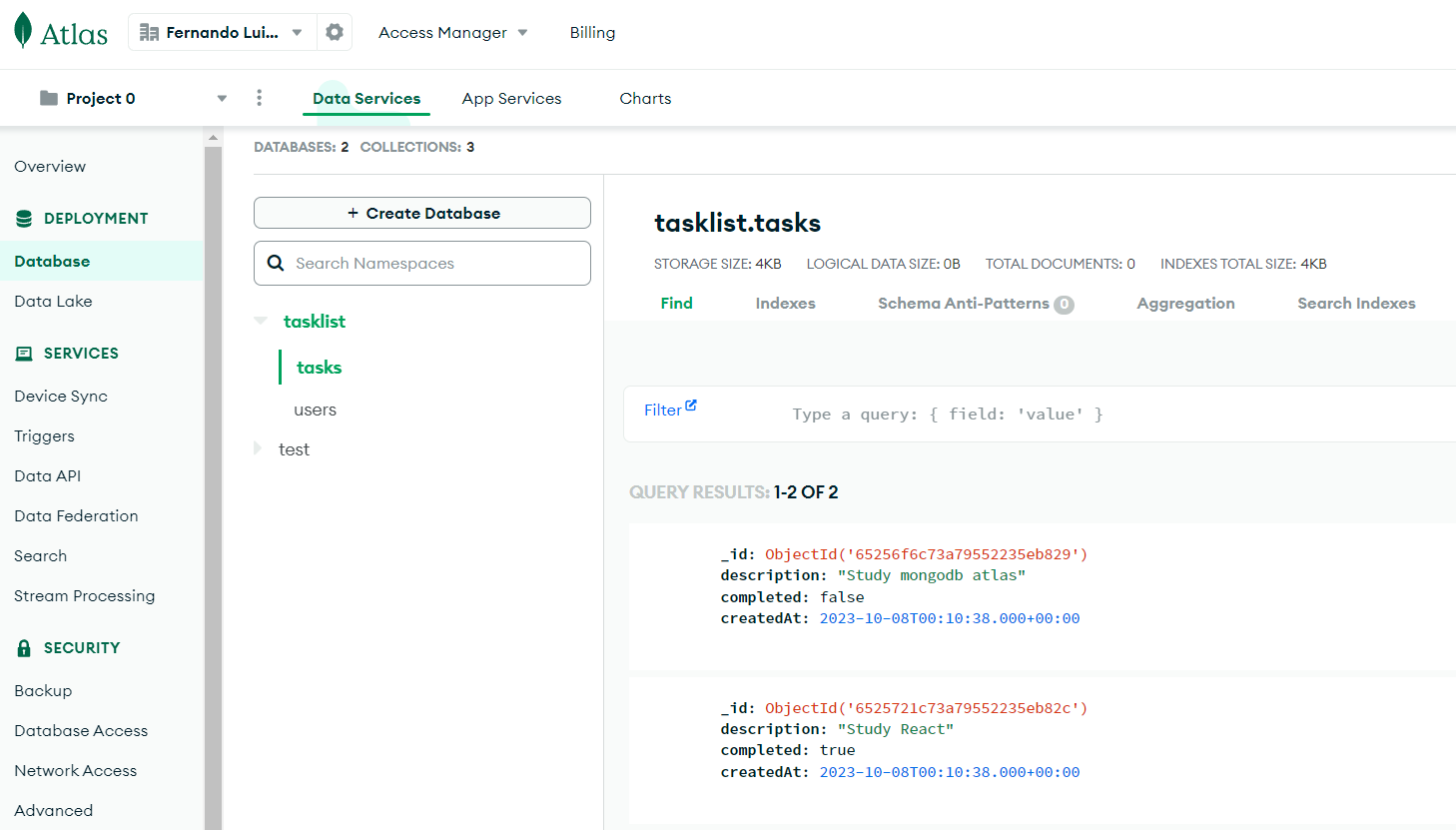
# **Clúster MongoDB Atlas**

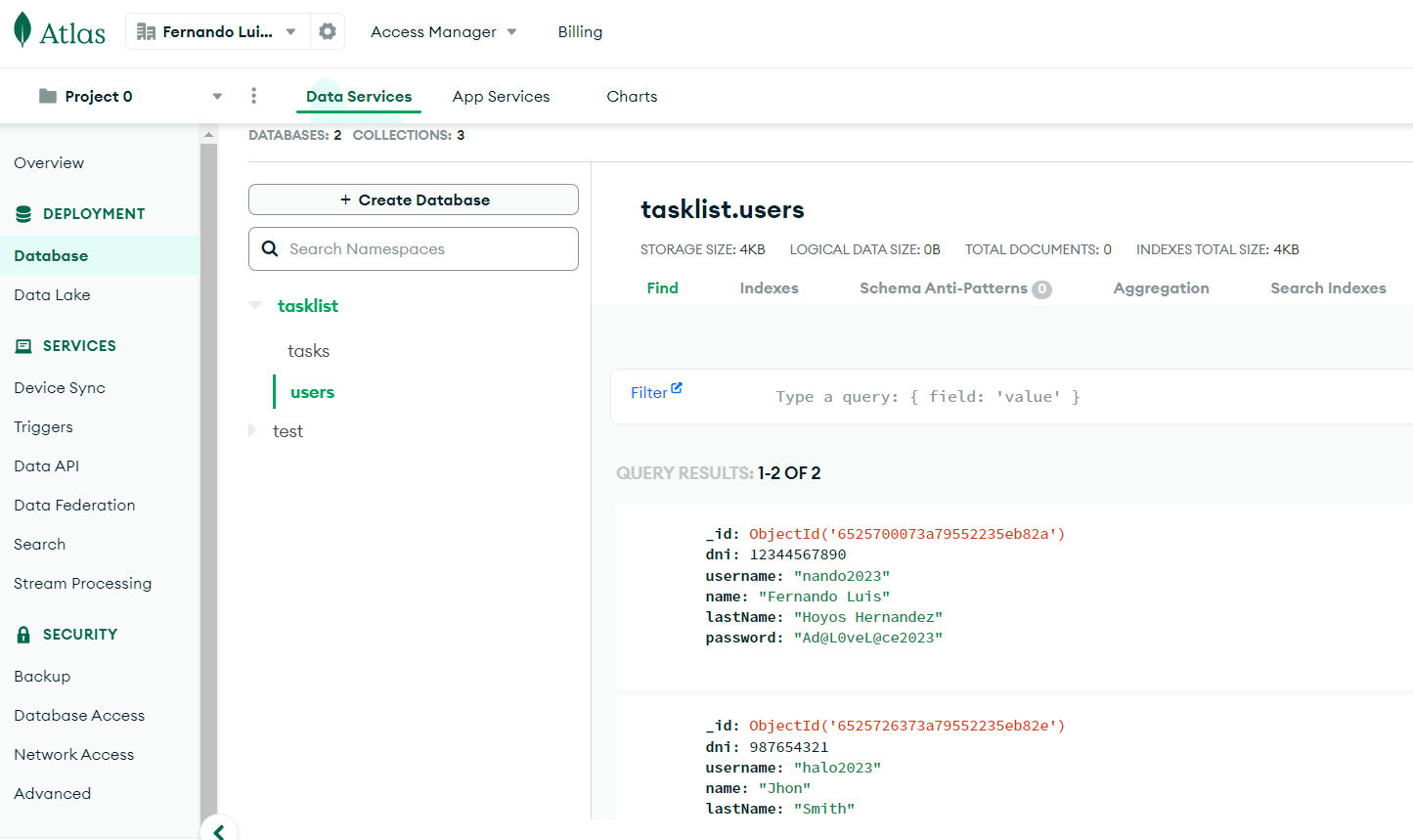
**Crea un documento con Google Docs, donde puedas adjuntar los pantallazos desde MongoDB Atlas, donde se vean tus colecciones y el documento con sus datos de ejemplo para cada una.**

Nombre de la base de datos desde mongodb atlas: **tasklist**

**Pantalla de documentos de la colección de tasks**

****

**Pantalla de documentos de la colección de users**

****

**Para cada colección añade en el documento de Google Docs como serían los comandos para las operaciones del CRUD usando el** [**driver de acuerdo a tu stack**](https://www.mongodb.com/docs/drivers/)**, teniendo en cuenta el documento de ejemplo o el documento creado con el comando Create, para las operaciones de Read, Update y Delete.**

**Operaciones CRUD usando el driver de “mongoose” en node.js**

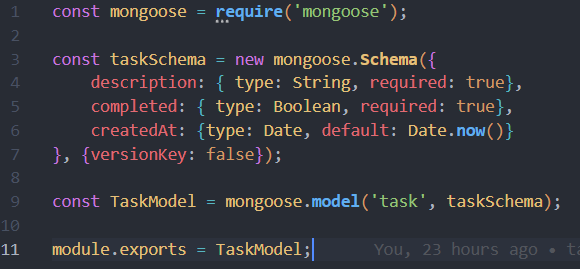
En primer lugar instalamos la librería de mongoose en node.js en nuestro proyecto con el comando:

**npm i mongoose**

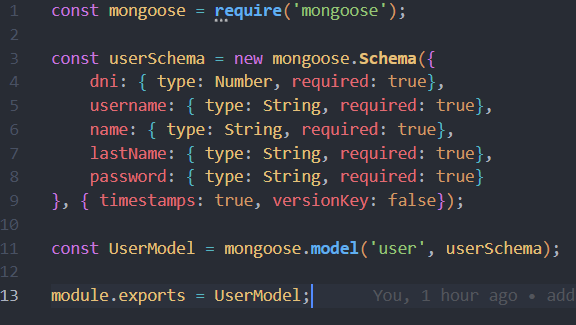
Luego lo que hacemos es crear un archivo llamado **database.js** en donde configuramos la conexión a la base de datos llamada **tasklist** configurada en la URI guardada en las variables de entorno del proyecto:  
  


Ahora creamos los modelos y schemas de las colecciones **tasks** y **users:**

**task.js**

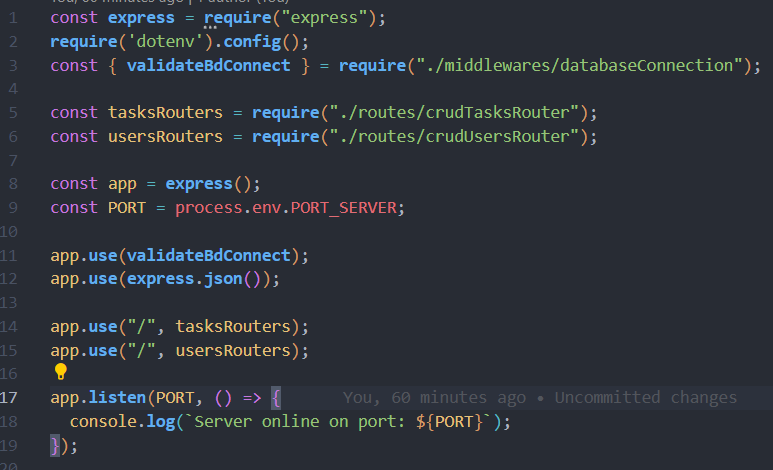


**user.js**

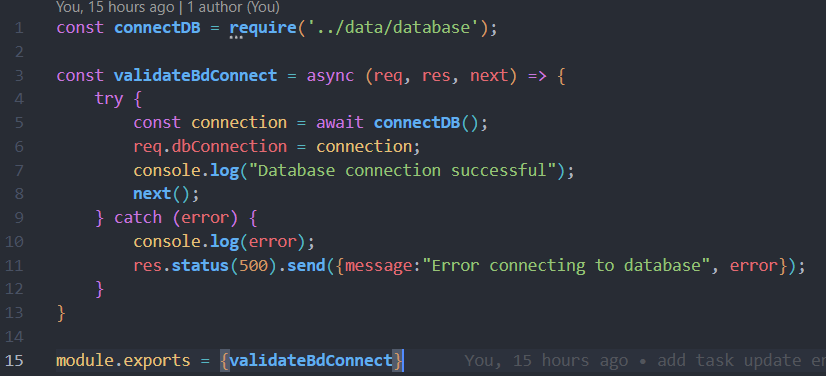
****

Configuramos el servidor en express con las rutas de **tasks** y users, además hacemos uso de un middleware que valida la conexión a la base de datos: **tasklist:**

**app.js**

****

Archivo middleware **databaseConnection.js**

****

**Ejecutando las operaciones CRUD con solicitudes HTTP:**

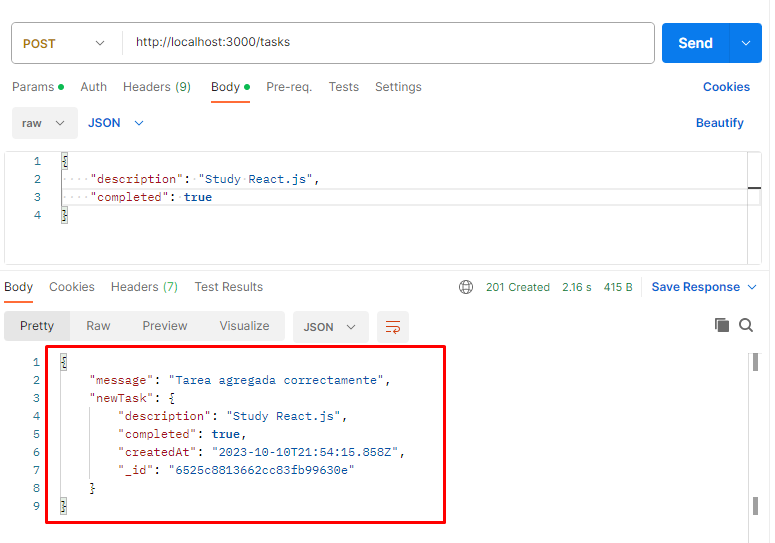
Archivo **crudTasksRouter.js**

**Colección de tasks:**

**Agregando tareas**:

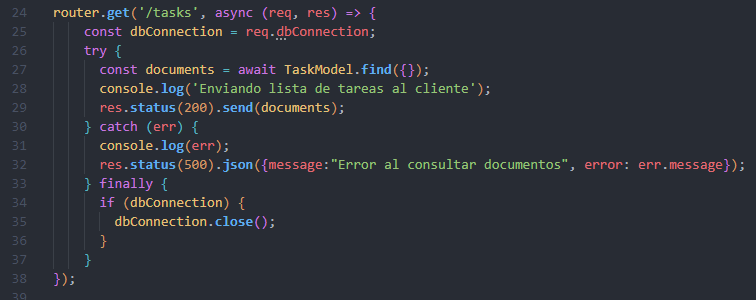
En la solicitud **POST** enviamos los datos de **description** y **completed** a través del body, luego llamamos al modelo **TaskModel** para crear una nueva tarea con los datos del body, luego se ejecuta el método **save()** para guardar los datos en la base de datos en la colección **tasks,** en caso de errores con el proceso estos se capturan y luego cerramos la conexión con la base de datos.



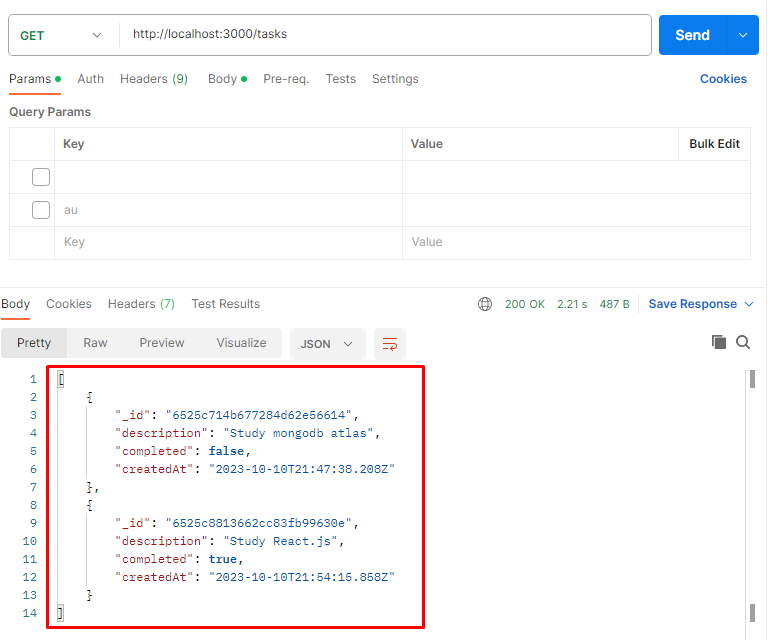
Cuando agregamos una tarea desde **POSTMAN** obtenemos el resultado:  
  


**Consultar tareas:**

Realizamos una petición **GET** en donde utilizamos el modelo **TaskModel** para hacer una búsqueda de todos los documentos existentes en la colección **tasks**. Luego enviamos el resultado de la lista de tareas guardadas en la colección al cliente. En caso de errores se capturan en el bloque catch y al final se cierra la conexión con la base de datos.

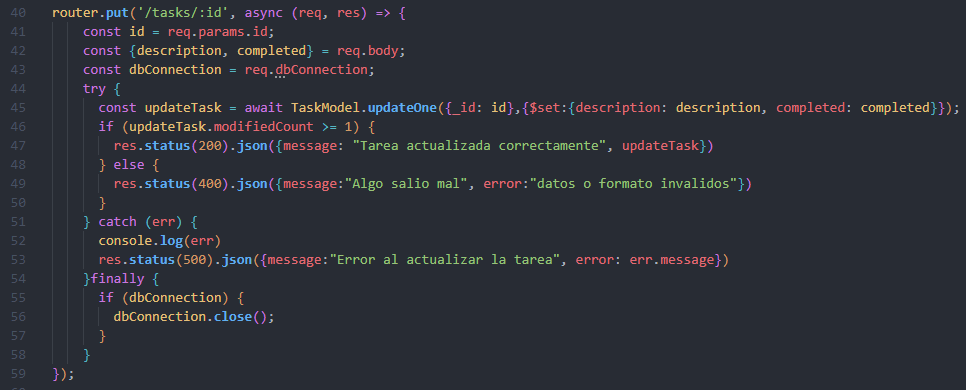


Cuando solicitamos las tareas existentes desde **POSTMAN** obtenemos el resultado:

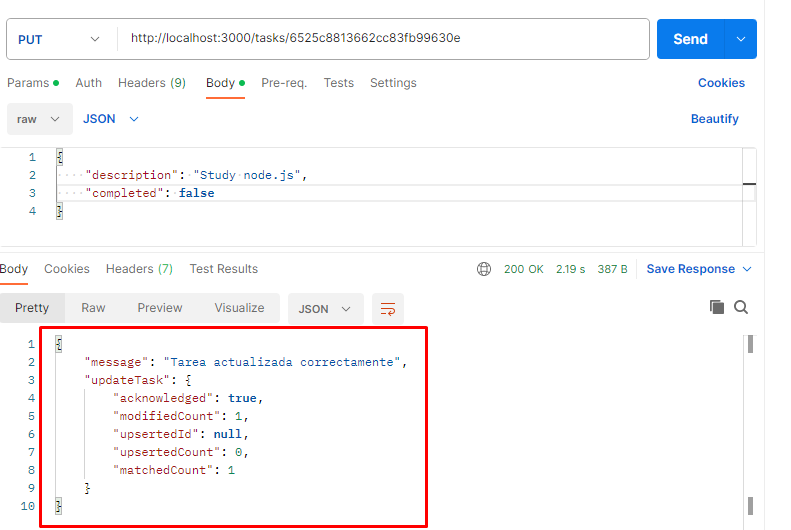


**Actualizar tareas:**

Realizamos una petición **PUT**, capturamos el id en los parametros de la ruta además del body con las propiedades description y completed llamamos a **TaskModel** para que haga una búsqueda con el método **updateOne()** pasándole como parámetro de búsqueda la propiedad **\_id**  y el valor del id del parametro que capturamos en la ruta para buscar el documento que queremos actualizar además del parámetro **&set** con las propiedades que queremos cambiar **description** o **completed** o ambas dándole el valor que necesitemos modificar. Si la operación sale exitosa se modifica el documento y se enviará la respuesta al cliente, en caso de error se captura la excepción en el catch y al final se cierra la conexión de la base de datos.

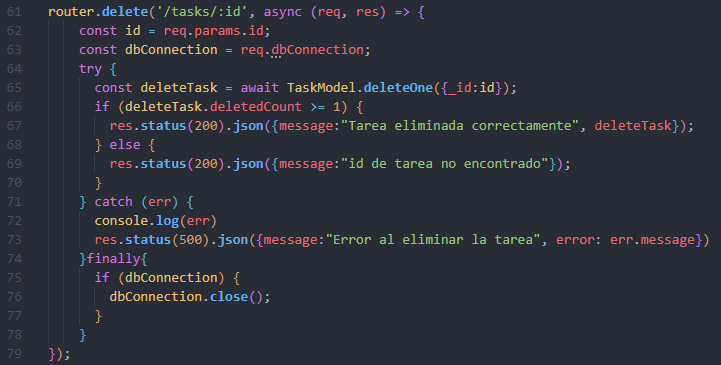


Cuando actualizamos una de las tareas existentes en la base de datos desde **POSTMAN** obtenemos el resultado:

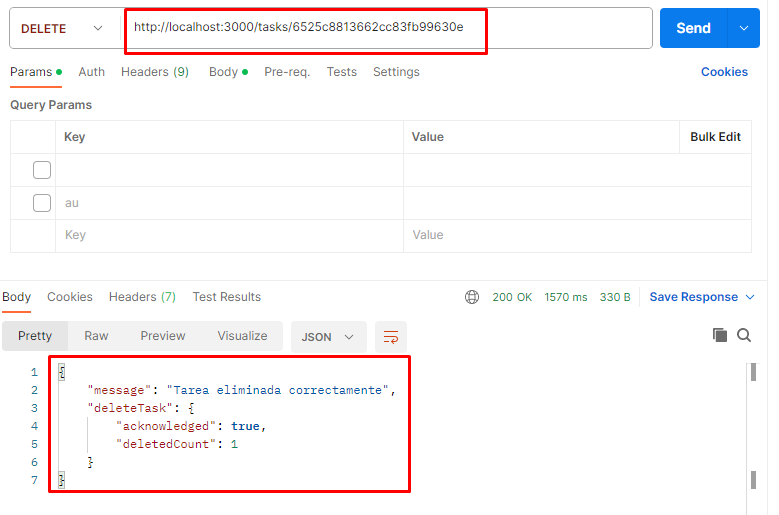


**Eliminar tareas:**

Configuramos una petición **DELETE**, donde capturamos el id recibido en la URL, con el id le pasamos al método **deleteOne(\_id:id)** del modelo **TaskModel** para que busque el documento que corresponda con el objectId a eliminar. Si la operación sale exitosa se envía la respuesta al cliente. En caso de errores estos se capturan en el catch y al final se cierra la conexión con la base de datos.



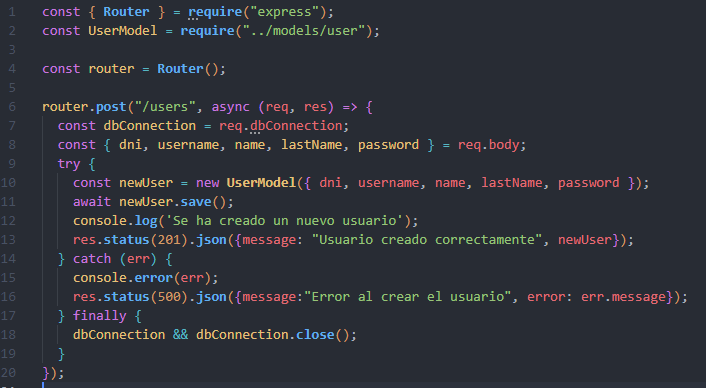
Cuando eliminamos una de las tareas existentes en la base de datos desde **POSTMAN** obtenemos el resultado:

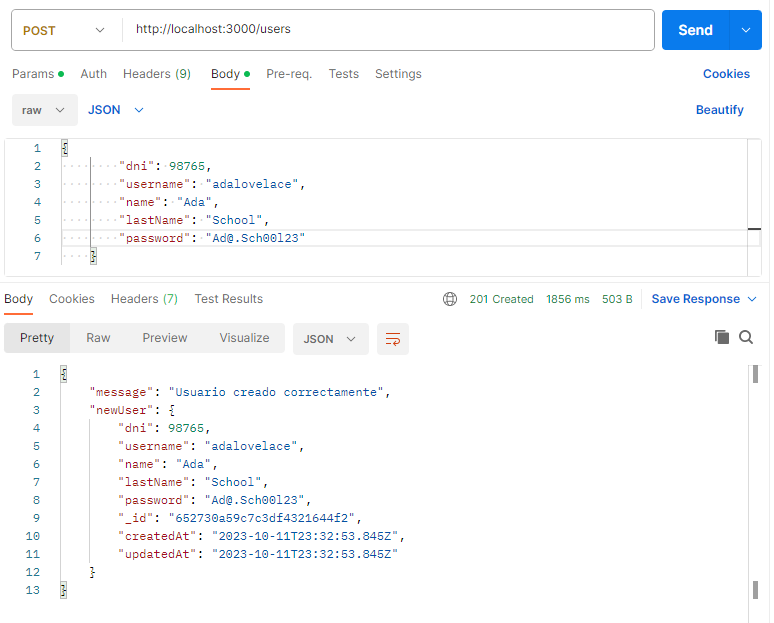


Archivo **crudUsersRouter.js**

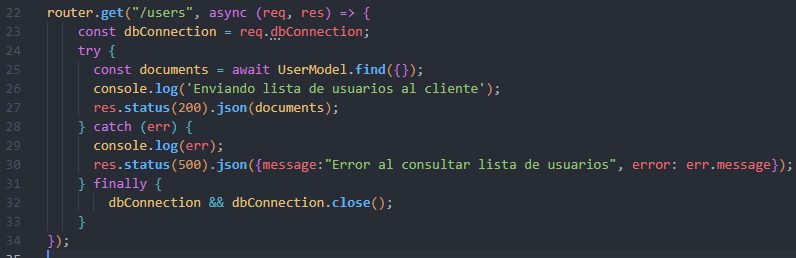
**Colección de Users:**

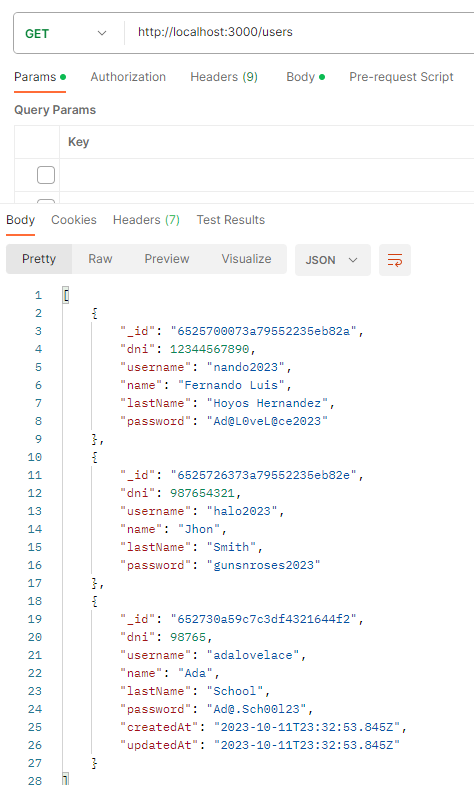
**Agregando usuarios**:



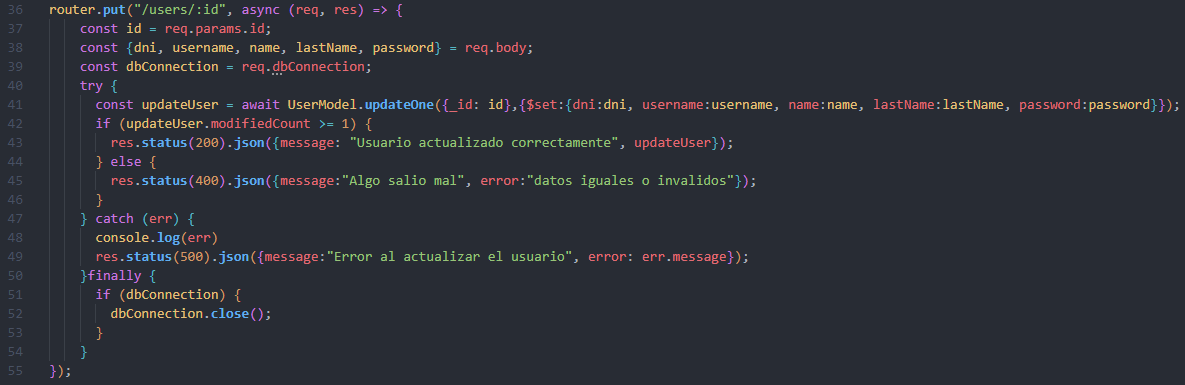


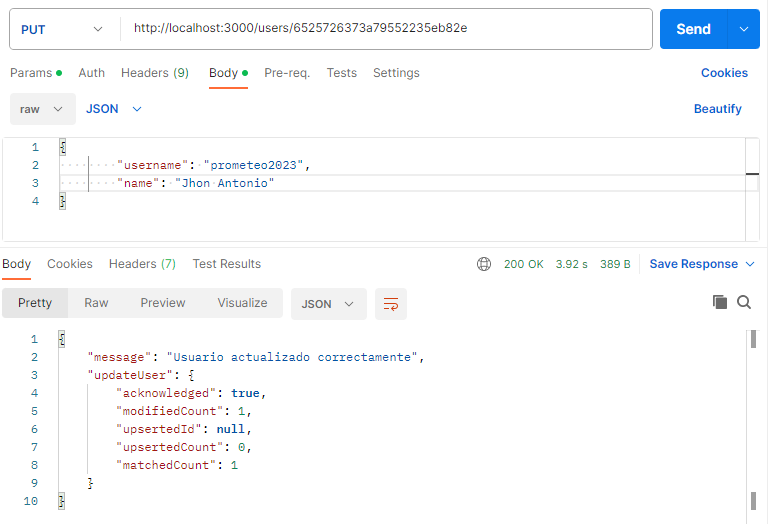
**Consultar usuarios:**



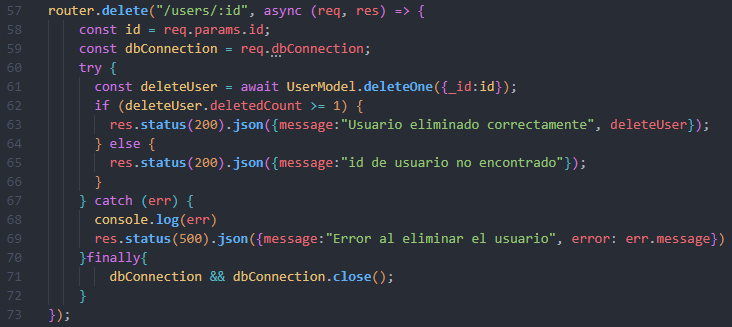


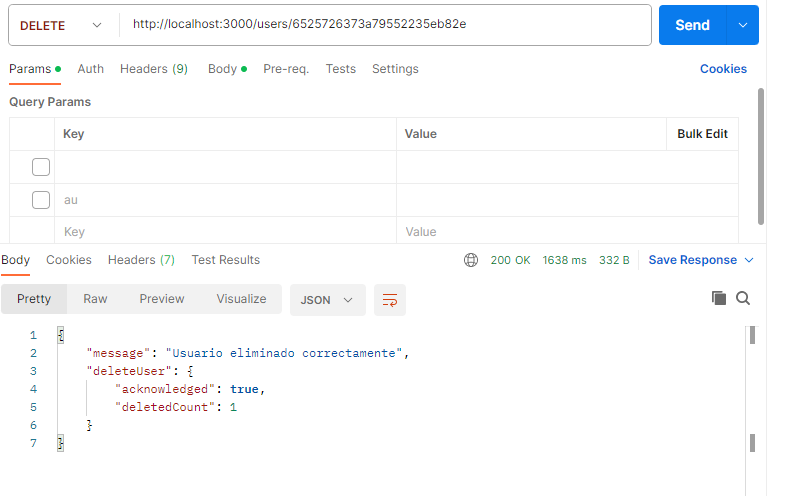
**Actualizando datos de un usuario:**

****

****

**Eliminando un usuario:**

****

****