**Tarea**

1. **Parte 1: Gestión de Usuarios**

\*Shell

1. Crea una base de datos llamada tallerMongoDB.

use tallerMongoDB2

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

db.createCollection("usuarios")

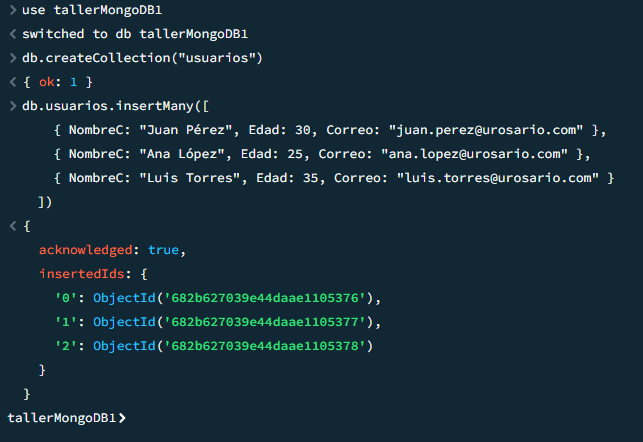
db.usuarios.insertMany([

   { NombreC: "Juan Pérez", Edad: 30, Correo: "juan.perez@urosario.com" },

   { NombreC: "Ana López", Edad: 25, Correo: "ana.lopez@urosario.com" },

   { NombreC: "Luis Torres", Edad: 35, Correo: "luis.torres@urosario.com" }

])



3. Realiza las siguientes consultas:

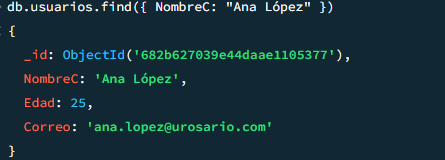
• Encuentra todos los usuarios:

db.usuarios.find()



• Encuentra el usuario con nombre Ana López:

db.usuarios.find({ NombreC: "Ana López" })



• Encuentra todos los usuarios mayores o iguales a 30 años:

db.usuarios.find({ Edad: { $gte: 30 } })



4. Actualiza los datos:

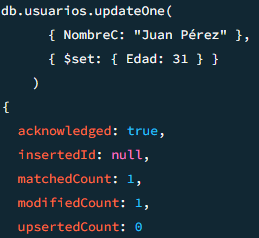
• Cambia la edad de Juan Pérez a 31 años:

db.usuarios.updateOne(

   { NombreC: "Juan Pérez" },

   { $set: { Edad: 31 } }

)



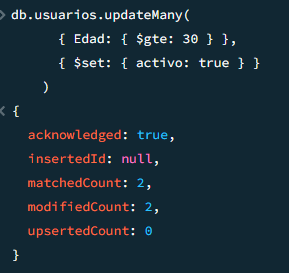
• Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años:

db.usuarios.updateMany(

   { Edad: { $gte: 30 } },

   { $set: { activo: true } }

)



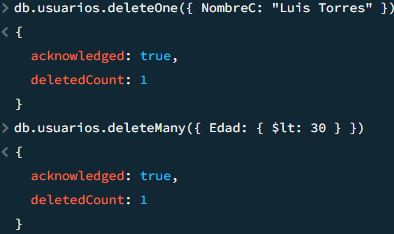
5. Elimina registros:

• Elimina el usuario Luis Torres:

db.usuarios.deleteOne({ NombreC: "Luis Torres" })

• Elimina todos los usuarios menores de 30 años:

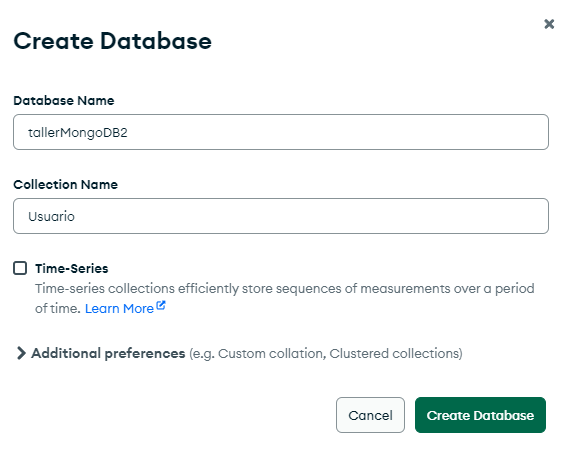
db.usuarios.deleteMany({ Edad: { $lt: 30 } })



\*Interfaz

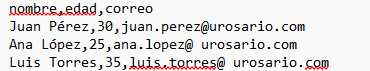
1. Crea una base de datos llamada tallerMongoDB.

-crear nueva base en el +



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

-Al crear la bd agregar la colección "Usuario" e importar los documentos



3. Realiza las siguientes consultas:

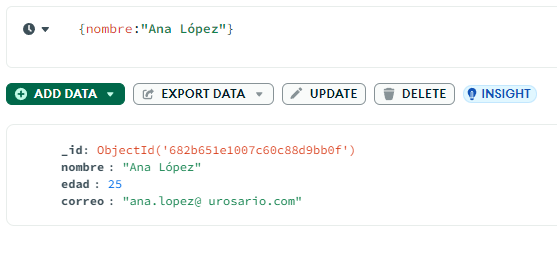
• Encuentra todos los usuarios:

Al entrar a la colección ya aparecen todos los usuarios



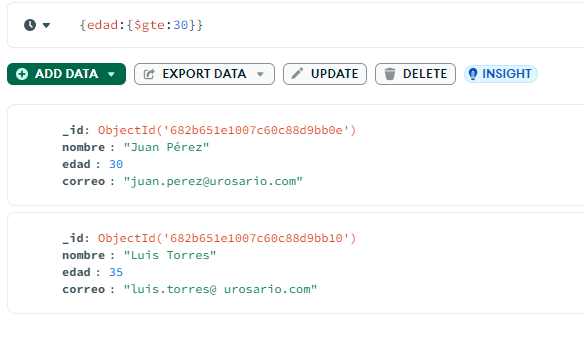
• Encuentra el usuario con nombre Ana López:

{Nombre:"Ana López"}



• Encuentra todos los usuarios mayores o iguales a 30 años:

 {Edad:{$gte:30}}



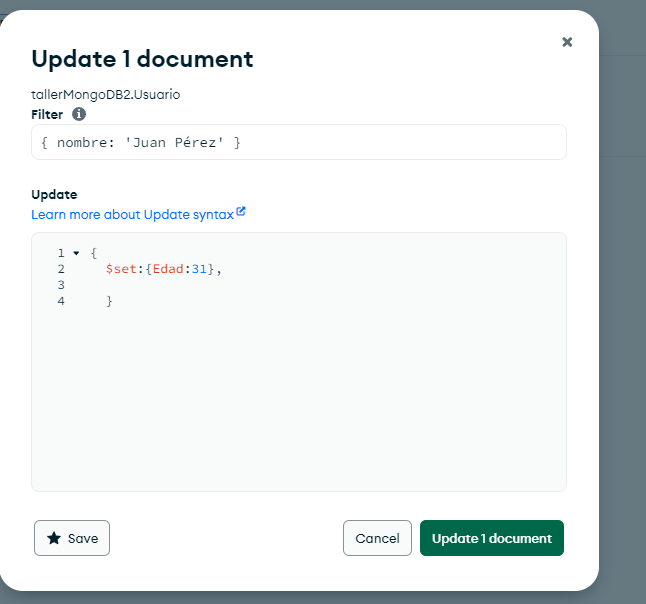
4. Actualiza los datos:

• Cambia la edad de Juan Pérez a 31 años:

colocar la condición {

   $set: {Edad:31},

}



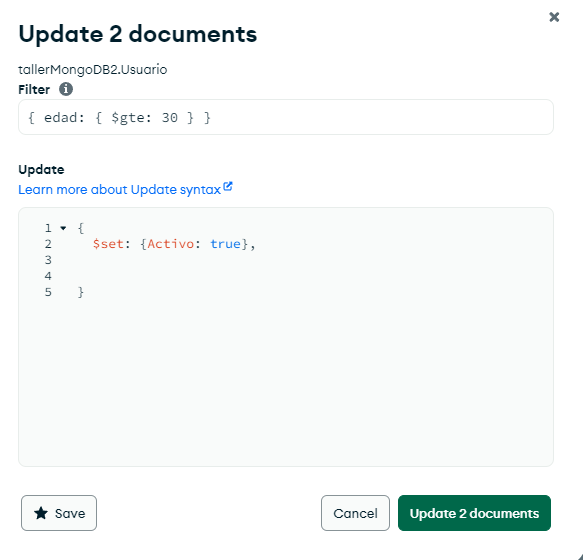
• Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años:

colocar la condición

{

   $set: {Activo: true},

}





5. Elimina registros:

• Elimina el usuario Luis Torres.

-colocar la condición y borrar

{

   NombreC: "Luis Torres"

}

• Elimina todos los usuarios menores de 30 años.

-colocar la condición y borrar

{

Edad: { "$lt": 30 }

}



\*node.js

1. Crea una base de datos llamada tallerMongoDB3.

mongoose.connect('mongodb://localhost:27017/tallerMongoDB3',{

     useNewUrlParser:true,

     useUnifiedTopology:true,

})

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

-Crear  la colección "Usuario"

const mongoose=require('mongoose');

const usuarioSchema=new mongoose.Schema({

    nombre:{type:String,require:true},

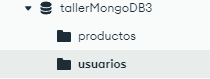
    edad:{type:Number,require:true},

    correo:{type:String,require:true},

    creadoEn:{type:Date,default:Date.now}

});

module.exports=mongoose.model('usuario',usuarioSchema)



- Inserción los documentos

POSTMAN (<http://localhost:3000/api/Usuario>)

{

“nombre”: “Juan Pérez”,

“edad”:30,

“correo”: “juan.perez@urosario.com”

}

{

“nombre”: “Ana Lopez”,

“edad”:25,

“correo”: “ana.lopez@ urosario.com”

}

{

“nombre”: “Luis Torres”,

“edad”:35,

“correo”: “[luis.torres@urosario.com](mailto:luis.torres@urosario.com)”

}



3. Realiza las siguientes consultas:

• Encuentra todos los usuarios: <http://localhost:3000/api/Usuario>

• Encuentra el usuario con nombre Ana López:

• Encuentra todos los usuarios mayores o iguales a 30 años:

4. Actualiza los datos:

• Cambia la edad de Juan Pérez a 31 años:

colocar la condición

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

5. Elimina registros:

• Elimina el usuario Luis Torres.

• Elimina todos los usuarios menores de 30 años.

PARTE 2

Parte 2: Gestión de Productos

\*Shell

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”):

use tallerMongoDB1

db.createCollection("productos")

db.productos.insertMany([

  { nombre: "Televisor LED", precio: 750, categoria: "Electrónica" },

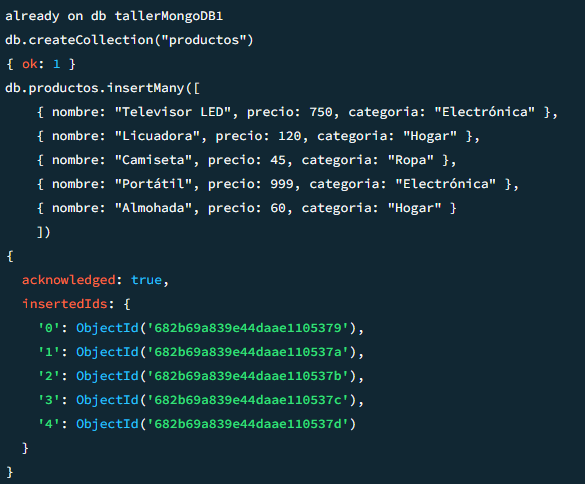
  { nombre: "Licuadora", precio: 120, categoria: "Hogar" },

  { nombre: "Camiseta", precio: 45, categoria: "Ropa" },

{ nombre: "Portátil", precio: 999, categoria: "Electrónica" },

  { nombre: "Almohada", precio: 60, categoria: "Hogar" }

])



2. Realiza las siguientes consultas:

• Encuentra todos los productos con precio mayor a $100: db.productos.find({ precio: { $gt: 100 } })



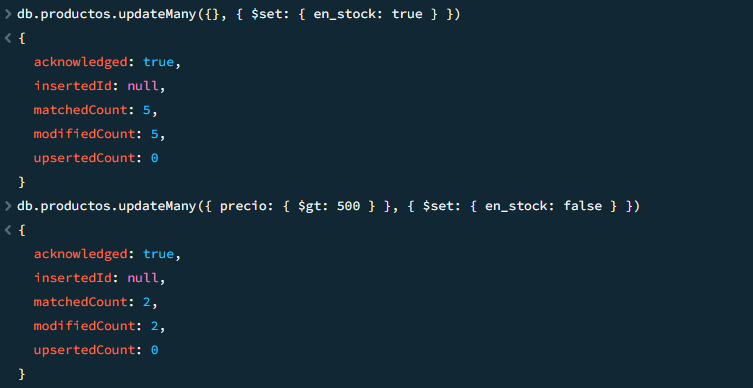
• Ordena los productos por precio de manera descendente: db.productos.find().sort({ precio: -1 })



3. Actualiza los datos:

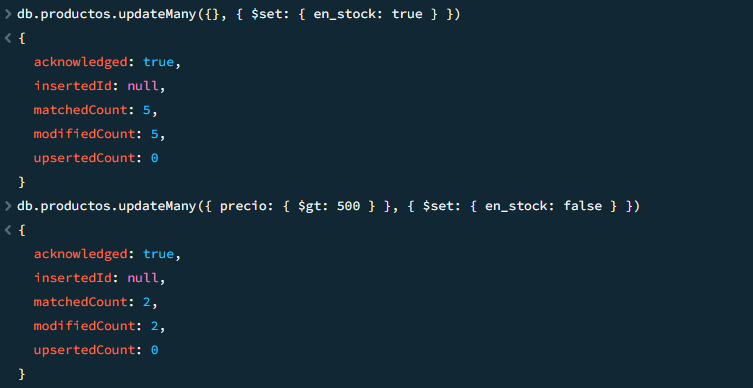
• Añade un campo en stock con valor true a todos los productos: db.productos.updateMany({}, { $set: { en\_stock: true } })

• Cambia el valor de en stock a false para los productos cuyo precio sea mayor a $500: db.productos.updateMany({ precio: { $gt: 500 } }, { $set: { en\_stock: false } })



4. Elimina registros:

• Elimina todos los productos cuyo precio sea menor a $50: db.productos.deleteMany({ precio: { $lt: 50 } })



\*Interfaz

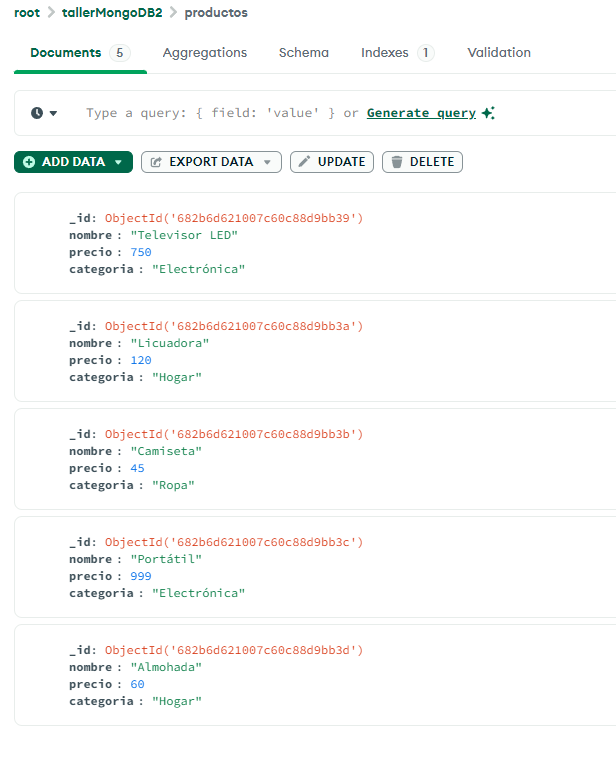
1. Crea una nueva colección llamada productos e inserta al menos 5 productos con los siguientes campos:

• nombre: Nombre del producto.

• precio: Precio del producto.

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

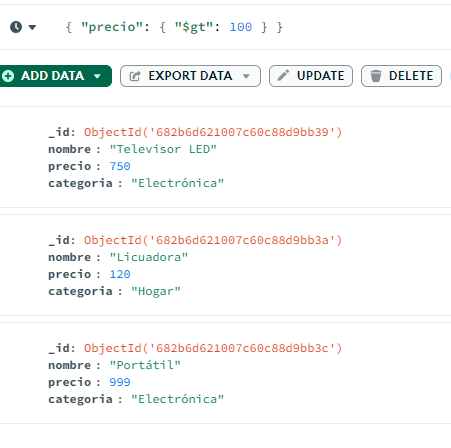
-Agregar la colección "Usuario" e importar los documentos



2. Realiza las siguientes consultas:

• Encuentra todos los productos con precio mayor a $100:

{ "precio": { "$gt": 100 } }



• Ordena los productos por precio de manera descendente:

“Sort”, selecciona precio y “Desc” (-1).

3. Actualiza los datos:

• Añade un campo en stock con valor true a todos los productos:

{ "$set": { "en\_stock": true } }



• Cambia el valor de en stock a false para los productos cuyo precio sea mayor a $500:

-Colocar condicion

{ "precio": { "$gt": 500 } }

{ "$set": { "en\_stock": false } }



4. Elimina registros:

• Elimina todos los productos cuyo precio sea menor a $50: { "precio": { "$lt": 50 } }

\*node.js

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”).

const mongoose=require('mongoose');

const ProductoSchema=new mongoose.Schema({

    nombre:{type:String,require:true},

    precio:{type:Number,require:true},

    categoria:{type:String,require:true},

    creadoEn:{type:Date,default:Date.now}

});

module.exports=mongoose.model('Producto',ProductoSchema)

1. Realiza las siguientes consultas:

• Encuentra todos los productos con precio mayor a $100.

• Ordena los productos por precio de manera descendente.

1. Actualiza los datos:

• Añade un campo en stock con valor true a todos los productos.

• 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

\*Shell

1. Realiza una agregación para calcular:

db.productos.aggregate([

{ $group: {\_id: "$categoria", precioPromedio: { $avg: "$precio" } }}

])



• El precio promedio de los productos agrupados por categoría.

1. Crea una consulta que devuelva:

• La categoría con el mayor precio promedio.

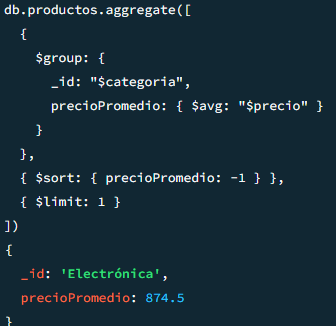
db.productos.aggregate([

{$group: {\_id: "$categoria", precioPromedio: { $avg: "$precio" } } },

{ $sort: { precioPromedio: -1 } },

{ $limit: 1 }

])



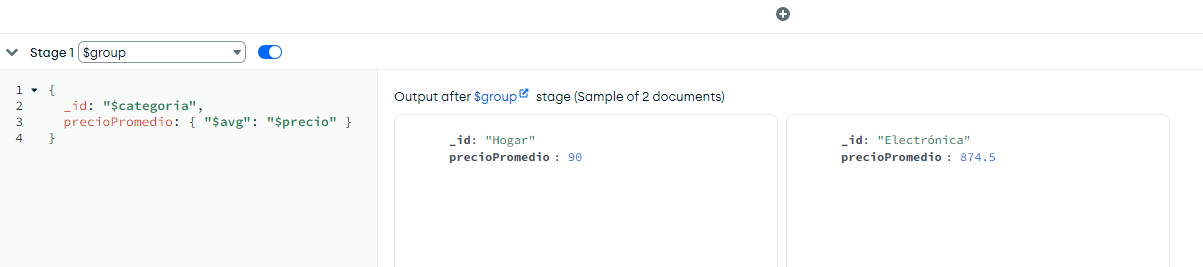
\*Interfaz

1. Realiza una agregación para calcular:

• El precio promedio de los productos agrupados por categoría.

{ \_id: "$categoria",precioPromedio: { "$avg": "$precio" }

}



1. Crea una consulta que devuelva:

• La categoría con el mayor precio promedio.

\*node.js