Contenido

[**Parte 1 – Gestión de Usuarios** 2](#_Toc198067742)

[**Forma 1: Consola MongoShell** 2](#_Toc198067743)

[**Forma 2: Interfaz Mongo** 5](#_Toc198067744)

[**Parte 3: Visual - Node** 9](#_Toc198067745)

[**Parte 2 – Gestión de Productos** 17](#_Toc198067746)

[**Parte 1: Consola MongoShell** 17](#_Toc198067747)

[**Parte 2: Interfaz de Mongo** 20](#_Toc198067748)

[**Parte 3: Visual - Node** 24](#_Toc198067749)

**Taller de Repaso Bases de Datos No Relacionales**

***Mariana Álvarez Carvajal***

Teniendo en cuenta sus conocimientos previos sobre bases de datos NoSQL, realice la siguiente actividad:

Eres parte del equipo de desarrollo de una aplicación de gestión de inventarios y usuarios. Como desarrollador, te han asignado la tarea de crear y manejar bases de datos con MongoDB. En este taller, usarás MongoDB para:

1. Gestionar una base de datos de usuarios.

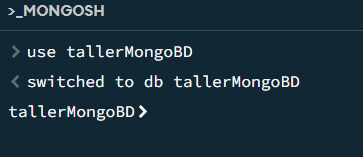
2. Administrar una colección de productos.

3. Realizar consultas, actualizaciones y análisis de datos.

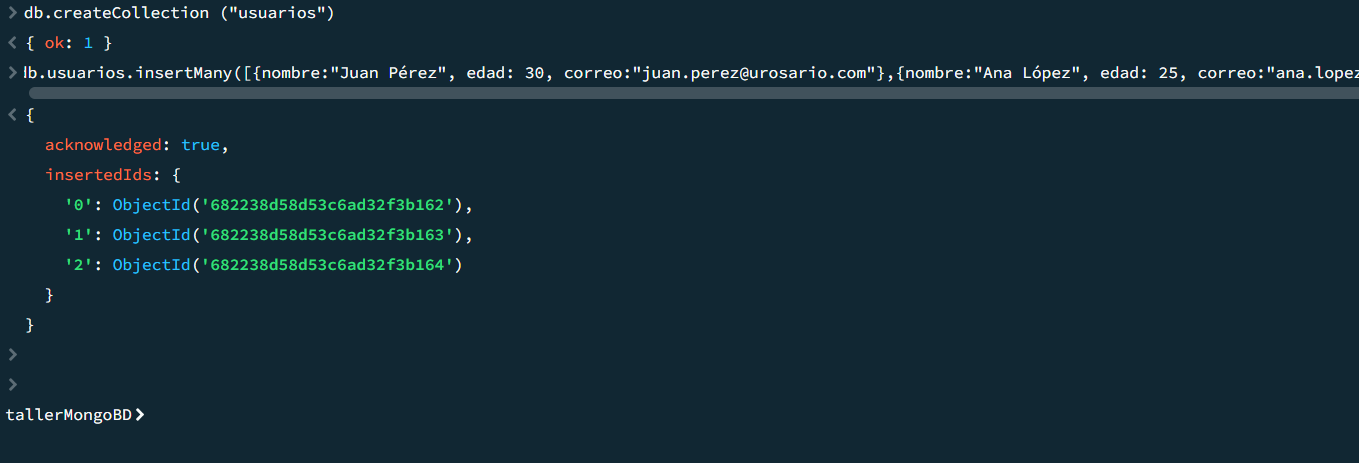
# **Parte 1 – Gestión de Usuarios**

## **Forma 1: Consola MongoShell**

1. Crea una base de datos llamada tallerMongoDB



1. Inserta la siguiente información en una colección llamada usuarios:



1. Realiza las siguientes consultas:
   1. Encuentra todos los usuarios



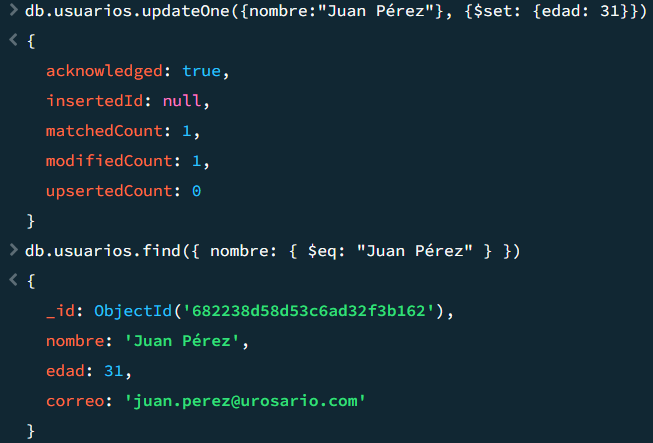
* 1. Encuentra el usuario con nombre Ana López



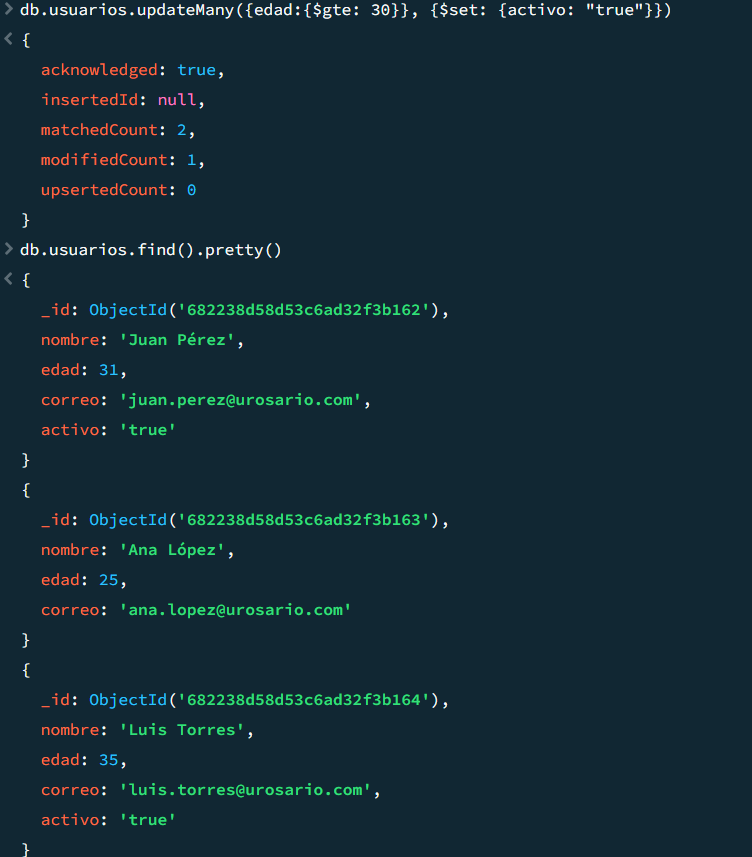
* 1. Encuentra todos los usuarios mayores o iguales a 30 años



1. Actualiza los datos:
   1. Cambia la edad de Juan Pérez a 31 años



* 1. Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años.



1. Elimina registros:
   1. Elimina el usuario Luis Torres

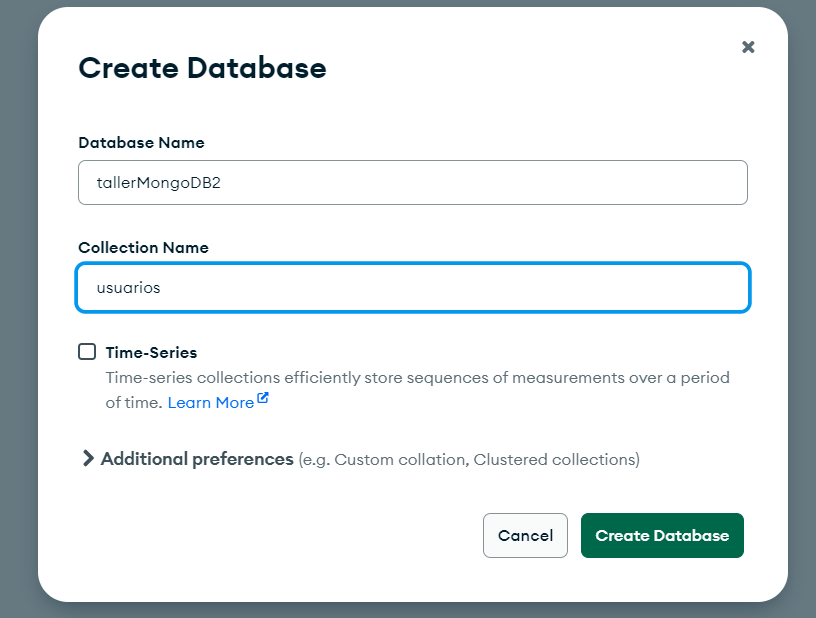


* 1. Elimina todos los usuarios menores de 30 años.

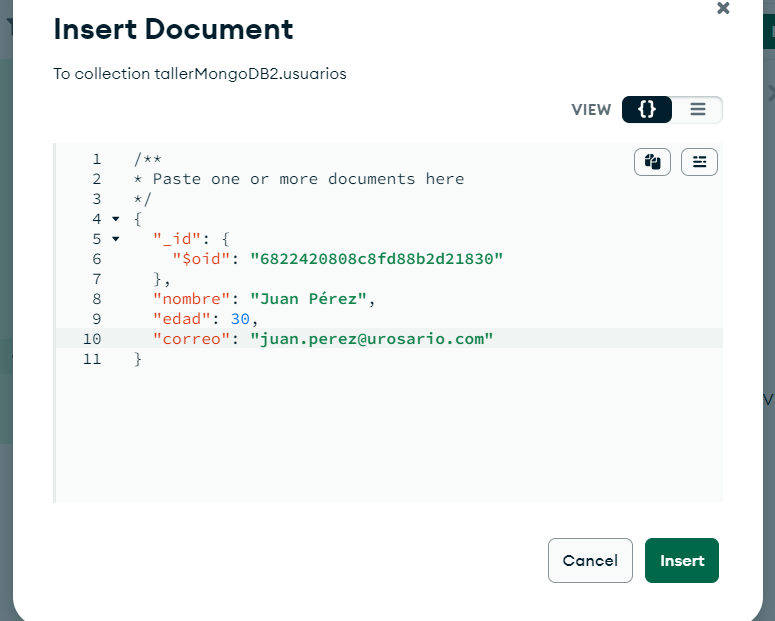


## **Forma 2: Interfaz Mongo**

1. Crea una base de datos llamada tallerMongoDB2

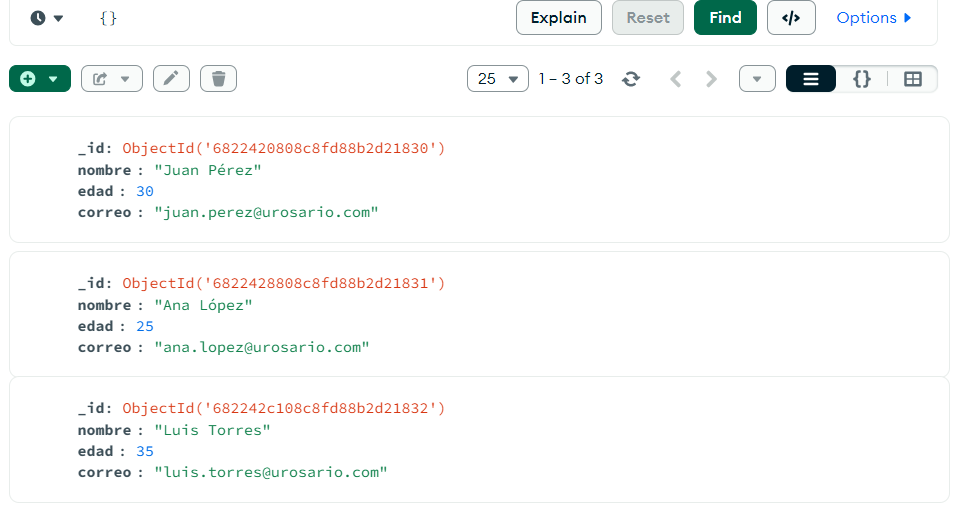


1. Inserta la siguiente información en una colección llamada usuarios

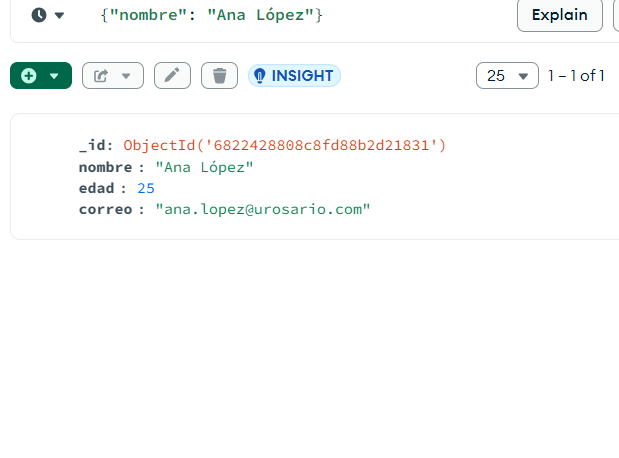




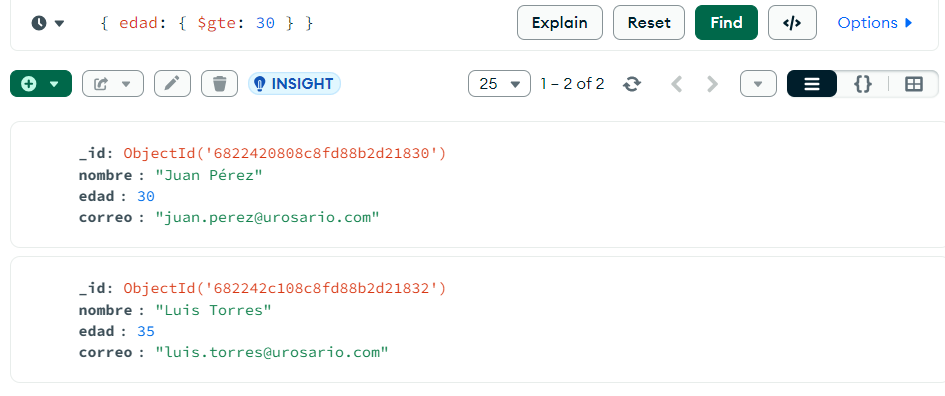
1. Realiza las siguientes consultas:
   1. Encuentra todos los usuarios



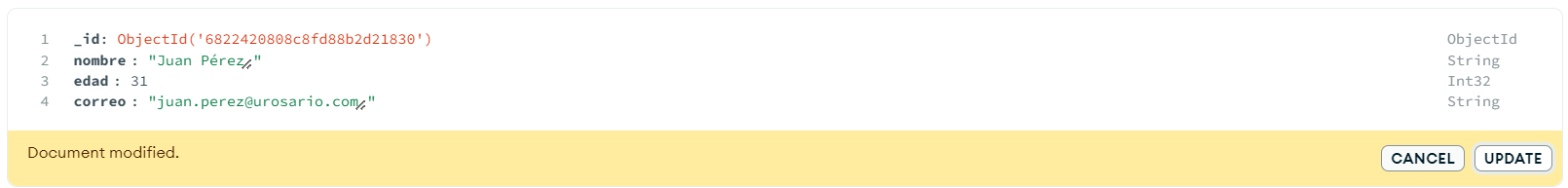
* 1. Encuentra el usuario con nombre Ana López

****

* 1. Encuentra todos los usuarios mayores o iguales a 30 años.

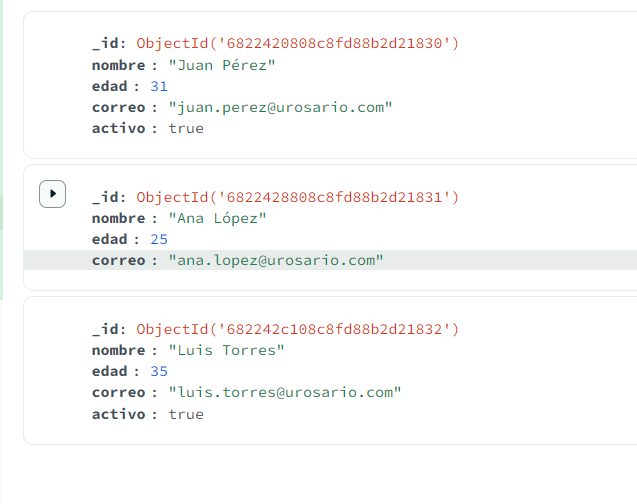
****

1. Actualiza los datos
   1. Cambia la edad de Juan Pérez a 31 años

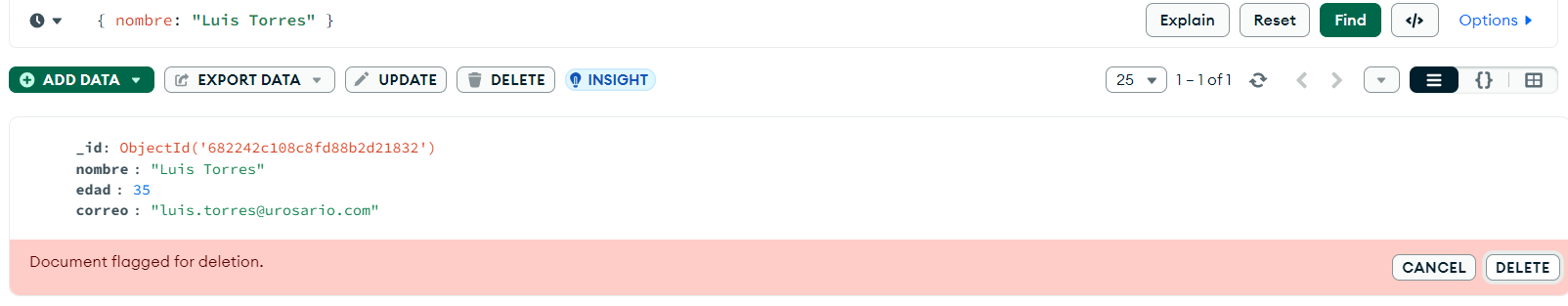


* 1. Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años



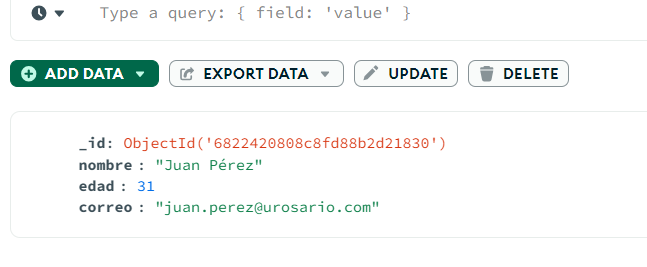


1. Elimina registros
   1. Elimina el usuario Luis Torres



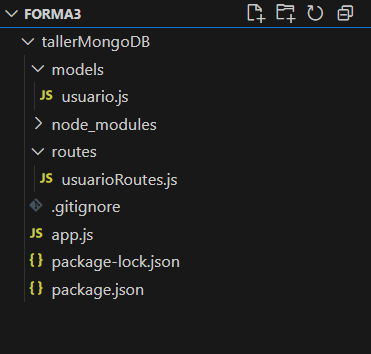


* 1. Elimina todos los usuarios menores de 30 años.



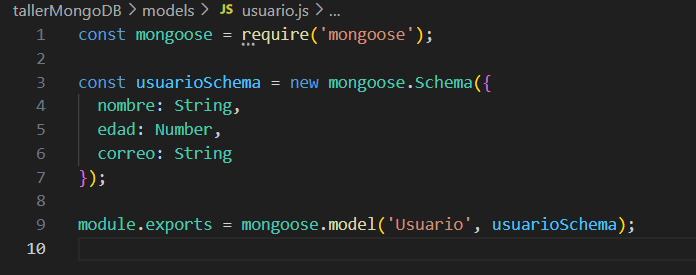
## **Parte 3: Visual - Node**

1. Crea una base de datos llamada tallerMongoDB

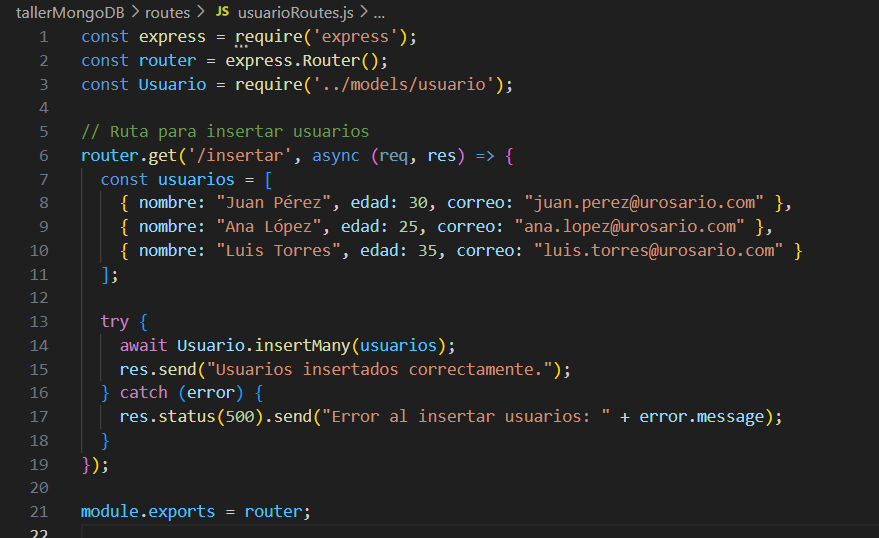


1. Inserción de Usuarios:

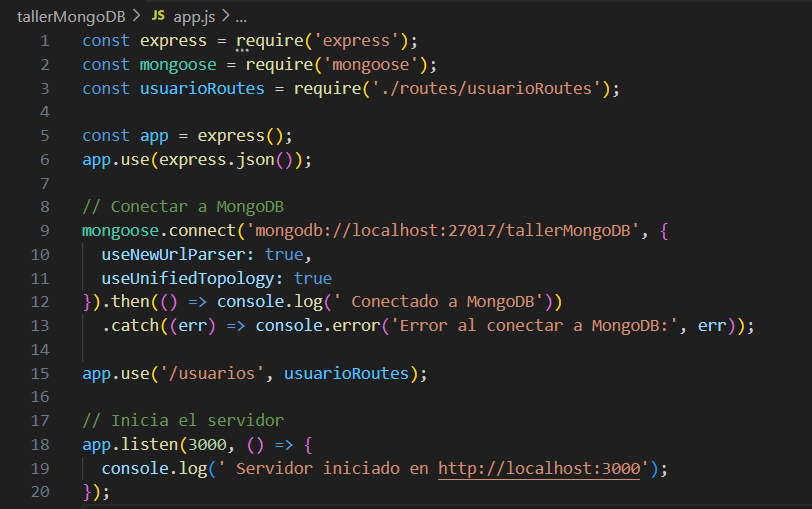
Modelos: usuarios.js

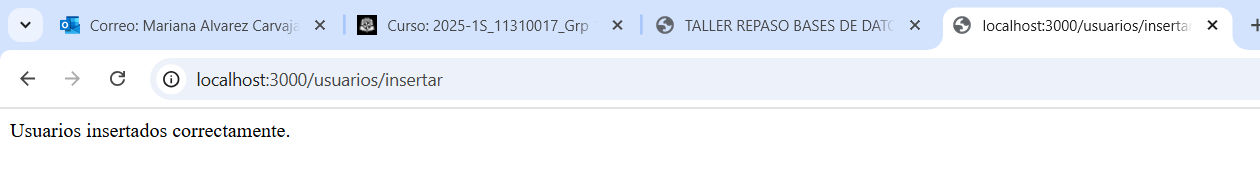


Routes: usuarioRoutes.js

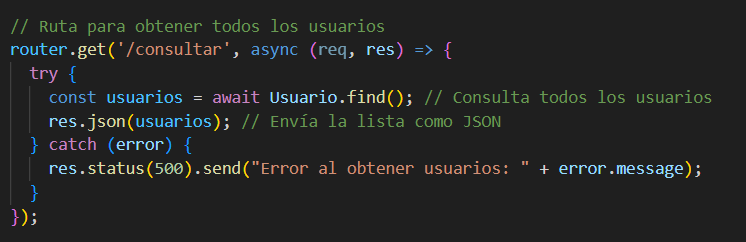


App.js:



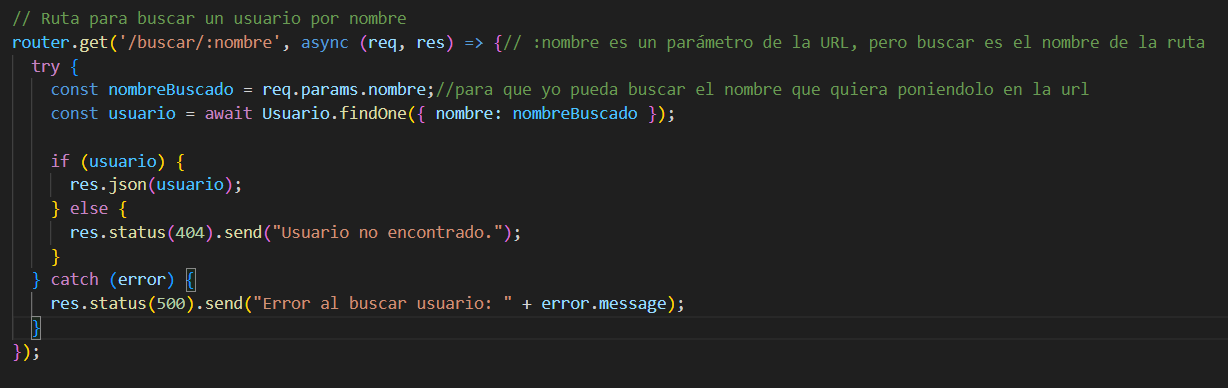


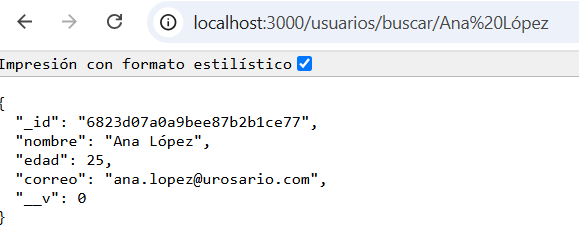
1. Realiza las siguientes consultas (Creo las rutas necesarias para cada consulta en usuarioRoutes.js):
   1. Encuentra todos los usuarios.



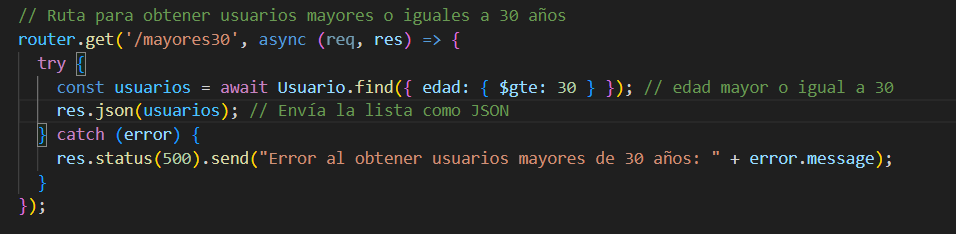


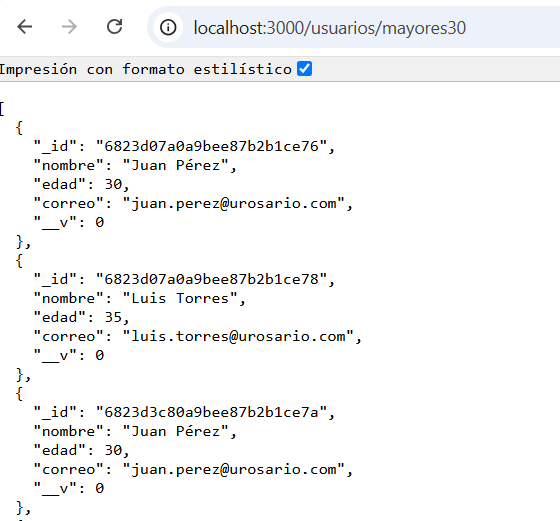
* 1. Encuentra el usuario con nombre Ana López: Hice una ruta general para buscar por nombre, en la URL se pone %20 para hacer el espacio del nombre y apellido



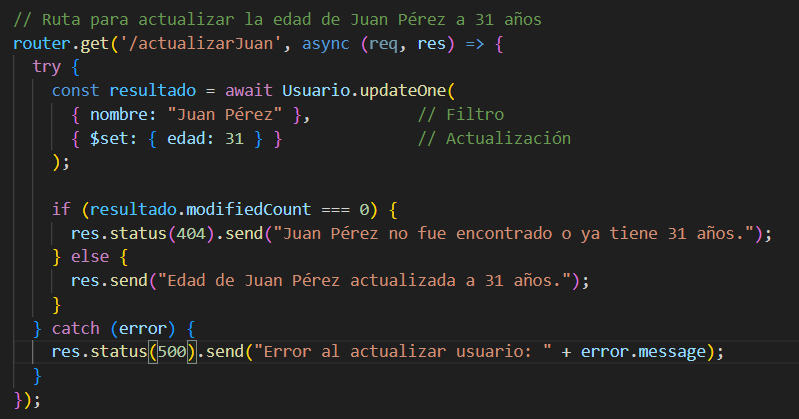


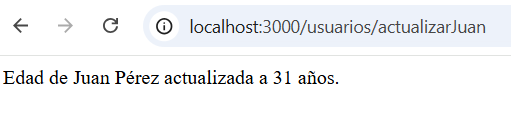
* 1. Encuentra todos los usuarios mayores o iguales a 30 años.

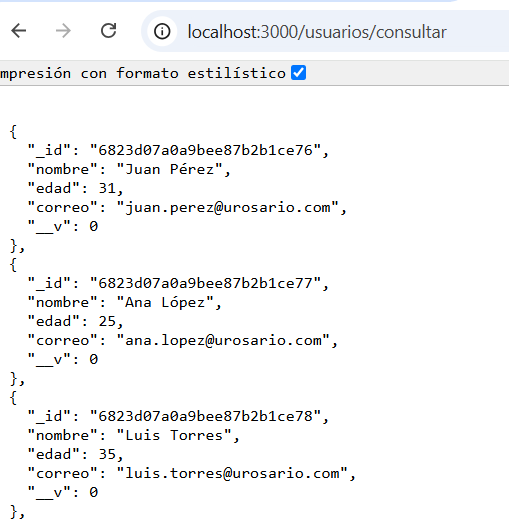




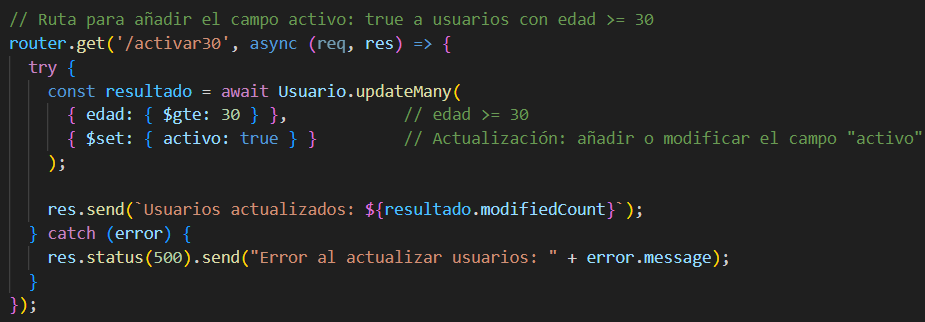
1. Actualiza los datos:
   1. Cambia la edad de Juan Pérez a 31 años

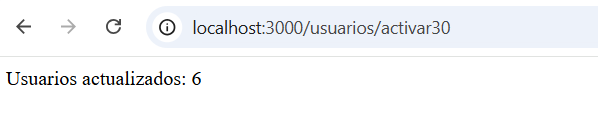






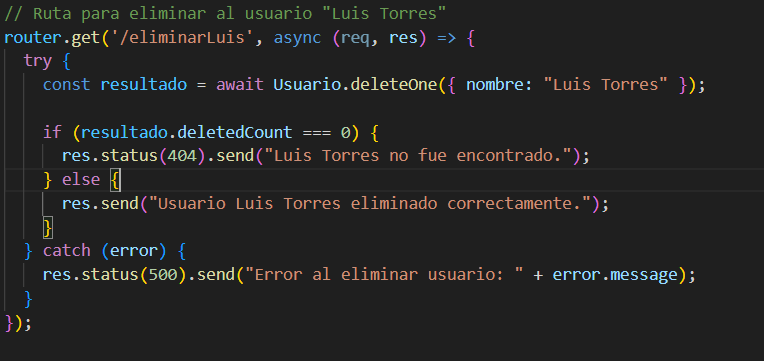
* 1. Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años. Cambio el modelo para incluir ese campo activo

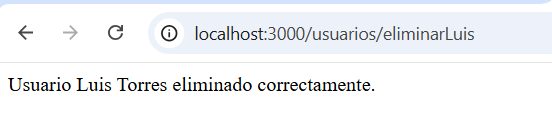


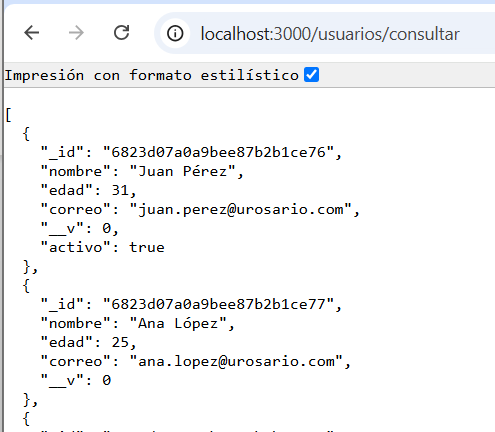




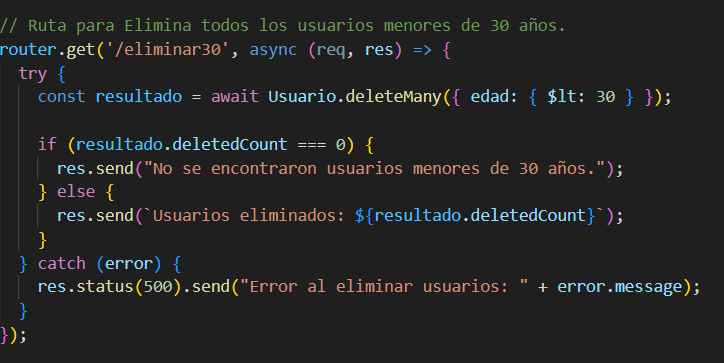
1. Elimina registros
   1. Elimina el usuario Luis Torres

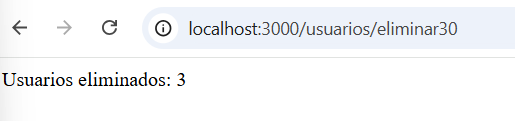






* 1. Elimina todos los usuarios menores de 30 años.







# **Parte 2 – Gestión de Productos**

## **Parte 1: Consola MongoShell**

1. Crea una nueva colección llamada productos e inserta al menos 5 productos con los siguientes campos:

• nombre: Nombre del producto.

• precio: Precio del producto.

• categoria: Categoría del producto (por ejemplo: “Electrónica”, “Hogar”, “Ropa”).





1. Realiza las siguientes consultas:
   1. Encuentra todos los productos con precio mayor a $100.



* 1. Ordena los productos por precio de manera descendente



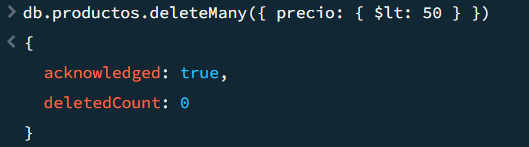
1. Actualiza los datos:
   1. Añade un campo en stock con valor true a todos los productos.



* 1. Cambia el valor de en stock a false para los productos cuyo precio sea mayor a $500.



1. Elimina Registros: Elimina todos los productos cuyo precio sea menor a $50.



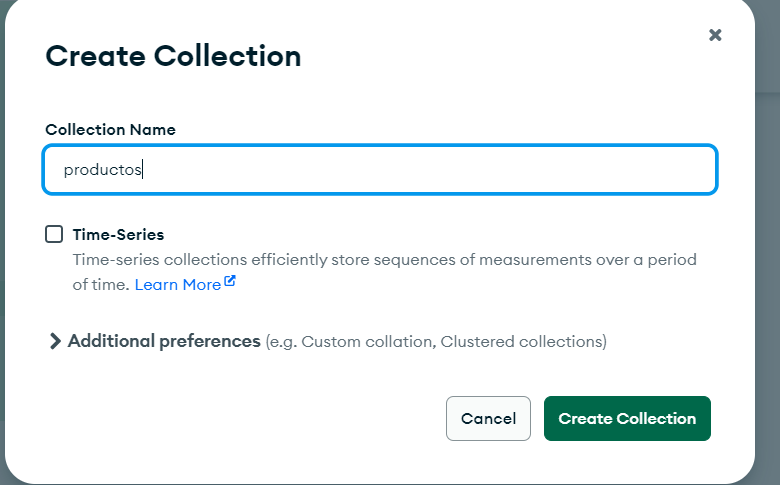
## **Parte 2: Interfaz de Mongo**

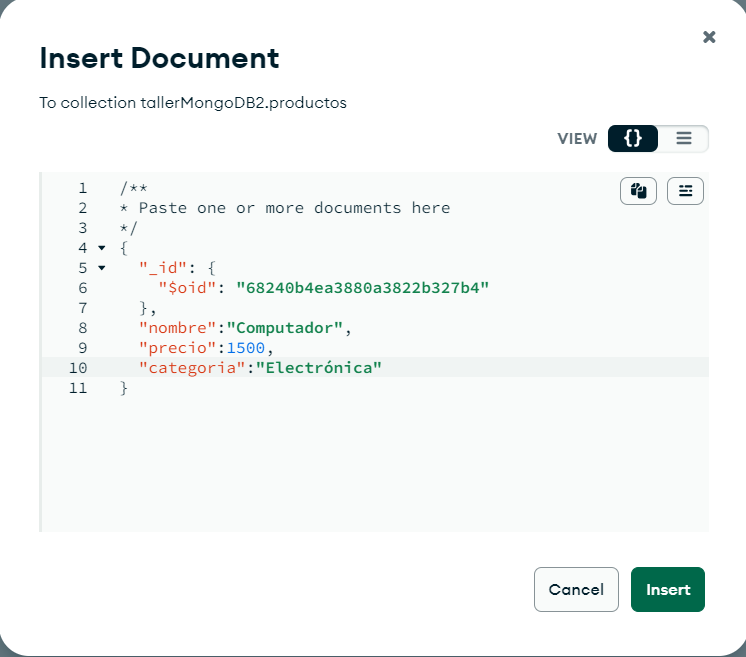
1. Crea una nueva colección llamada productos e inserta al menos 5 productos con los siguientes campos:

• nombre: Nombre del producto.

• precio: Precio del producto.

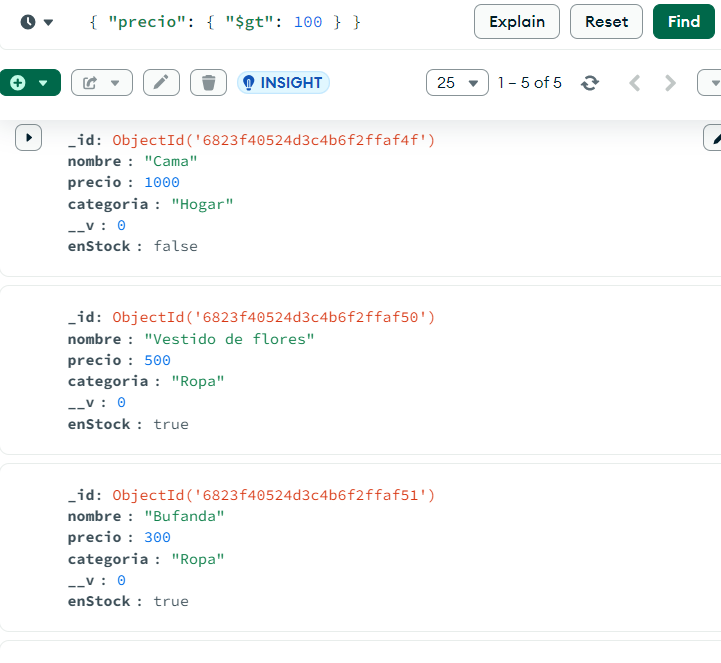
• categoria: Categoría del producto (por ejemplo: “Electrónica”, “Hogar”, “Ropa”).



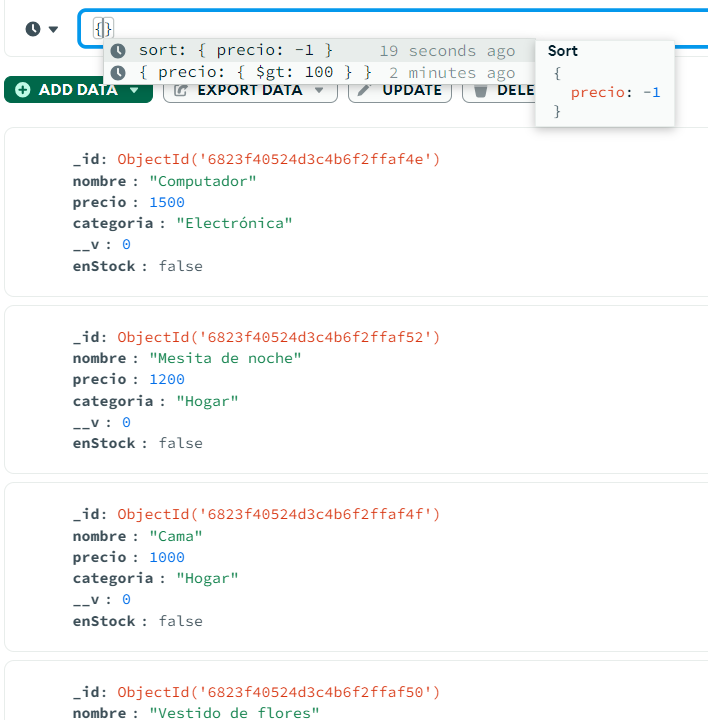




1. Realiza las siguientes consultas
   1. Encuentra todos los productos con precio mayor a $100

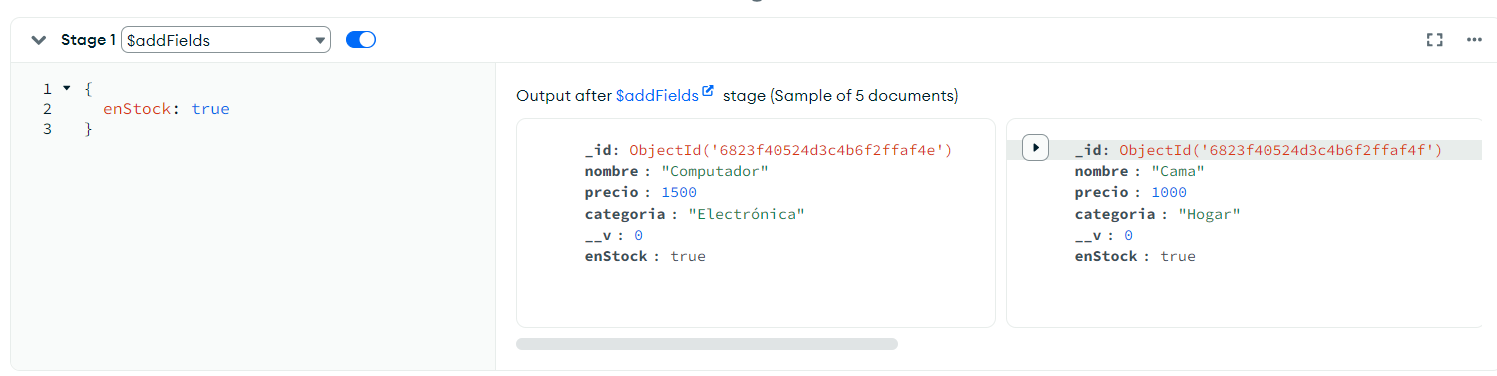


* 1. Ordena los productos por precio de manera descendente

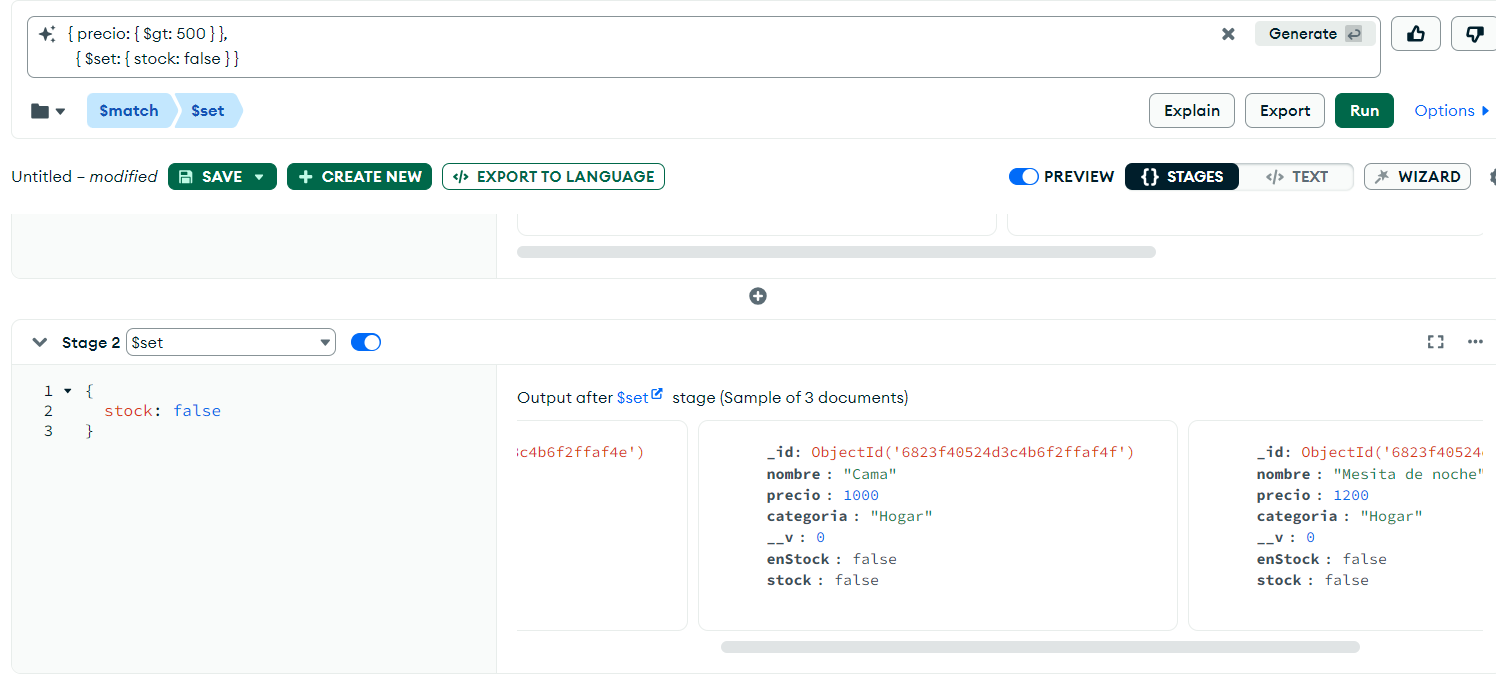




1. Actualiza los datos:
   1. Añade un campo en stock con valor true a todos los productos.



* 1. Cambia el valor de en stock a false para los productos cuyo precio sea mayor a $500.



1. Elimina registros: Elimina todos los productos cuyo precio sea menor a $50



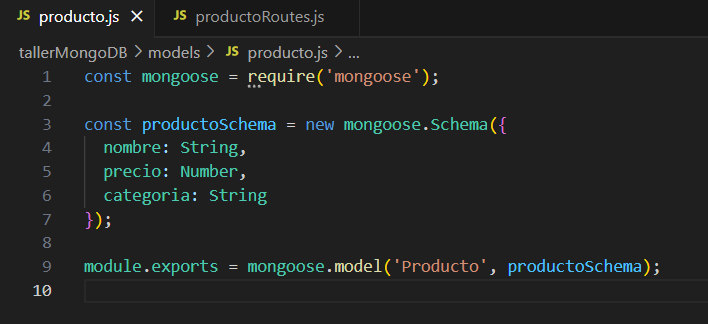
## **Parte 3: Visual - Node**

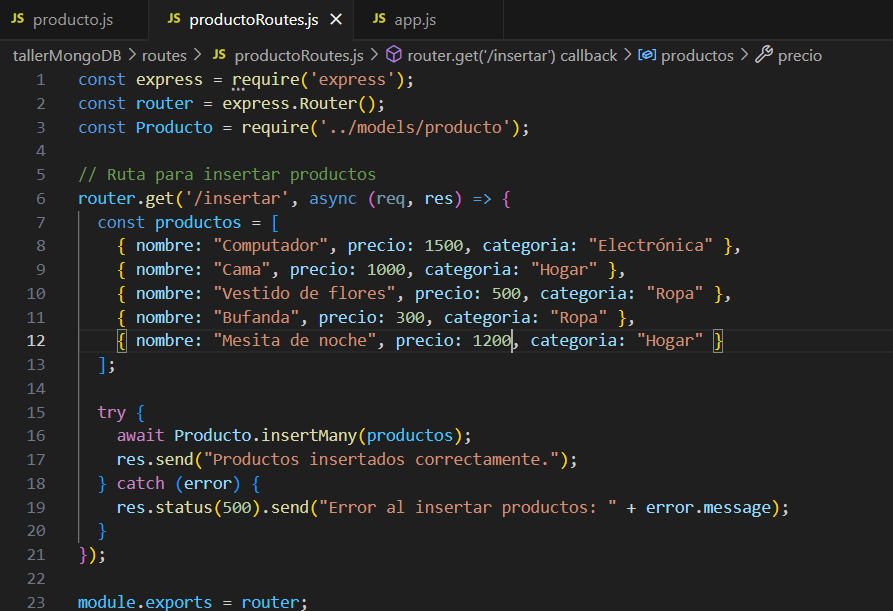
1. Crea una nueva colección llamada productos e inserta al menos 5 productos con los siguientes campos:

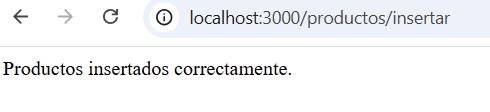
• nombre: Nombre del producto.

• precio: Precio del producto.

• categoria: Categoría del producto (por ejemplo: “Electrónica”, “Hogar”, “Ropa”).

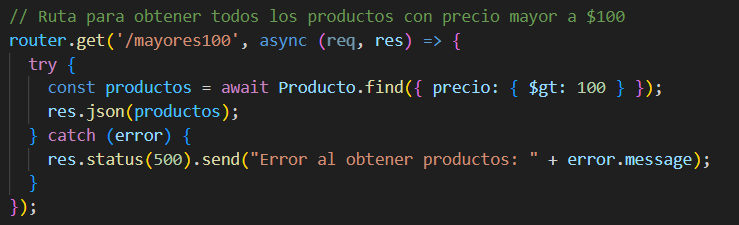








1. Realiza las siguientes consultas
   1. Encuentra todos los productos con precio mayor a $100

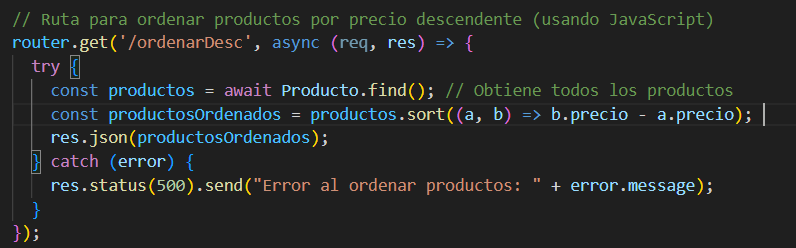




* 1. Ordena los productos por precio de manera descendente

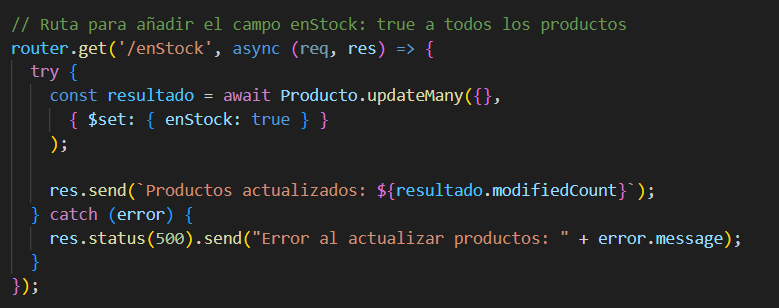
<https://stackoverflow.com/questions/52030110/sorting-strings-in-descending-order-in-javascript-most-efficiently>

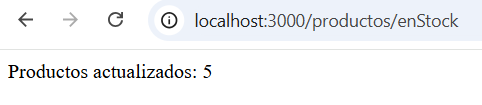
“Para ordenar datos de forma descendente en Node.js, se utiliza el método sort() de los arreglos (arrays) y una función de comparación personalizada. Esta función compara dos elementos y determina su orden en la lista ordenada. Se usa la función sort((a, b) => b - a) para ordenar números de mayor a menor, y sort((a, b) => b.localeCompare(a)) para cadenas de texto.”





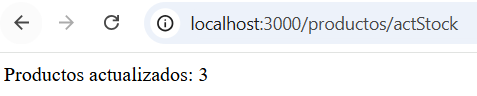
1. Actualiza los datos:
   1. Añade un campo en stock con valor true a todos los productos.

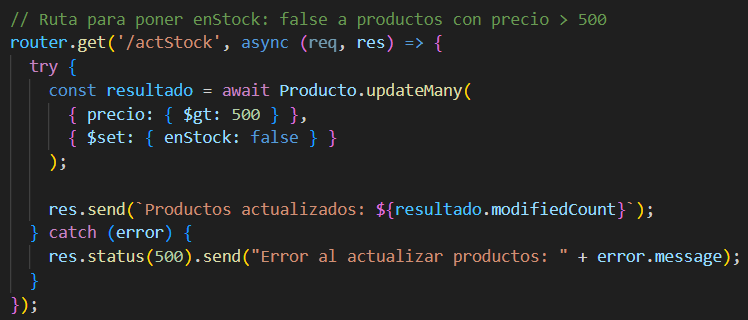






* 1. Cambia el valor de en stock a false para los productos cuyo precio sea mayor a $500.







1. Elimina registros
   1. Elimina todos los productos cuyo precio sea menor a $50

