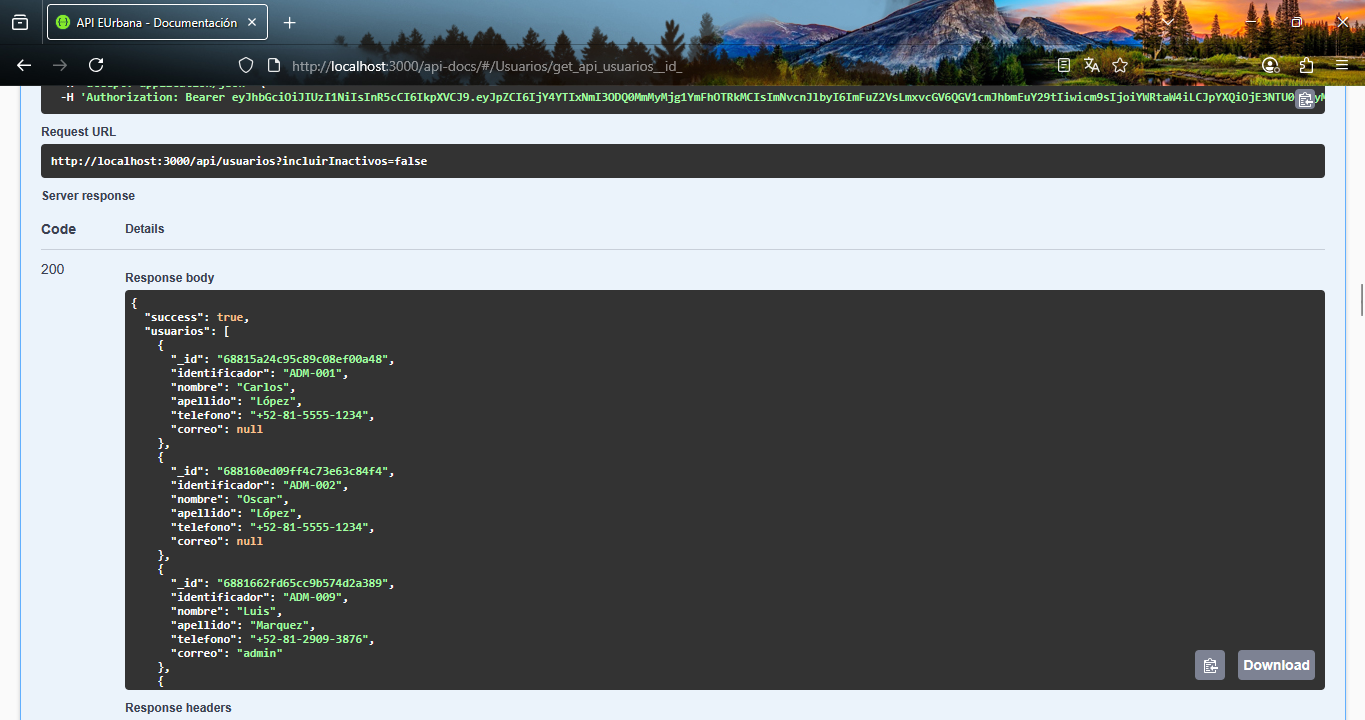
## 6) Criterios de validación

* El sistema devuelve un arreglo usuarios con los registros existentes.
* Cada usuario contiene los campos básicos: \_id, identificador, nombre, apellido, telefono, correo.
* El campo total refleja el número real de usuarios devueltos.
* El código de estado es 200 y success: true.

## 7) Casos negativos probados / esperados

* **401 Unauthorized:** si no se envía un token o es inválido.
* **403 Forbidden:** si el rol del usuario no tiene permisos.
* **500 Internal Server Error:** en caso de error en la base de datos o malformación de la query.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-01:** Rendimiento de la API (respuesta en < 500ms).
* **RNF-06:** Control de acceso.

## 10) Observaciones

* La ruta requiere autenticación obligatoria (JWT válido).
* El query param incluirInactivos permite traer usuarios desactivados (por defecto es false).

# Prueba – Obtener un usuario por ID

**Endpoint:** GET /api/usuarios/{id}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/get_api_usuarios__id_>  
**Propósito:** Permitir consultar la información detallada de un usuario específico mediante su identificador único en la base de datos.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve correctamente los datos de un usuario existente cuando se consulta por su ID en MongoDB.

## 2) Precondiciones

* API en ejecución y conexión a BD activa.
* Usuario autenticado con un token válido de administrador u operador autorizado.
* Existencia de al menos un usuario registrado en la base de datos.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/usuarios/{id}  
Ejemplo de ID usado en la prueba: 6881662fd65cc9b574d2a389

**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

*curl -X 'GET' \*

*'http://localhost:3000/api/usuarios/6881662fd65cc9b574d2a389' \*

*-H 'accept: application/json' \*

*-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...*'

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

*{*

*"success": true,*

*"usuario": {*

*"\_id": "6881662fd65cc9b574d2a389",*

*"identificador": "ADM-009",*

*"nombre": "Luis",*

*"apellido": "Marquez",*

*"telefono": "+52-81-2909-3876",*

*"correo": "admin"*

*}*

*}*

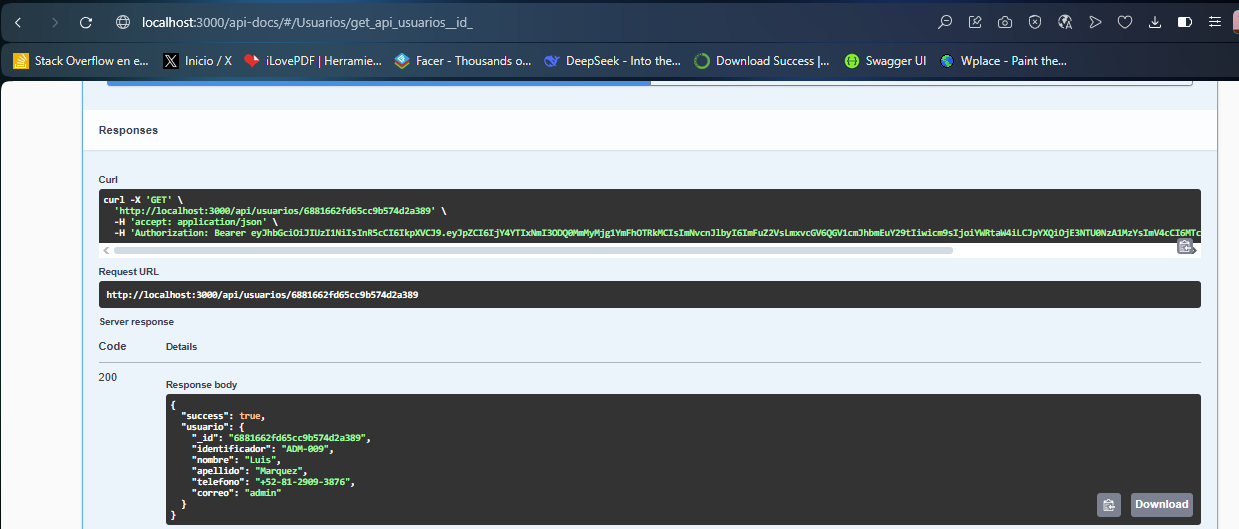
## 6) Criterios de validación

* Se recibe **200 OK** y success: true.
* El objeto usuario contiene los campos básicos (\_id, identificador, nombre, apellido, telefono, correo).
* El ID devuelto coincide con el consultado.

## 7) Casos negativos probados / esperados

* **404 Not Found:** cuando se consulta un ID inexistente o malformado.
* **401 Unauthorized:** si no se envía token válido.
* **403 Forbidden:** si el usuario autenticado no tiene permisos.
* **500 Internal Server Error:** si ocurre un problema en la conexión a la base de datos.

## 8) Evidencia

).

## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-06:** Control de acceso.
* **RNF-01:** Rendimiento de la API (< 500ms por consulta).

## 10) Observaciones

* La ruta está protegida: requiere autenticación JWT.
* La consulta devuelve un único usuario por su **ObjectId** de MongoDB.

# Prueba – Actualizar usuario por ID

**Endpoint:** PUT /api/usuarios/{id}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/put_api_usuarios__id>\_  
**Propósito:** Modificar los datos de un usuario existente identificado por su **ObjectId** en MongoDB.

## 1) Objetivo de la prueba

Confirmar que el sistema permite actualizar correctamente los campos del usuario y devuelve el recurso actualizado.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Usuario autenticado con token válido y permisos (admin/similar).
* Existe un usuario con el **ID** que se va a actualizar.

## 3) Datos de entrada

**Método:** PUT  
**URL:** http://localhost:3000/api/usuarios/{id}  
**Path param de ejemplo:** {id} = 507f1f77bcf86cd799439011  
**Headers:**

* Content-Type: application/json

**Body (JSON) usado en la prueba (ejemplo “actualización completa”):**

*{*

*"identificador": "SUP-002",*

*"nombre": "María Elena",*

*"apellido": "Rodríguez",*

*"telefono": "+52-81-1111-2222",*

*"rol": "supervisor"*

*}*

## 4) Solicitud de ejemplo (curl)

*curl -X PUT "http://localhost:3000/api/usuarios/507f1f77bcf86cd799439011" \*

*-H "Content-Type: application/json" \*

*-H "Authorization: Bearer <TOKEN\_VALIDO>" \*

*-d '{*

*"identificador": "SUP-002",*

*"nombre": "María Elena",*

*"apellido": "Rodríguez",*

*"telefono": "+52-81-1111-2222",*

*"rol": "supervisor"*

*}'*

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

*{*

*"success": true,*

*"message": "Usuario actualizado exitosamente",*

*"usuario": {*

*"\_id": "507f1f77bcf86cd799439011",*

*"identificador": "USR-001",*

*"nombre": "Juan",*

*"apellido": "Pérez",*

*"telefono": "+52-81-1234-5678",*

*"rol": "supervisor"*

*}*

*}*

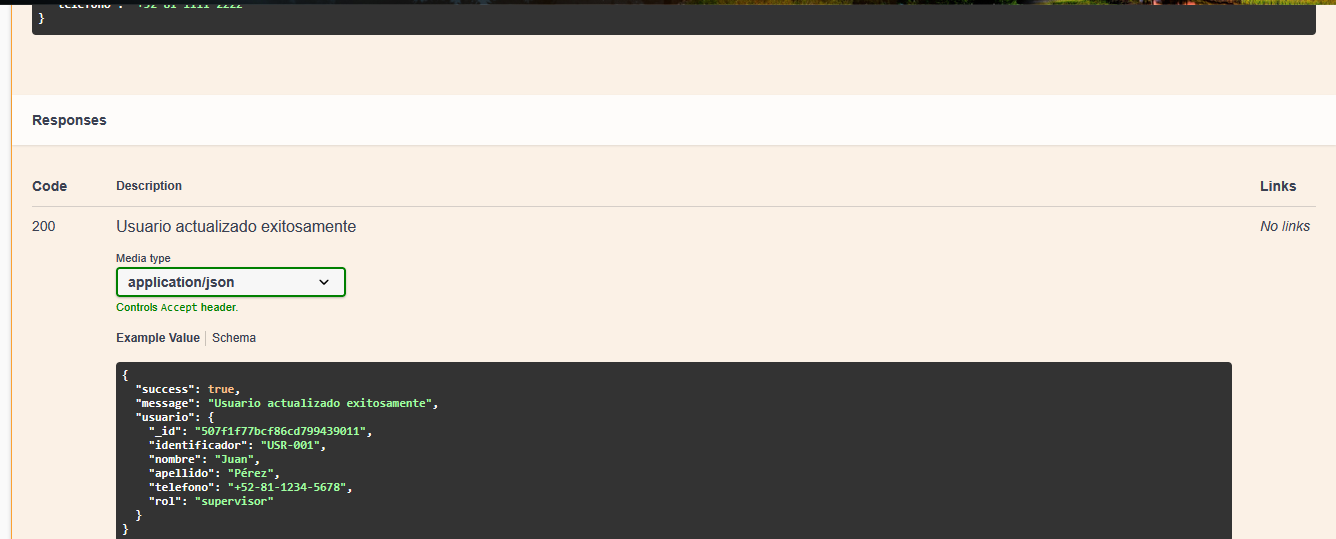
## 6) Criterios de validación

* Respuesta con **200** y success: true.
* El objeto usuario refleja los **cambios** enviados en el body.
* El \_id del usuario actualizado coincide con el {id} de la URL.

## 7) Casos negativos probados / esperados

* **400 Bad Request:** Datos de entrada inválidos o faltantes (ej. tipos incorrectos).
* **404 Not Found:** El {id} no existe o está mal formado.
* **401/403:** Token ausente o sin permisos suficientes.
* **500 Internal Server Error:** Error en la actualización o en la conexión a la BD.

## 8) Evidencia

****

## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-06:** Control de acceso (autorización por rol).
* **RNF-01:** Rendimiento de la API (< 500 ms).

## 10) Observaciones

* La ruta está **protegida** (JWT).

# Prueba – Eliminar usuario por ID

**Endpoint:** DELETE /api/usuarios/{id}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/delete_api_usuarios__id>\_  
**Propósito:** Eliminar (baja) un usuario existente identificado por su **ObjectId** en MongoDB.

## 1) Objetivo de la prueba

Comprobar que el sistema elimina correctamente un usuario existente y retorna la confirmación con el recurso eliminado.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Usuario autenticado con token válido y permisos (admin/similar).
* Existe un usuario con el **ID** que se va a eliminar.

## 3) Datos de entrada

**Método:** DELETE  
**URL:** http://localhost:3000/api/usuarios/{id}  
**Path param de ejemplo:** {id} = 507f1f77bcf86cd799439011  
**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

*curl -X DELETE "http://localhost:3000/api/usuarios/507f1f77bcf86cd799439011" \*

*-H "accept: application/json" \*

#### -H "Authorization: Bearer <TOKEN\_VALIDO>"

#### 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):** ***{***

***"success": true,***

***"message": "Usuario eliminado exitosamente",***

***"usuarioEliminado": {***

***"\_id": "507f1f77bcf86cd799439011",***

***"identificador": "USR-001",***

***"nombre": "Juan",***

***"apellido": "Pérez",***

***"telefono": "+52-81-1234-5678",***

***"rol": "supervisor"***

***}***

***}***

## 6) Criterios de validación

* Se recibe **200 OK** y success: true.
* usuarioEliminado.\_id coincide con el {id} enviado.
* El usuario deja de aparecer en listados posteriores (GET /api/usuarios) o aparece con estado acorde a la política (baja lógica/física).

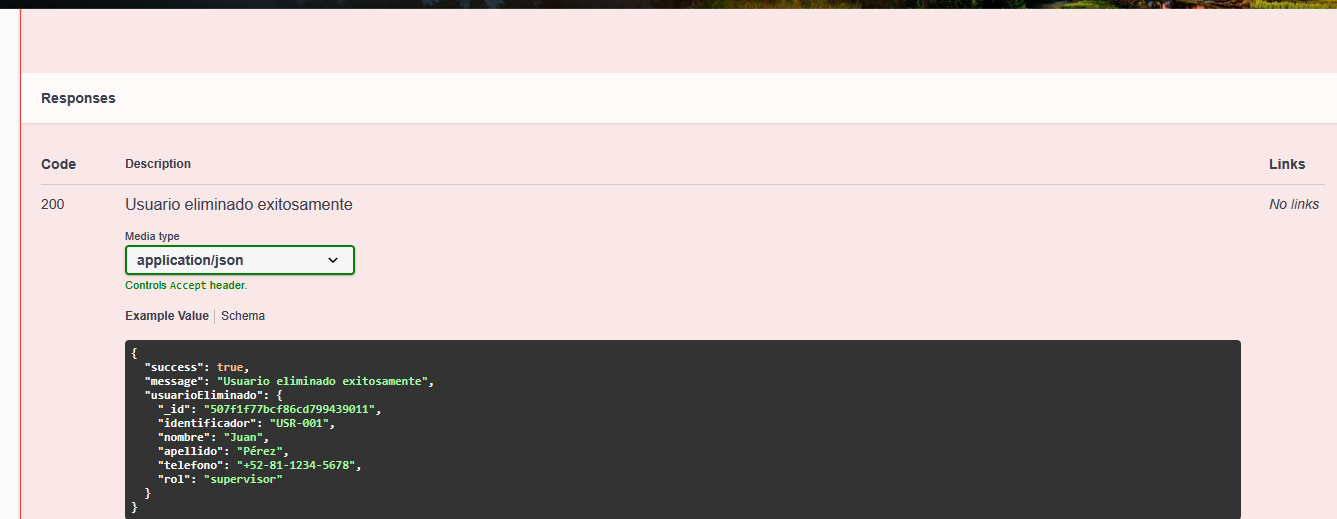
## 7) Casos negativos probados / esperados

* **404 Not Found:**

{ "success": false, "message": "Usuario no encontrado" }

* **401/403:** Token ausente o sin permisos.
* **500 Internal Server Error:** Problema interno al eliminar o al acceder a la BD.

## 8) Evidencia

****

## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-06:** Control de acceso.
* **RNF-01:** Rendimiento de la API (< 500 ms).

## 10) Observaciones

* La ruta está protegida (JWT).

# Prueba – Obtener usuario por identificador

**Endpoint:** GET /api/usuarios/identificador/{identificador}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/get_api_usuarios_identificador__identificador>\_  
**Propósito:** Consultar los datos de un usuario usando su **identificador funcional** (p. ej., USR-001), no el ObjectId de MongoDB.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve correctamente la información del usuario cuando se consulta por su **identificador**.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Ruta protegida: contar con **token JWT** válido
* Existe al menos un usuario con el identificador a consultar (ej. USR-001).

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/usuarios/identificador/{identificador}  
**Path param de ejemplo:** {identificador} = USR-001  
**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

***curl -X GET "http://localhost:3000/api/usuarios/identificador/USR-001" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***{***

***"\_id": "507f1f77bcf86cd799439011",***

***"identificador": "USR-001",***

***"nombre": "Juan",***

***"apellido": "Pérez",***

***"telefono": "+52-81-1234-5678",***

***"rol": "supervisor"***

***}***

## 6) Criterios de validación

* Se recibe **200** con el objeto del usuario correspondiente al identificador consultado.
* El campo identificador del response **coincide** con el enviado en la URL.
* Estructura de campos esperada: \_id, identificador, nombre, apellido, telefono, rol.

## 7) Casos negativos probados / esperados

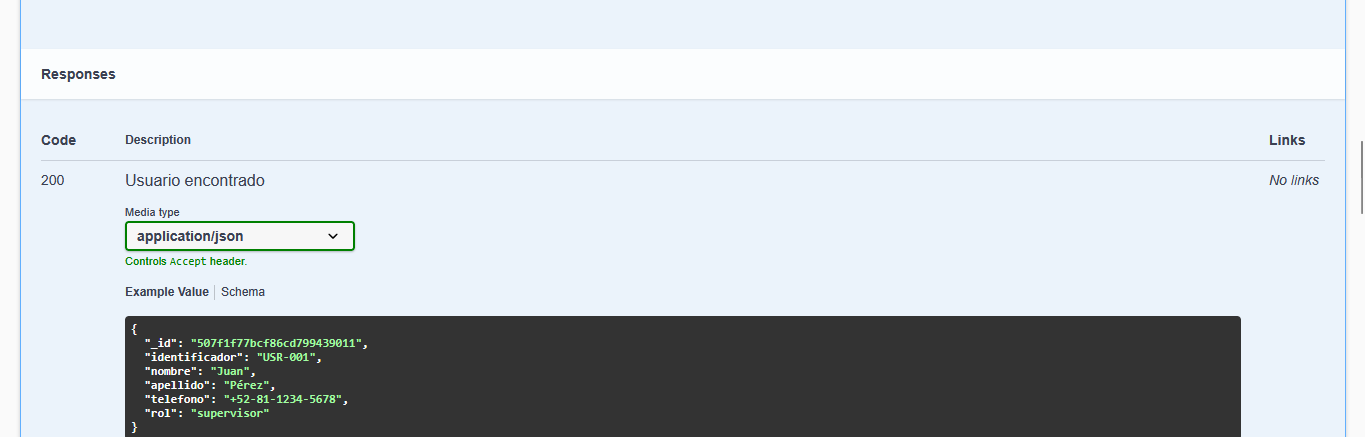
* **404 Not Found:**

{ "error": "Usuario no encontrado" }

 **401/403:** Token ausente o sin permisos (si la ruta está protegida).

 **500 Internal Server Error:** Fallo interno (p. ej., error en la consulta a BD).

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-01:** Rendimiento de la API (< 500 ms).
* **RNF-06:** Control de acceso

## 10) Observaciones

* Útil cuando el **identificador** es la clave operativa del área (más legible que el ObjectId).
* Si existen reglas de formato (prefijos como ADM-, USR-, etc.), conviene validarlas y documentarlas en Swagger para evitar 400 por formatos inválidos.

# Prueba – Obtener usuarios por **rol**

**Endpoint:** GET /api/usuarios/rol/{rol}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Usuarios/get_api_usuarios_rol__rol>\_  
**Propósito:** Listar todos los usuarios que pertenecen a un **rol** específico.

## 1) Objetivo de la prueba

Validar que el sistema filtra y devuelve correctamente los usuarios cuyo campo rol coincide con el parámetro solicitado.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Autenticación con token JWT válido.
* Existen usuarios con el rol a consultar.
* **Roles permitidos:** admin, supervisor, usuario.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/usuarios/rol/{rol}  
**Path param de ejemplo:** {rol} = supervisor  
**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

***curl -X GET "http://localhost:3000/api/usuarios/rol/supervisor" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***[***

***{***

***"\_id": "507f1f77bcf86cd799439011",***

***"identificador": "USR-001",***

***"nombre": "Juan",***

***"apellido": "Pérez",***

***"telefono": "+52-81-1234-5678",***

***"rol": "supervisor"***

***}***

***]***

Nota: Si no hay usuarios con ese rol, se espera **array vacío** [] (no error).

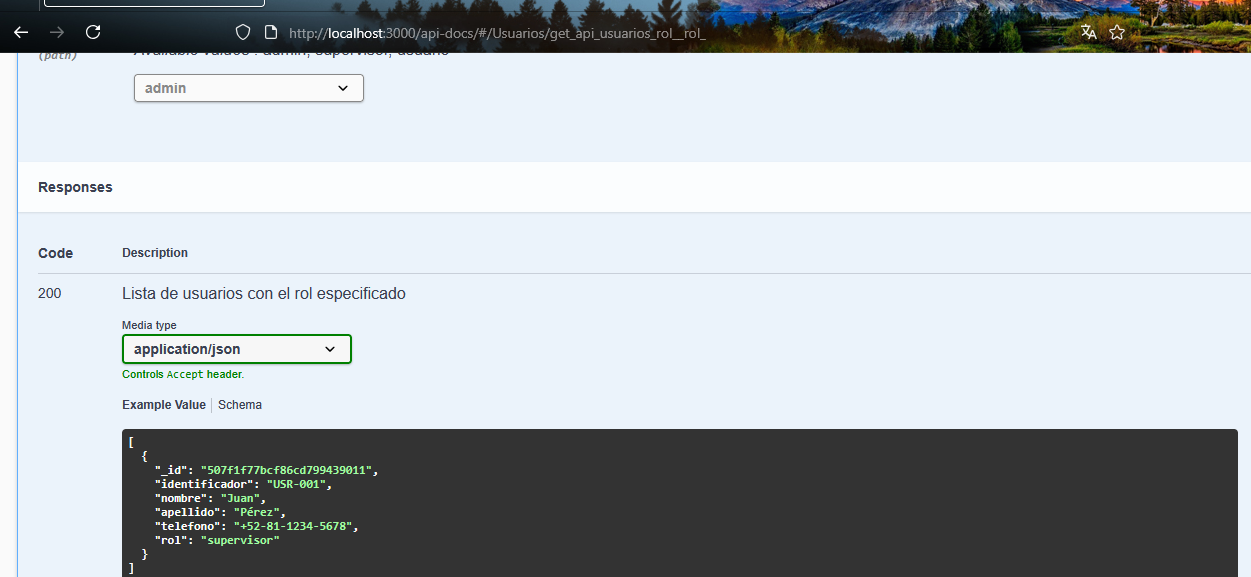
## 6) Criterios de validación

* Respuesta **200 OK**.
* El cuerpo es un **array**.
* **Todos** los elementos devueltos tienen rol === {rol}.
* Estructura de cada usuario: \_id, identificador, nombre, apellido, telefono, rol.

## 7) Casos negativos probados / esperados

* **400 Bad Request :** Rol inválido (no está en admin|supervisor|usuario).
* **401/403:** Token ausente o sin permisos
* **500 Internal Server Error:** Falla al consultar la BD u otro error interno.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-10:** Gestión de usuarios y roles.
* **RNF-01:** Rendimiento (< 500 ms).
* **RNF-06:** Control de acceso (si aplica).

## 10) Observaciones

* Validar en Swagger que el parámetro {rol} tenga los enum (admin, supervisor, usuario) para evitar solicitudes inválidas.

# Prueba – Obtener todas las luminarias activas

**Endpoint:** GET /api/luminarias  
**Propósito:** Listar todas las luminarias **activas** registradas en el sistema.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve la lista completa de luminarias **activas**, con su información básica y coordenadas.

## 2) Precondiciones

* API en ejecución y BD conectada.
* (Si aplica) Autenticado con token JWT válido.
* Base con luminarias registradas y campo activo: true.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/luminarias  
**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

***curl -X GET "http://localhost:3000/api/luminarias" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***[***

***{***

***"\_id": "507f1f77bcf86cd799439011",***

***"identificador": "LUM-001-MTY",***

***"tipo\_luminaria": "LED",***

***"pais": "México",***

***"estado": "Nuevo León",***

***"ciudad": "Monterrey",***

***"region": "Centro",***

***"coordenadas": {***

***"lat": 25.6866,***

***"lng": -100.3161***

***},***

***"fecha\_instalacion": "2023-01-15T00:00:00.000Z",***

***"activo": true***

***}***

***]***

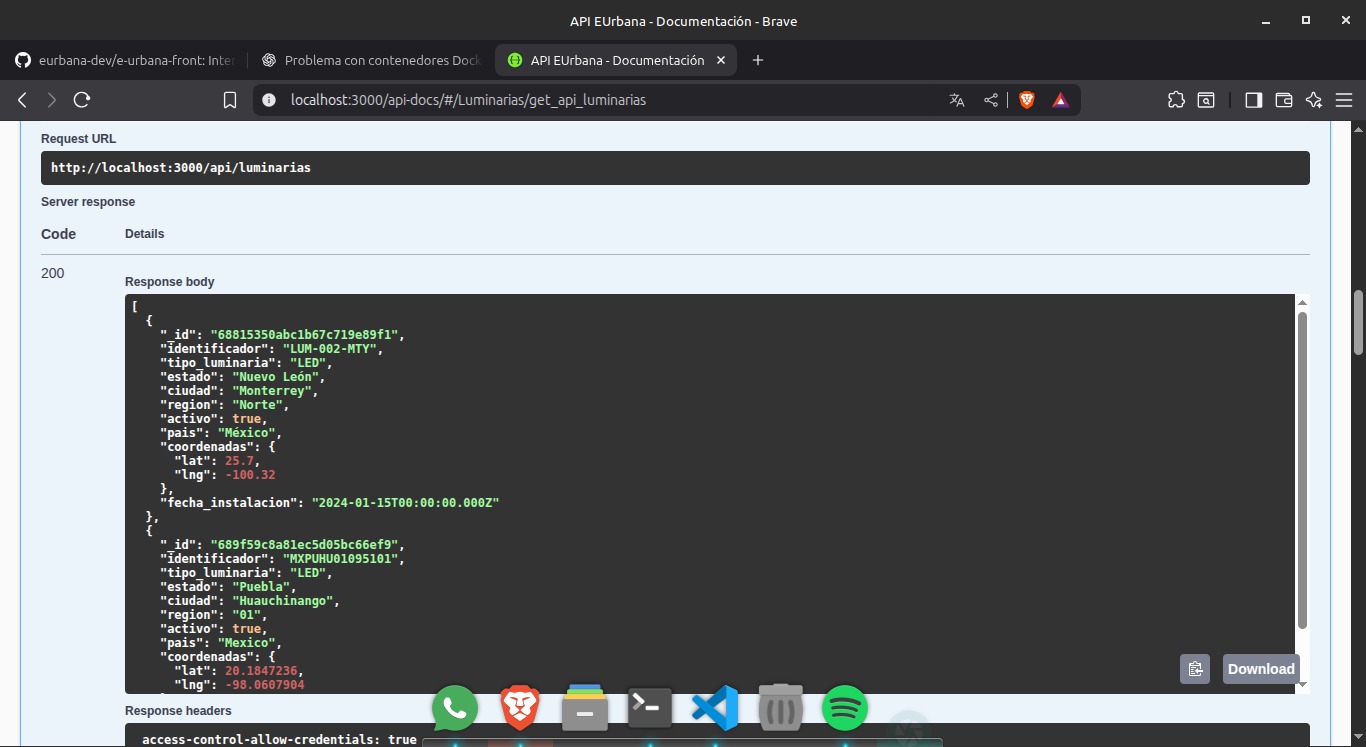
## 6) Criterios de validación

* Respuesta **200 OK**.
* Cada luminaria activa presenta campos esperados:
  + \_id, identificador, tipo\_luminaria, pais, estado, ciudad, region,
  + coordenadas.lat, coordenadas.lng, fecha\_instalacion, activo: true.

## 7) Casos negativos probados / esperados

* **401/403:** Sin token o sin permisos (si la ruta está protegida).
* **500 Internal Server Error:** Falla de conexión a BD o error interno.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-02:** Gestión de luminarias y sensores.
* **RF-03:** Visualización de datos (base para dashboards/mapas).
* **RNF-01:** Rendimiento de la API (< 500 ms).

## 10) Observaciones

* Útil como fuente para el **mapa** y **dashboard**.

# Prueba – Obtener todos los registros de sensores (Consumo)

**Endpoint:** GET /api/consumo  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/get_api_consumo>  
**Propósito:** Listar los **registros de sensores** (consumo, lúmenes, estado) reportados por las luminarias, con opción de limitar la cantidad de resultados.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve correctamente la lista de registros de consumo y que el **parámetro limite** restringe la cantidad de elementos retornados.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Autenticación con token JWT válido.
* Existen registros de consumo en la base de datos.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/consumo  
**Query params (opcionales):**

* limite (integer, default: **100**) → Límite de registros a retornar.

**Headers:**

* accept: application/json
* Authorization: Bearer <TOKEN\_VALIDO> (si la ruta está protegida)

## 4) Solicitudes de ejemplo (curl)

**a) Sin límites (usa el default 100):**

***curl -X GET "http://localhost:3000/api/consumo" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

**b) Con límite explícito (p. ej. 20):**

***curl -X GET "http://localhost:3000/api/consumo?limite=20" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***[***

***{***

***"\_id": "507f1f77bcf86cd799439011",***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***}***

***]***

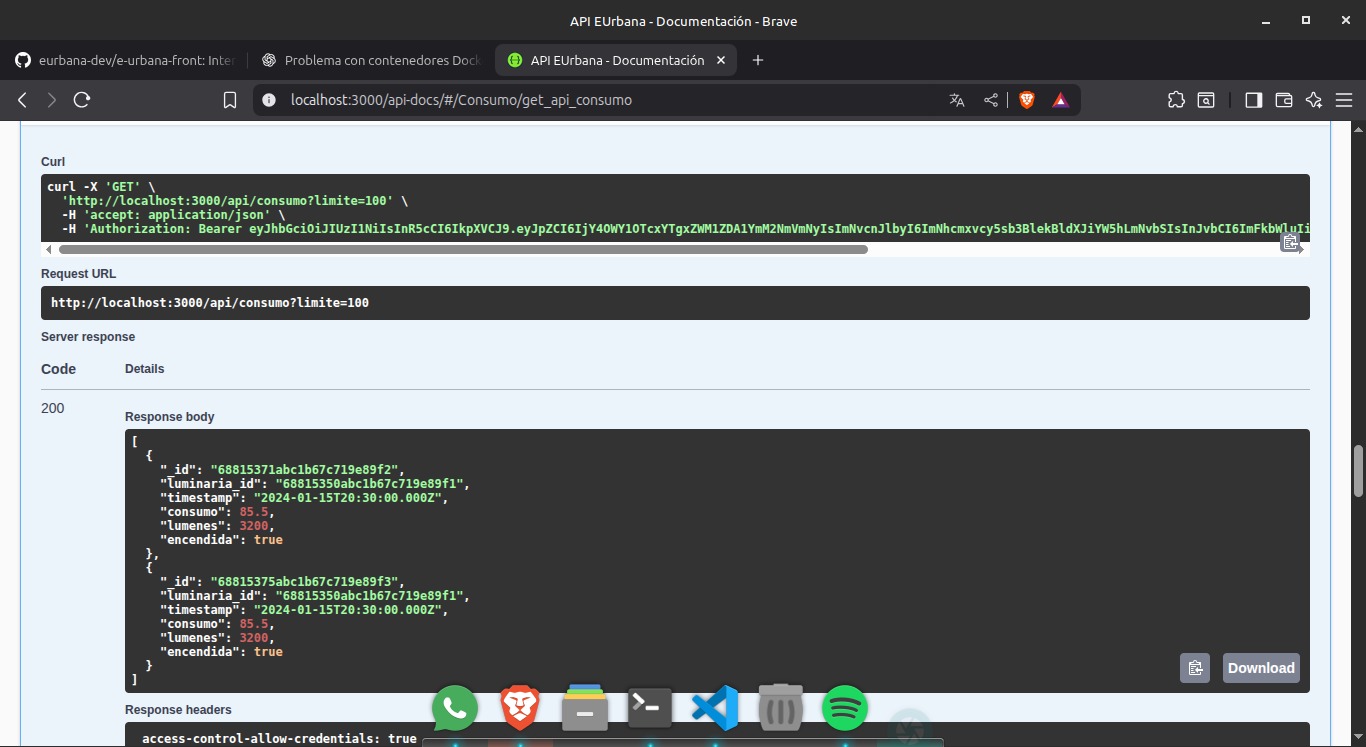
## 6) Criterios de validación

* Respuesta **200 OK**.

## 7) Casos negativos probados / esperados

* **400 Bad Request (recomendado):** limite no numérico o negativo.
* **401/403:** Token ausente o inválido
* **500 Internal Server Error:** Falla de conexión a BD o error interno.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-03:** Visualización de datos energéticos.
* **RF-06:** Filtrado de datos (parcial con limite).
* **RNF-01:** Rendimiento de la API (< 500 ms).

# Prueba – Crear nuevo registro de sensor (Consumo)

**Endpoint:** POST /api/consumo  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/post_api_consumo>  
**Propósito:** Registrar un **nuevo dato de sensor** (consumo, lúmenes, estado) asociado a una luminaria existente.

## 1) Objetivo de la prueba

Comprobar que el sistema **crea** un registro de consumo válido y retorna el **ID** del registro creado.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Autenticación con **JWT**.

## 3) Datos de entrada

**Método:** POST  
**URL:** http://localhost:3000/api/consumo  
**Headers:**

* Content-Type: application/json

**Body (JSON) usado en la prueba**

***{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***}***

## 4) Solicitud de ejemplo (curl)

***curl -X POST "http://localhost:3000/api/consumo" \***

***-H "Content-Type: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>" \***

***-d '{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***}'***

## 5) Salida esperada

**Código HTTP:** 201 Created  
**Respuesta (JSON):**

***{***

***"message": "Registro de sensor creado exitosamente",***

***"id": "507f1f77bcf86cd799439011"***

***}***

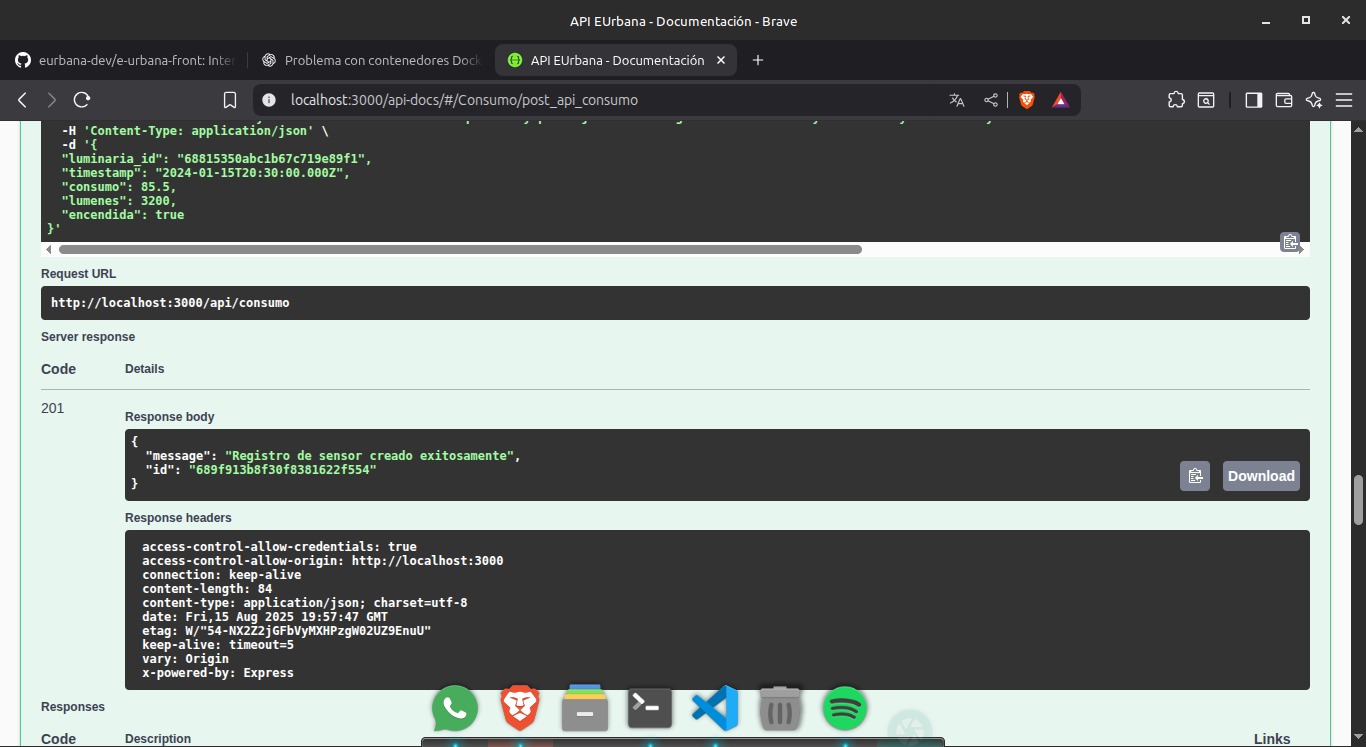
## 6) Criterios de validación

* Se recibe **201** y el **id** del registro creado.
* El luminaria\_id corresponde a una luminaria existente.

## 7) Casos negativos probados / esperados

* **400 Bad Request:**
  + Campos faltantes o tipos inválidos
  + timestamp inválido.
  + luminaria\_id mal formado.
* **404 Not Found :** luminaria\_id apunta a luminaria inexistente.
* **401/403:** Falta/Inválido el token
* **500 Internal Server Error:** Error al insertar o al acceder a la BD.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-05:** Recepción de datos en tiempo real (IoT).
* **RF-03:** Visualización de datos energéticos
* **RNF-01:** Rendimiento de la API (< 500 ms).

# Prueba – Obtener registro de sensor por **ID** (Consumo)

**Endpoint:** GET /api/consumo/{id}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/get_api_consumo__id>\_  
**Propósito:** Consultar un **registro de sensor** específico mediante su **ObjectId** de MongoDB.

## 1) Objetivo de la prueba

Verificar que el sistema devuelve correctamente un registro de consumo existente cuando se consulta por su **ID**.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Existe al menos un registro de consumo en la colección.
* Autenticación con token JWT válido.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/consumo/{id}  
**Path param (ejemplo):** {id} = 507f1f77bcf86cd799439011  
**Headers:**

* accept: application/json

## 4) Solicitud de ejemplo (curl)

***curl -X GET "http://localhost:3000/api/consumo/507f1f77bcf86cd799439011" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***{***

***"\_id": "507f1f77bcf86cd799439011",***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***}***

## 6) Criterios de validación

* Se recibe **200 OK**.
* El objeto devuelto contiene los campos esperados:  
  \_id, luminaria\_id, timestamp, consumo, lumenes, encendida.
* El \_id del response **coincide** con el {id} consultado.

## 7) Casos negativos probados / esperados

* **404 Not Found:**

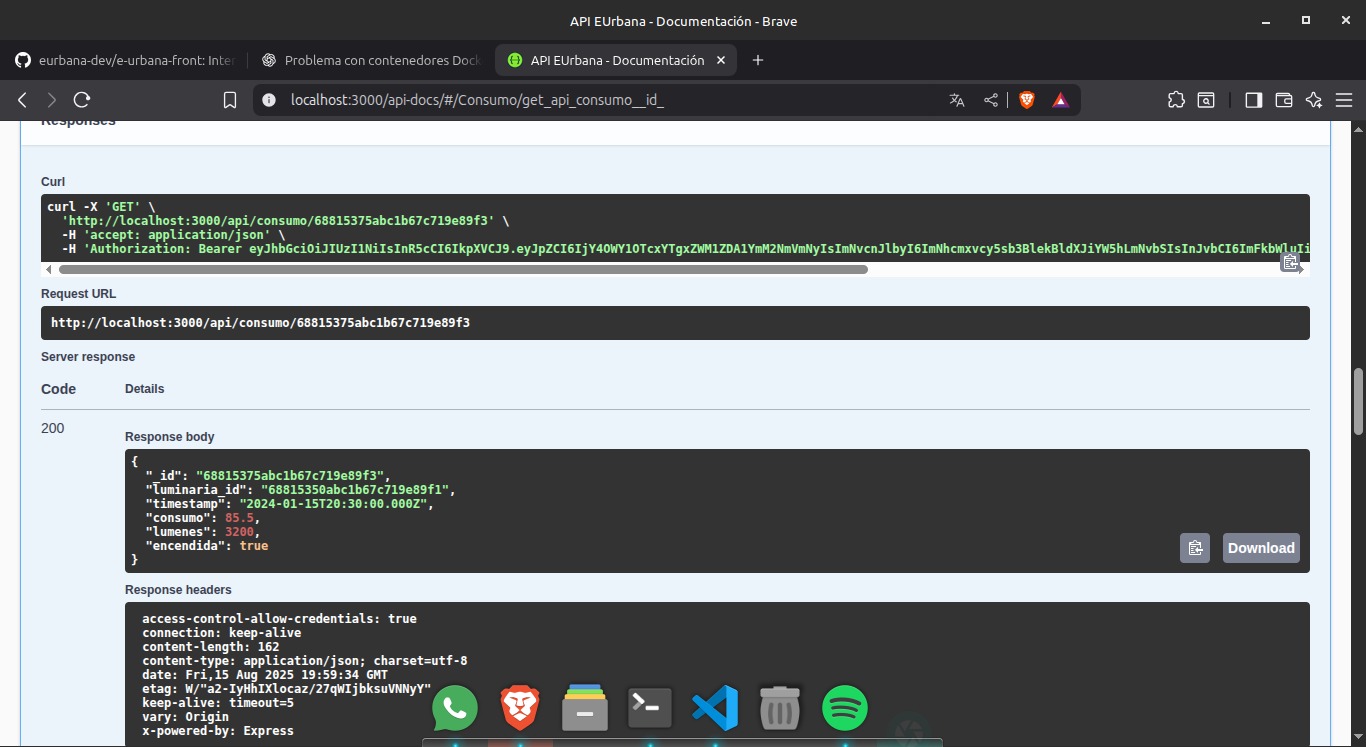
{ "error": "Registro de sensor no encontrado" }

 **400 Bad Request (recomendado):** {id} con formato inválido

 **401/403:** Token ausente o sin permisos

 **500 Internal Server Error:** Error interno o fallo de conexión a BD.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-03:** Visualización de datos energéticos.
* **RF-06:** Filtrado/consulta por entidad específica.
* **RNF-01:** Rendimiento de la API (< 500 ms).

## 10) Observaciones

* Útil para depuración y para vincular un punto del gráfico/tabla con su registro fuente.

# Prueba – Crear **múltiples** registros de sensor (Bulk)

**Endpoint:** POST /api/consumo/bulk  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/post_api_consumo_bulk>  
**Propósito:** Insertar **varios** registros de sensores en una sola operación para optimizar el rendimiento y reducir la latencia de múltiples peticiones.

## 1) Objetivo de la prueba

Comprobar que el sistema acepta un arreglo de registros válidos, inserta **todos** los elementos y retorna la **cantidad** insertada y sus **IDs**.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Existen las luminarias referenciadas en luminaria\_id.
* JWT válida.

## 3) Datos de entrada

**Método:** POST  
**URL:** http://localhost:3000/api/consumo/bulk  
**Headers:**

* Content-Type: application/json

**Body (JSON) usado en la prueba:**

***{***

***"registros": [***

***{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***},***

***{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:29:00.000Z",***

***"consumo": 82.1,***

***"lumenes": 3150,***

***"encendida": true***

***}***

***]***

***}***

## 4) Solicitud de ejemplo (curl)

***curl -X POST "http://localhost:3000/api/consumo/bulk" \***

***-H "Content-Type: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>" \***

***-d '{***

***"registros": [***

***{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:30:00.000Z",***

***"consumo": 85.5,***

***"lumenes": 3200,***

***"encendida": true***

***},***

***{***

***"luminaria\_id": "507f1f77bcf86cd799439012",***

***"timestamp": "2024-01-15T20:29:00.000Z",***

***"consumo": 82.1,***

***"lumenes": 3150,***

***"encendida": true***

***}***

***]***

***}'***

## 5) Salida esperada

**Código HTTP:** 201 Created  
**Respuesta (JSON):**

***{***

***"message": "Registros creados exitosamente",***

***"insertados": 25,***

***"ids": [***

***"507f1f77bcf86cd7994390aa",***

***"507f1f77bcf86cd7994390ab"***

***]***

***}***

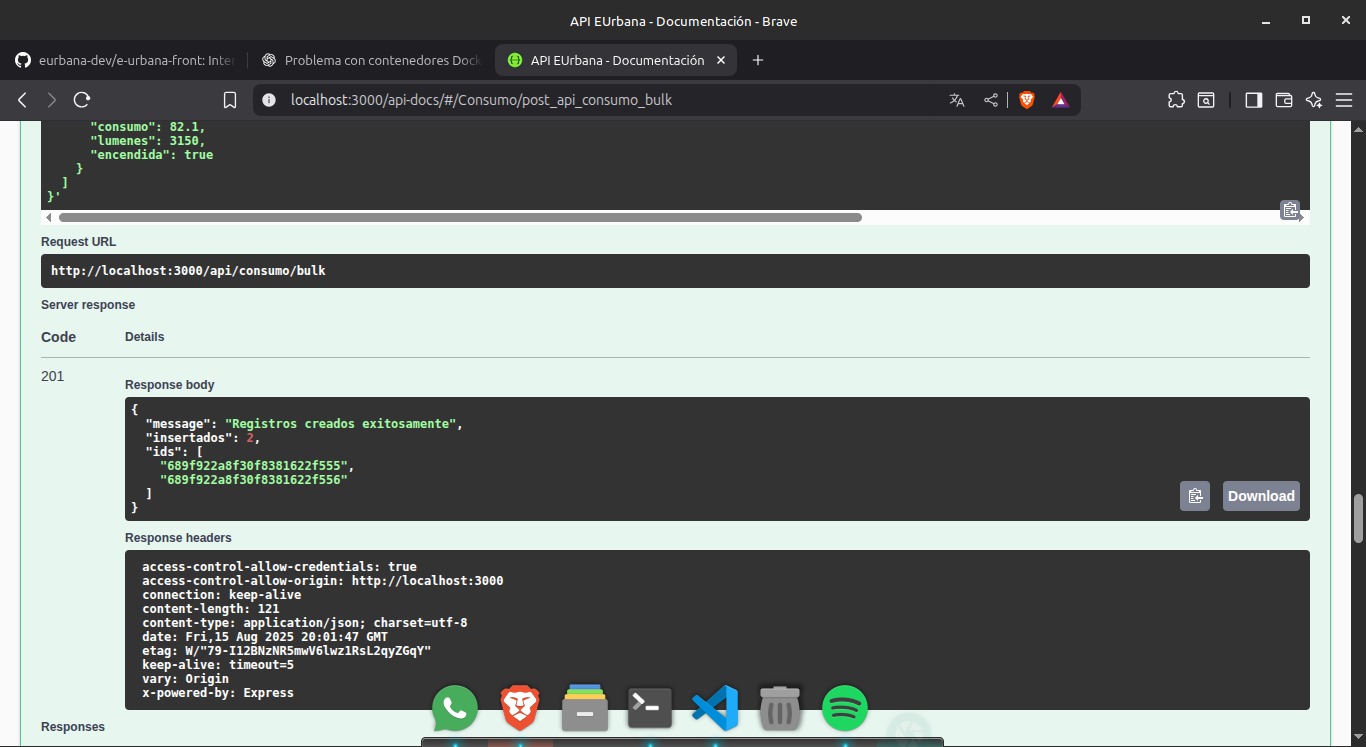
## 6) Criterios de validación

* Respuesta **201** y mensaje de éxito.

## 7) Casos negativos probados / esperados

* **400 Bad Request:**
  + Body sin registros o no es un arreglo.
  + Algún registro con tipos inválidos o campos faltantes.
* **404 Not Found (recomendado):** luminaria\_id no existe.
* **401/403:** Token ausente/ inválido (si aplica).
* **500 Internal Server Error:** Error durante inserción masiva o fallo BD.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-05:** Recepción de datos en tiempo real (IoT).
* **RNF-01:** Rendimiento (reduce overhead al insertar en lote).
* **RNF-09:** Optimización de consultas/índices (recomendado para volumen).

## 10) Observaciones

* Recomendar límite de tamaño por batch (p. ej., 1k–5k registros) para evitar timeouts.

# Prueba – Obtener **estadísticas de consumo** por luminaria

**Endpoint:** GET /api/consumo/estadisticas/{luminaria\_id}  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/get_api_consumo_estadisticas__luminaria_id>\_  
**Propósito:** Calcular y devolver estadísticas agregadas (totales, promedios, máximos/mínimos, tiempo encendida) de una luminaria determinada, opcionalmente en un **rango de fechas**.

## 1) Objetivo de la prueba

Verificar que el endpoint retorna correctamente los **agregados** de consumo para una luminaria específica y que respeta los filtros de **fecha\_inicio** y **fecha\_fin** cuando se proporcionan.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Existe la luminaria con el **luminaria\_id** consultado y **registros de consumo** asociados.
* Autenticación con token JWT válido.

## 3) Datos de entrada

**Método:** GET  
**URL:** http://localhost:3000/api/consumo/estadisticas/{luminaria\_id}

**Path param:**

* luminaria\_id (string, requerido) → ID de la luminaria.

**Query params (opcionales):**

* fecha\_inicio (string, date-time ISO 8601) → Inicio del rango, ej. 2024-01-01T00:00:00.000Z
* fecha\_fin (string, date-time ISO 8601) → Fin del rango, ej. 2024-01-31T23:59:59.000Z

**Headers:**

* accept: application/json
* Authorization: Bearer <TOKEN\_VALIDO>

## 4) Solicitudes de ejemplo (curl)

**a) Sin rango de fechas (toma todo el histórico):**

***curl -X GET "http://localhost:3000/api/consumo/estadisticas/507f1f77bcf86cd799439012" \***

***-H "accept: application/json"***

***-H "Authorization: Bearer <TOKEN\_VALIDO>***

**b) Con rango de fechas:**

***curl -X GET "http://localhost:3000/api/consumo/estadisticas/507f1f77bcf86cd799439012?fecha\_inicio=2024-01-01T00:00:00.000Z&fecha\_fin=2024-01-31T23:59:59.000Z" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***{***

***"\_id": "507f1f77bcf86cd799439012",***

***"consumoTotal": 2560.5,***

***"consumoPromedio": 85.35,***

***"lumenesTotal": 96000,***

***"lumenesPromedio": 3200,***

***"tiempoEncendida": 25,***

***"totalRegistros": 30,***

***"consumoMaximo": 95.2,***

***"consumoMinimo": 0***

***}***

## 6) Criterios de validación

* **200 OK** y objeto con las métricas esperadas.
* \_id coincide con luminaria\_id solicitado.
* totalRegistros corresponde al número de documentos considerados (todo o dentro del rango).

## 7) Casos negativos probados / esperados

* **404 Not Found:** No hay datos para la luminaria

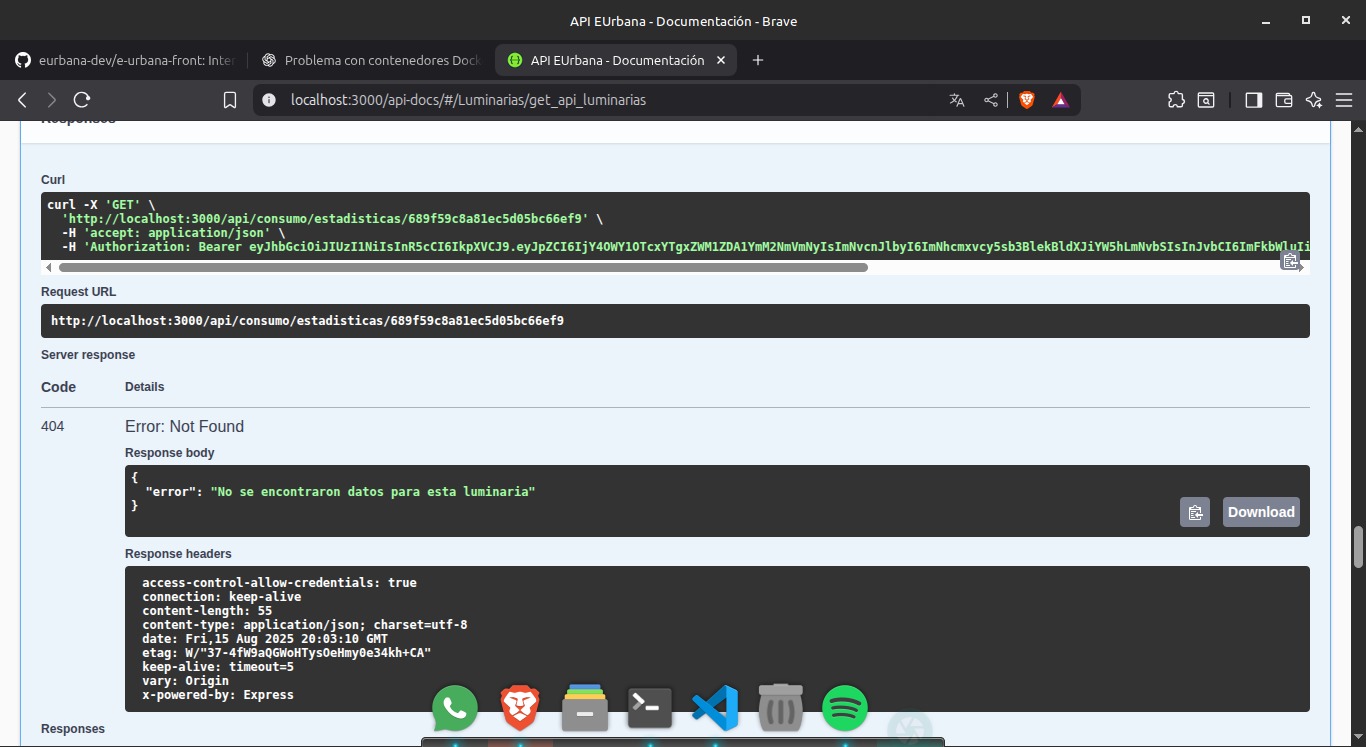
{ "error": "No se encontraron datos para la luminaria" }

 **400 Bad Request (recomendado):** Fechas en formato inválido o fecha\_inicio > fecha\_fin.

 **401/403:** Token ausente o sin permisos

 **500 Internal Server Error:** Error en la agregación o en la conexión a BD.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RF-03:** Visualización de datos energéticos.
* **RF-06:** Filtrado por rango de tiempo.
* **RNF-01:** Rendimiento de la API (< 500 ms para rangos moderados).
* **RNF-09:** Optimización de consultas/índices (clave para agregaciones).

# Prueba – Eliminar registros antiguos (mantenimiento de datos)

**Endpoint:** DELETE /api/consumo/limpieza/antiguos  
**Ruta en Swagger:** <http://localhost:3000/api-docs/#/Consumo/delete_api_consumo_limpieza_antiguos>  
**Propósito:** Borrar registros de sensores **anteriores** a una fecha\_limite para controlar el crecimiento de la base y mejorar el rendimiento.

## 1) Objetivo de la prueba

Verificar que, dado un parámetro fecha\_limite válido, el sistema elimina todos los registros **con timestamp anterior** a esa fecha y devuelve el **conteo** de documentos eliminados.

## 2) Precondiciones

* API en ejecución y BD conectada.
* Autenticación JWT válida y permisos de rol
* Existen registros con timestamp **anteriores** a fecha\_limite.

## 3) Datos de entrada

**Método:** DELETE  
**URL:** http://localhost:3000/api/consumo/limpieza/antiguos  
**Query params (requerido):**

* fecha\_limite (string, date-time) — ej.: 2024-01-01T00:00:00.000Z

**Headers:**

* accept: application/json

## 4) Solicitudes de ejemplo (curl)

**a) Eliminación con fecha límite :**

***curl -X DELETE "http://localhost:3000/api/consumo/limpieza/antiguos?fecha\_limite=2024-01-01T00:00:00.000Z" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

**b) (Negativo) Sin fecha\_limite:**

***curl -X DELETE "http://localhost:3000/api/consumo/limpieza/antiguos" \***

***-H "accept: application/json" \***

***-H "Authorization: Bearer <TOKEN\_VALIDO>"***

## 5) Salida esperada

**Código HTTP:** 200 OK  
**Respuesta (JSON):**

***{***

***"message": "Registros antiguos eliminados exitosamente",***

***"eliminados": 1250***

***}***

eliminados debe reflejar el número real de documentos borrados.

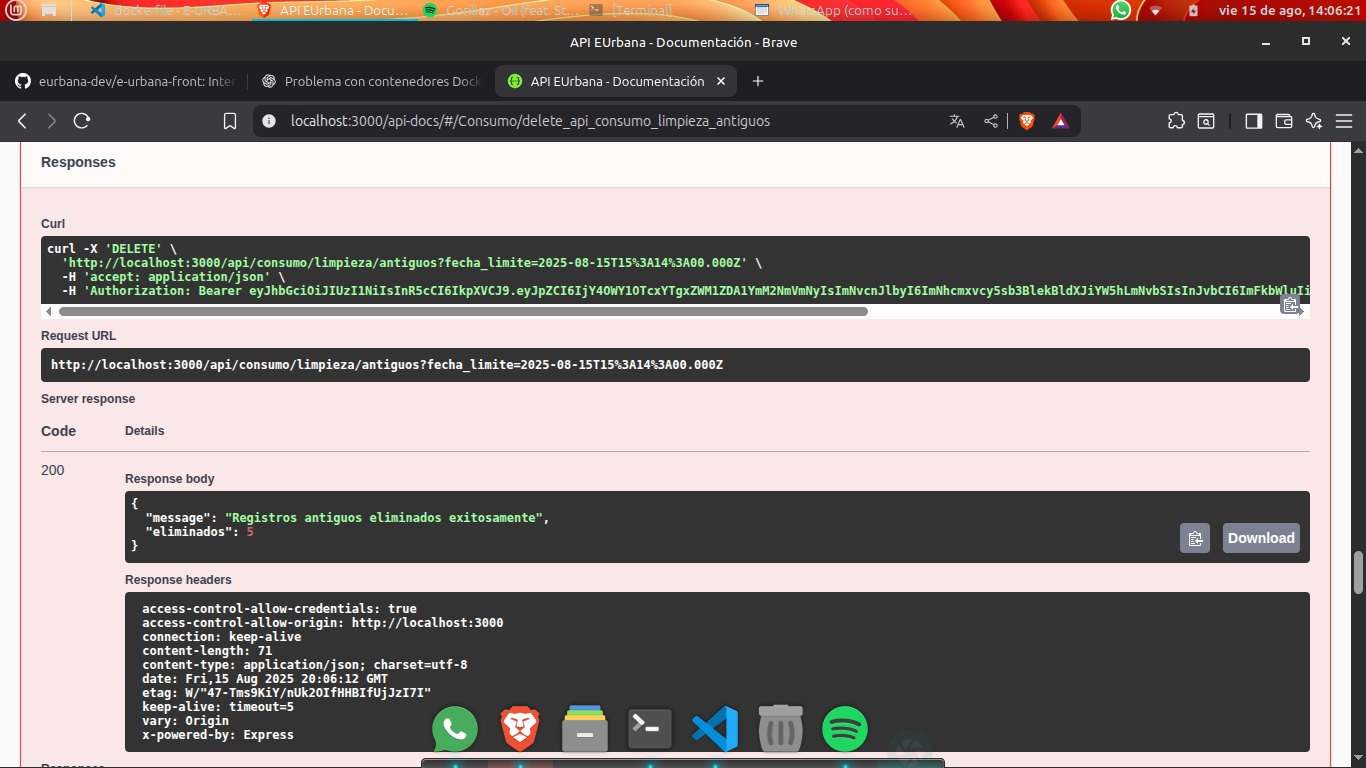
## 6) Criterios de validación

* Respuesta **200 OK** con message y eliminados numérico ≥ 0.
* Los registros con timestamp < fecha\_limite ya no están presentes en consultas posteriores (GET /api/consumo).

## 7) Casos negativos probados / esperados

* **400 Bad Request:** Falta fecha\_limite o formato inválido.
* **401/403:** Token ausente o sin permisos suficientes
* **500 Internal Server Error:** Falla de conexión BD, error en operación de borrado u operación masiva.

## 8) Evidencia



## 9) Trazabilidad a requisitos

* **RNF-09:** Optimización de consultas/índices (mantenimiento de datos mejora performance).
* **RF-03:** Visualización de datos energéticos (depende de que el histórico sea manejable).
* **RNF-01:** Rendimiento de la API (operaciones de mantenimiento eficientes).