

TALLER MONGO DB

METODO 1

Crea una base de datos llamada tallerMongoDB.

```
db.createCollection("usuarios")
{ ok: 1 }
```

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

```
db.usuarios.insertMany([{"nombre":"juan perez",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"}])  
{  
  acknowledged: true,  
  insertedIds: {  
    '0': ObjectId('68225bc8ae913d8a2386135c'),  
    '1': ObjectId('68225bc8ae913d8a2386135d'),  
    '2': ObjectId('68225bc8ae913d8a2386135e')  
  }  
}
```

3. • Encuentra todos los usuarios.

```
db.usuarios.find()
{
  _id: ObjectId('68225bc8ae913d8a2386135c'),
  nombre: 'juan perez',
  edad: 30,
  correo: 'juan.perez@urosario.com'
}
{
  _id: ObjectId('68225bc8ae913d8a2386135d'),
  nombre: 'ana lopez',
  edad: 25,
  correo: 'ana.lopez@urosario.com'
}
{
  _id: ObjectId('68225bc8ae913d8a2386135e'),
  nombre: 'Luis torres',
  edad: 35,
  correo: 'luis.torres@urosario.com'
}
```

Encuentra el usuario con nombre Ana López

```
db.usuarios.find({nombre:{$eq:"ana lopez"})  
{  
  _id: ObjectId('68225bc8ae913d8a2386135d'),  
  nombre: 'ana lopez',  
  edad: 25,  
  correo: 'ana.lopez@urosario.com'  
}
```

Encuentra todos los usuarios mayores o iguales a 30 años

```
db.usuarios.find({edad:{$gte:30}})  
{  
  _id: ObjectId('68226eff1fbda6fb5b9cbb79'),  
  nombre: 'juan perez',  
  edad: 30,  
  correo: 'juan.perez@urosario.com'  
}  
{  
  _id: ObjectId('68226eff1fbda6fb5b9cbb7b'),  
  nombre: 'Luis torres',  
  edad: 35,  
  correo: 'luis.torres@urosario.com'  
}
```

4. Actualiza los datos

Cambia la edad de Juan Pérez a 31 años.

```
db.usuarios.updateOne({nombre:{$eq:"juan perez"}},{$set:{edad:31}})  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

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

```
db.usuarios.updateMany({edad:{$gt:30}},{$set:{activo:"true"}})  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 2,  
  modifiedCount: 2,  
  upsertedCount: 0  
}
```

5. Elimina registros: •

Elimina el usuario Luis Torres.

```
db.usuarios.deleteMany({nombre:{$eq:"Luis torres"}})  
{  
  acknowledged: true,  
  deletedCount: 1  
}
```

Elimina todos los usuarios menores de 30 años.

```
db.usuarios.deleteMany({edad:{$lt:30}})  
{  
  acknowledged: true,  
  deletedCount: 1  
}
```

Parte 2: Gestión de Productos

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

```
db.productos.insertMany([{"nombre":"camisa oversized",precio:20,categoria:"ropa"},  
{nombre:"lavadora",precio:700,categoria:"hogar"},  
{nombre:"cafetera",precio:30,categoria:"electronica"}])  
{  
    acknowledged: true,  
    insertedIds: {  
        '0': ObjectId('68226046ae913d8a2386135f'),  
        '1': ObjectId('68226046ae913d8a23861360'),  
        '2': ObjectId('68226046ae913d8a23861361')  
    }  
}
```

```
db.productos.insertMany([{"nombre":"ps5",precio:600,categoria:"electronica"},  
{nombre:"portatil",precio:700,categoria:"electronica"}])  
{  
    acknowledged: true,  
    insertedIds: {  
        '0': ObjectId('68226af1ae913d8a23861367'),  
        '1': ObjectId('68226af1ae913d8a23861368')  
    }  
}
```

. Realiza las siguientes consultas: •

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

```
db.productos.find({precio:{$gt:100}})  
{  
  _id: ObjectId('68226aec913d8a23861365'),  
  nombre: 'lavadora',  
  precio: 700,  
  categoria: 'hogar'  
}  
{  
  _id: ObjectId('68226af1ae913d8a23861367'),  
  nombre: 'ps5',  
  precio: 600,  
  categoria: 'electronica'  
}  
{  
  _id: ObjectId('68226af1ae913d8a23861368'),  
  nombre: 'portatil',  
  precio: 700,  
  categoria: 'electronica'  
}
```

Ordena los productos por precio de manera descendente

```
db.productos.find().sort({precio:-1})  
{  
  _id: ObjectId('68226aec913d8a23861365'),  
  nombre: 'lavadora',  
  precio: 700,  
  categoria: 'hogar'  
}  
{  
  _id: ObjectId('68226af1ae913d8a23861368'),  
  nombre: 'portatil',  
  precio: 700,  
  categoria: 'electronica'  
}  
{  
  _id: ObjectId('68226af1ae913d8a23861367'),  
  nombre: 'ps5',  
  precio: 600,  
  categoria: 'electronica'  
}  
{  
  _id: ObjectId('68226aec913d8a23861366'),  
  nombre: 'cafetera',  
  precio: 30,  
  categoria: 'electronica'  
}  
{  
  _id: ObjectId('68226aec913d8a23861364'),  
  nombre: 'camisa oversized',  
  precio: 20,  
  categoria: 'ropa'
```

Actualiza los datos:

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

```
db.productos.updateMany({}, {$set: {"en stock": "true"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
```

- 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"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

Elimina registros:

- Elimina todos los productos cuyo precio sea menor a \$50

```
db.productos.deleteMany({precio: {$lt: 50}})
{
  acknowledged: true,
  deletedCount: 2
}
```

Parte 3: Agregaciones y Análisis 1. Realiza una agregación para calcular:

- El precio promedio de los productos agrupados por categoría.

```
db.productos.aggregate([{$group: {_id: "$categoria", precioPromedio: { $avg: "$precio" }}}])
{
  _id: 'hogar',
  precioPromedio: 700
}
{
  _id: 'electronica',
  precioPromedio: 650
}
```

// para este punto busque documentación:

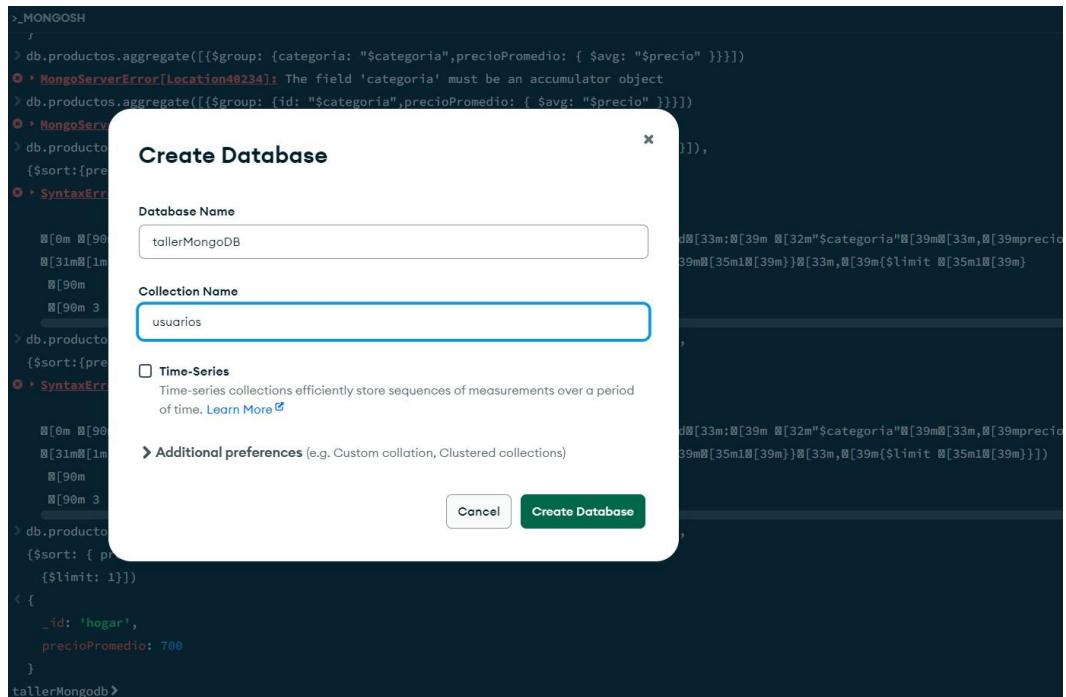
<https://www.mongodb.com/docs/manual/aggregation/>

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}])  
{  
  _id: 'hogar',  
  precioPromedio: 700  
}
```

METOODO 2

Crea una base de datos llamada tallerMongoDB. 2. Inserta la siguiente información en una colección llamada usuarios:



Insert Document

To collection tallerMongoDB.usuarios

VIEW { } ⋮

```
1 ▾ [  
2 ▾ {  
3   "nombre": "juan perez", "edad": 30, "correo": "juan.perez@urosario.com"  
4 },  
5 ▾ {  
6   "nombre": "ana lopez", "edad": 25, "correo": "ana.lopez@urosario.com"  
7 },  
8 ▾ [  
9   {  
10    "nombre": "Luis torres", "edad": 35, "correo": "luis.torres@urosario.com"  
11  }]  
12 ]
```

Cancel

Insert



```
_id: ObjectId('68226eff1fbda6fb5b9cbb79')  
nombre : "juan perez"  
edad : 30  
correo : "juan.perez@urosario.com"
```



```
_id: ObjectId('68226eff1fbda6fb5b9cbb7a')  
nombre : "ana lopez"  
edad : 25  
correo : "ana.lopez@urosario.com"
```



```
_id: ObjectId('68226eff1fbda6fb5b9cbb7b')  
nombre : "Luis torres"  
edad : 35  
correo : "luis.torres@urosario.com"
```

3. Realiza las siguientes consultas:

- Encuentra todos los usuarios

The screenshot shows a MongoDB query interface with the following details:

- Header:** Includes "Generate query", "Explain", "Reset", and "Find" buttons.
- Action Buttons:** ADD DATA, EXPORT DATA, UPDATE, DELETE.
- Pagination:** 25, 1-3 of 3, navigation arrows.
- Document 1:**

```
_id: ObjectId('68226eff1fbda6fb5b9cbb79')
nombre : "juan perez"
edad : 30
correo : "juan.perez@urosario.com"
```
- Document 2:**

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7a')
nombre : "ana lopez"
edad : 25
correo : "ana.lopez@urosario.com"
```
- Document 3:**

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7b')
nombre : "Luis torres"
edad : 35
correo : "luis.torres@urosario.com"
```

Encuentra el usuario con nombre Ana López.

The screenshot shows a MongoDB query interface with the following details:

- Query Bar:** {"nombre": {\$eq:"ana lopez"}}, highlighted with a blue border.
- Header:** Includes "Generate query", "Explain", "Reset", and "Find" buttons.
- Action Buttons:** ADD DATA, EXPORT DATA, UPDATE, DELETE, INSIGHT.
- Pagination:** 25, 1-1 of 1, navigation arrows.
- Document:**

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7a')
nombre : "ana lopez"
edad : 25
correo : "ana.lopez@urosario.com"
```

Encuentra todos los usuarios mayores o iguales a 30 años.

⌚ ▾ {"edad":{\$gte:30}}

[Generate query](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#) [INSIGHT](#) 25

```
_id: ObjectId('68226eff1fbda6fb5b9cbb79')
nombre : "juan perez"
edad : 30
correo : "juan.perez@urosario.com"

_id: ObjectId('68226eff1fbda6fb5b9cbb7b')
nombre : "Luis torres"
edad : 35
correo : "luis.torres@urosario.com"
```

Cambia la edad de Juan Pérez a 31 años

Update 1 document

tallerMongoDB.usuarios

Filter ⓘ { nombre: { \$eq: 'juan perez' } }

Update

Learn more about Update syntax ⓘ

```
1 ▾ {
2 ▾   $set: {
3   edad:31
4   },
5 }
```

★ Save Cancel [Update 1 document](#)

```
_id: ObjectId('68226eff1fbda6fb5b9cbb79')
nombre : "juan perez"
edad : 31
correo : "juan.perez@urosario.com"
```

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

Update 2 documents

tallerMongoDB.usuarios

Filter ⓘ

```
{ edad: { $gte: 30 } }
```

Update

Learn more about Update syntax ⓘ

```
1 ▼ {  
2 ▼   $set: {  
3   activo:"true"  
4   },  
5 }
```

Save Cancel **Update 2 documents**

```
_id: ObjectId('68226eff1fbda6fb5b9cbb79')  
nombre : "juan perez"  
edad : 31  
correo : "juan.perez@urosario.com"  
activo : "true"
```

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7b')  
nombre : "Luis torres"  
edad : 35  
correo : "luis.torres@urosario.com"  
activo : "true"
```

- . Elimina registros: • Elimina el usuario Luis Torres.

⚠ Delete 1 document

tallerMongoDB.usuarios

Filter **i** { nombre: { \$eq: 'Luis torres' } }

Export

Preview (sample of 1 document)

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7b')
nombre : "Luis torres"
edad : 35
correo : "luis.torres@urosario.com"
activo : "true"
```

Cancel **Delete 1 document**

- Elimina todos los usuarios menores de 30 años.

⚠ Delete 1 document

tallerMongoDB.usuarios

Filter **i** { edad: { \$lt: 30 } }

Export

Preview (sample of 1 document)

```
_id: ObjectId('68226eff1fbda6fb5b9cbb7a')
nombre : "ana lopez"
edad : 25
correo : "ana.lopez@urosario.com"
```

Cancel **Delete 1 document**

Parte 2: Gestión de Productos 1. Crea una nueva colección llamada productos e inserta al menos 5 productos con los siguientes campos:

Create Collection

Collection Name

productos

Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

› Additional preferences (e.g. Custom collation, Clustered collections)

Cancel

Create Collection

VIEW

```
1 ▾ [  
2   {"nombre":"camisa oversize", "precio":20, "categoria":"ropa"},  
3   {"nombre":"lavadora", "precio":700, "categoria":"hogar"},  
4   {"nombre":"cafetera", "precio":30, "categoria":"electronica"},  
5   {"nombre":"ps5", "precio":600, "categoria":"electronica"},  
6   {"nombre":"portatil", "precio":700, "categoria":"electronica"}  
7 ]
```

Cancel

Insert

```

_id: ObjectId('6822738a1fbda6fb5b9cbb80')
nombre : "camisa oversized"
precio : 20
categoria : "ropa"

_id: ObjectId('6822738a1fbda6fb5b9cbb81')
nombre : "lavadora"
precio : 700
categoria : "hogar"

_id: ObjectId('6822738a1fbda6fb5b9cbb82')
nombre : "cafetera"
precio : 30
categoria : "electronica"

_id: ObjectId('6822738a1fbda6fb5b9cbb83')
nombre : "ps5"
precio : 600
categoria : "electronica"

_id: ObjectId('6822738a1fbda6fb5b9cbb84')
x : "portatil"
: 700
ria : "electronica"

```

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

The screenshot shows a MongoDB query interface. At the top, there is a search bar containing the query `{precio: {$gt: 30}}`. Below the search bar are several buttons: ADD DATA, EXPORT DATA, UPDATE, DELETE, and INSIGHT. The main area displays three document results, each with an _id, nombre, precio, and categoria field. The first document is a lavadora at 700. The second is a ps5 at 600. The third is a portatil at 700.

```

_id: ObjectId('6822738a1fbda6fb5b9cbb81')
nombre : "lavadora"
precio : 700
categoria : "hogar"

_id: ObjectId('6822738a1fbda6fb5b9cbb83')
nombre : "ps5"
precio : 600
categoria : "electronica"

_id: ObjectId('6822738a1fbda6fb5b9cbb84')
x : "portatil"
: 700
ria : "electronica"

```

- Ordena los productos por precio de manera descendente

Type a query: { field: 'value' } or [Generate query](#)

Project: {}

Sort: {precio:-1}

Max Time MS: 60000

Collation: { locale: 'simple' }

Skip: 0

Limit: 0

Index Hint: { field: -1 }

ADD DATA **EXPORT DATA** **UPDATE** **DELETE** **INSIGHT**

25 1 - 5 of 5

```

_id: ObjectId('6822738a1fbda6fb5b9cbb81')
nombre: "lavadora"
precio: 700
categoria: "hogar"

_id: ObjectId('6822738a1fbda6fb5b9cbb84')
nombre: "portatil"
precio: 700
categoria: "electronica"

_id: ObjectId('6822738a1fbda6fb5b9cbb83')
nombre: "ps5"
precio: 600
categoria: "electronica"

```

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

```

1 ▾ {
2 ▾   $set: {
3     "en stock": "true"
4   },
5 }

```

<pre>_id: ObjectId('6822738a1fbda6fb5b9cbb80') nombre : "camisa oversized" precio : 20 categoria : "ropa" en stock : "true"</pre>	<input type="button" value="Edit"/>
<pre>_id: ObjectId('6822738a1fbda6fb5b9cbb81') nombre : "lavadora" precio : 700 categoria : "hogar" en stock : "true"</pre>	
<pre>_id: ObjectId('6822738a1fbda6fb5b9cbb82') nombre : "cafetera" precio : 30 categoria : "electronica" en stock : "true"</pre>	
<pre>_id: ObjectId('6822738a1fbda6fb5b9cbb83') nombre : "ps5" precio : 600 categoria : "electronica" en stock : "true"</pre>	

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

Update 3 documents

tallerMongoDB.productos

Filter

{ precio: { \$gt: 500 } }

Update

Learn more about Update syntax 

```
1 ▼ {
2 ▼   $set: {
3     "en stock": "false"
4   },
5 }
```

⌚ {precio:{\$gt:500}}

Get

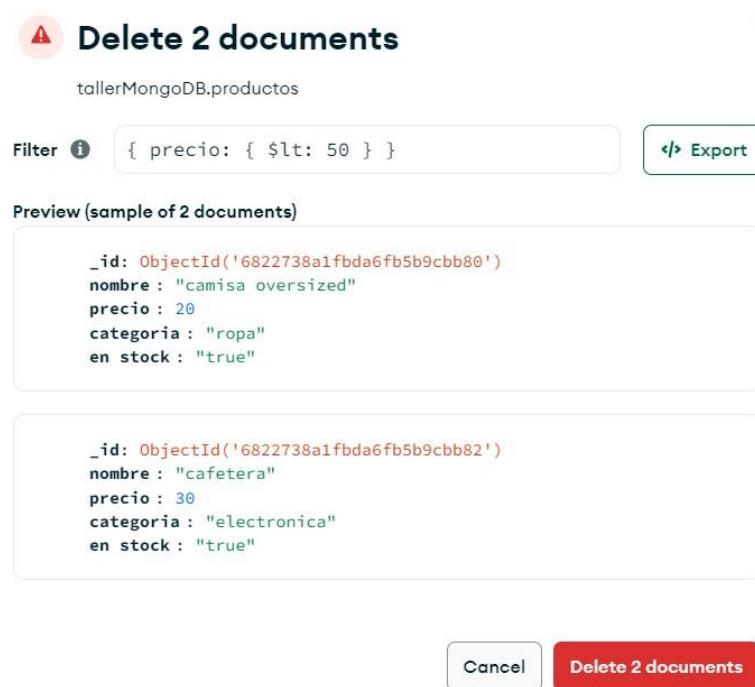
ADD DATA **EXPORT DATA** **UPDATE** **DELETE** **INSIGHT**

```
_id: ObjectId('6822738a1fbda6fb5b9cbb81')
nombre : "lavadora"
precio : 700
categoria : "hogar"
en stock : "false"

_id: ObjectId('6822738a1fbda6fb5b9cbb83')
nombre : "ps5"
precio : 600
categoria : "electronica"
en stock : "false"

_id: ObjectId('6822738a1fbda6fb5b9cbb84')
nombre : "portatil"
precio : 700
categoria : "electronica"
en stock : "false"
```

- Elimina todos los productos cuyo precio sea menor a \$50.



Parte 3: Agregaciones y Análisis 1. Realiza una agregación para calcular:

- El precio promedio de los productos agrupados por categoría.

Stage 1 \$group

```

1  /**
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5  {
6    _id: "$categoria",
7    precioPromedio: {
8      $avg: "$precio"
9    }
10 }

```

Output after \$group stage (Sample of 2 documents)

`_id: "hogar"
precioPromedio : 700`

`_id: "electronica"
precioPromedio : 650`

La categoría con el mayor precio promedio

//me costo bastante, lo que hice fue usar la parte de compass “aggregations” y dividi la consulta de mayor promedio en 3, primero promedios luego orden luego limite

Stage 2 \$sort

```

1 /**
2  * Provide the number of documents to limit
3 */
4 {precio:-1}

```

Output after \$sort stage (Sample of 2 documents)

`_id: "electronica"
precioPromedio : 650`

`_id: "hogar"
precioPromedio : 700`

Stage 3 \$limit

```

1 /**
2  * Provide the number of documents to limit
3 */
4 1

```

Output after \$limit stage (Sample of 1 document)

`_id: "electronica"
precioPromedio : 650`

METODO 3

Crea una base de datos llamada tallerMongoDB.

```

mongoose.connect('mongodb://localhost:27017/tallerMongoDB',{
  useNewUrlParser:true,
  useUnifiedTopology:true,
})
.then(()=>console.log("MongoDB conectado"))
.catch(err=>console.error(err))

```

tallerMongoDB

Inserta la siguiente información en una colección llamada usuarios:

- Juan Pérez, 30 años, correo: juan.perez@urosario.com
- Ana López, 25 años, correo: ana.lopez@urosario.com
- Luis Torres, 35 años, correo: luis.torres@urosario.com

```
const UsuariosSchema=new mongoose.Schema({
  nombre:{type:String,require:true},
  edad:Number,
  correo:String,
  creadoEn:{type:Date,default:Date.now}
});
```

The screenshot shows a Postman interface with a POST request to `http://localhost:3000/api/Usuarios`. The request body contains the following JSON data:

```
1 | [
2 |   {
3 |     "nombre": "Juan Pérez",
4 |     "edad": 30,
5 |     "correo": "juan.perez@urosario.com"
6 |   }
]
```

The response status is `201 Created`, time `8 ms`, size `421 B`.

Realizamos lo mismo con los demás documentos, el resultado esta a continuación:

```
_id: ObjectId('682a7dc6ca1a318d7b4a9ac8')
nombre : "Juan Pérez"
edad : 30
correo : "juan.perez@urosario.com"
creadoEn : 2025-05-19T00:39:34.161+00:00
__v : 0



---

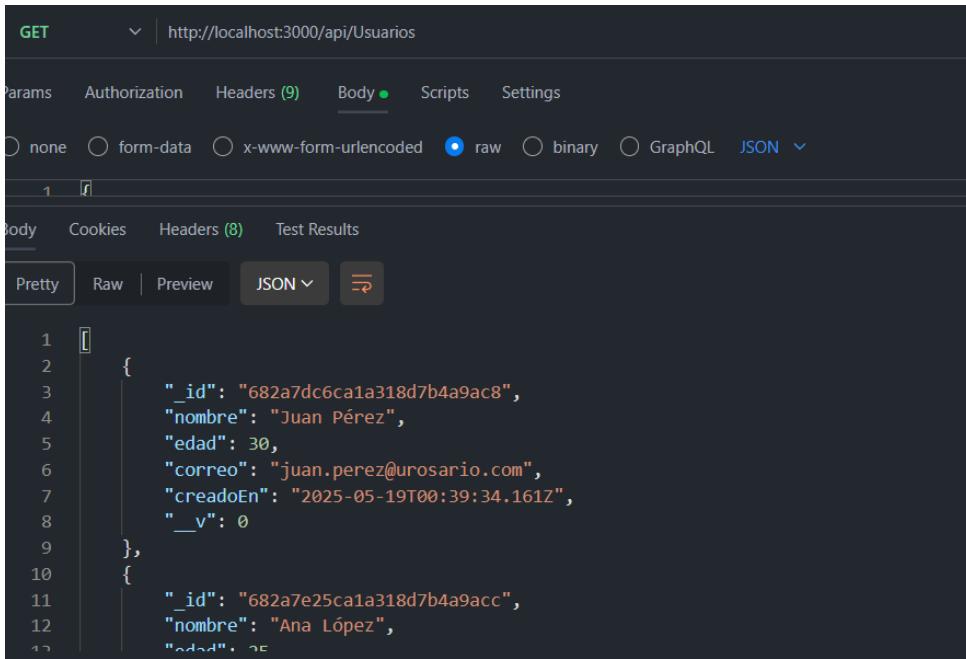

_id: ObjectId('682a7e25ca1a318d7b4a9acc')
nombre : "Ana López"
edad : 25
correo : "ana.lopez@urosario.com"
creadoEn : 2025-05-19T00:41:09.484+00:00
__v : 0



---


_id: ObjectId('682a7e32ca1a318d7b4a9ace')
nombre : "Luis Torres"
edad : 35
correo : "luis.torres@urosario.com"
creadoEn : 2025-05-19T00:41:22.209+00:00
__v : 0
```

Consultar todos los usuarios:



The screenshot shows a Postman interface with a GET request to `http://localhost:3000/api/Usuarios`. The response body is displayed in JSON format, showing two users:

```
1 [  
2   {  
3     "_id": "682a7dc6ca1a318d7b4a9ac8",  
4     "nombre": "Juan Pérez",  
5     "edad": 30,  
6     "correo": "juan.perez@urosario.com",  
7     "creadoEn": "2025-05-19T00:39:34.161Z",  
8     "__v": 0  
9   },  
10  {  
11    "_id": "682a7e25ca1a318d7b4a9acc",  
12    "nombre": "Ana López",  
13    "edad": 25  
14  }]
```

Consultar los usuarios con nombre ana Lopez

Lo que se hace es extender el get, y ponerle un filtro usando el operador “\$eq”

```
router.get('/', async (req, res) => {  
  try {  
    const { nombre, edad } = req.query;  
  
    let filtro = {};  
    if (nombre) filtro.nombre = { $eq: nombre };  
    const clientes = await Cliente.find(filtro);  
    res.json(clientes);  
  } catch (error) {  
    res.status(500).json({ error: error.message });  
  }  
});
```

GET | http://localhost:3000/api/Usuarios?nombre=Ana López

Params • Authorization Headers (9) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 [f]

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

```
1 [
2   {
3     "_id": "682a7e25ca1a318d7b4a9acc",
4     "nombre": "Ana López",
5     "edad": 25,
6     "correo": "ana.lopez@urosario.com",
7     "creadoEn": "2025-05-19T00:41:09.484Z",
8     "__v": 0
9   }
10 ]
```

Usuarios con edad mayor o igual a 30

Similar a con nombre, simplemente agregamos un parámetro mas de búsqueda con “\$gte”

```
router.get('/', async (req, res) => {
  try {
    const { nombre, edad } = req.query;

    let filtro = {};
    if (nombre) filtro.nombre = { $eq: nombre };
    if (edad) filtro.edad = { $gte: edad };
    const clientes = await Cliente.find(filtro);
    res.json(clientes);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

Cambia la edad de Juan Pérez a 31 años

```
router.put('/:id', async(req,res)=>{
  try{
    const cliente=await Cliente.findByIdAndUpdate(req.params.id, req.body,{new:true});
    if (!cliente) return res.status(404).json({error: 'Producto no encontrado'});
    res.json(cliente);

  }catch(error){
    res.status(500).json({ error: error.message});
  }
})
```

The screenshot shows a POSTMAN interface with a PUT request to `http://localhost:3000/api/Usuarios/682a7dc6ca1a318d7b4a9ac8`. The Body tab is selected, showing raw JSON data:

```
1 {
2   "nombre": "Juan Pérez",
3   "edad": 31,
4   "correo": "juan.perez@urosario.com"
5 }
```

The response body shows the updated user data:

```
1 {
2   "_id": "682a7dc6ca1a318d7b4a9ac8",
3   "nombre": "Juan Pérez",
4   "edad": 31,
```

Añade el campo activo: true a todos los usuarios con edad mayor o igual a 30 años.
Creamos una nueva ruta en usuariosRoutes, para hacer el update a todos los usuarios

```

router.put('/activar-mayores-30', async (req, res) => {
  try {
    const resultado = await Cliente.updateMany(
      { edad: { $gte: 30 } }, // filtro: edad >= 30
      { $set: { activo: true } } // actualización: agrega campo activo:true
    );

    res.json({
      mensaje: 'Usuarios actualizados',
      modificados: resultado.modifiedCount
    });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

```

PUT http://localhost:3000/api/Usuarios/activar-mayores-30

Params Authorization Headers (9) Body Scripts Settings

(none) form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {}

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

1 []
2 "mensaje": "Usuarios actualizados"
3]

Eliminar a Luis torres:

```

router.delete('/:id',async(req,res)=>{
  try{
    const cliente=await Cliente.findByIdAndDelete(req.params.id);
    if (!cliente) return res.status(404).json({error: 'Producto no encontrado'});
    res.json(cliente);

  }catch(error){
    res.status(500).json({ error: error.message})
  }
};

module.exports=router;

```

DELETE http://localhost:3000/api/Usuarios/682a7e32ca1a318d7b4a9ace

Params Authorization Headers (7) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

```

1 [
2   "_id": "682a7e32ca1a318d7b4a9ace",
3   "nombre": "Luis Torres",
4   "edad": 35,

```

Eliminar todos los usuarios menores de 30

```
router.delete('/menores-30', async (req, res) => {
  try {
    const resultado = await Cliente.deleteMany({ edad: { $lt: 30 } });
    res.json({ mensaje: `Se eliminaron ${resultado.deletedCount} usuarios menores de 30 años.` });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

http://localhost:3000/api/Usuarios/menores-30

DELETE http://localhost:3000/api/Usuarios/menores-30

Save | No Environment

Params Authorization Headers (7) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

```

1 [
2   "mensaje": "Se eliminaron 1 usuarios menores de 30 años."
3 ]

```

Status: 200 OK Time: 21 ms Size: 326 B

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

```

const mongoose=...require('mongoose');

const itemSchema=new mongoose.Schema({
    nombreDelProducto:{type:String,require:true},
    Precio:Number,
    categoria:String,
    creadoEn:{type:Date,default:Date.now}
});

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

```

The screenshot shows a browser's developer tools Network tab. A POST request to the URL `/api/usuarios` is selected. The request body is displayed in JSON format:

```
{
  "nombredelProducto": "Mouse 6 botnes",
  "Precio": 40,
  "categoria": "Tecnología"
}
```

Below the request, there are tabs for Cookies, Headers (8), and Test Results. Under Headers, there are Raw, Preview, and JSON dropdowns. The JSON dropdown is currently active, showing the same JSON object as the request body.

Es exactamente igual a los post de usuarios, simplemente cambiamos los campos y la URL

```
_id: ObjectId('682a8b139414d2b46508368c')
nombredelProducto : "Mouse 6 botnes"
Precio : 40
categoria : "Tecnología"
creadoEn : 2025-05-19T01:36:19.906+00:00
__v : 0
```

```
-----
```

```
_id: ObjectId('682a8be49414d2b46508368e')
nombredelProducto : "Camisa jugador barcelona"
Precio : 1000
categoria : "Ropa"
creadoEn : 2025-05-19T01:39:48.181+00:00
__v : 0
```

```
-----
```

```
_id: ObjectId('682a8c069414d2b465083690')
nombredelProducto : "Lavadora"
Precio : 700
categoria : "hogar"
creadoEn : 2025-05-19T01:40:22.071+00:00
__v : 0
```

```
-----
```

```
_id: ObjectId('682a8c129414d2b465083692')
nombredelProducto : "cafetera"
Precio : 30
categoria : "hogar"
creadoEn : 2025-05-19T01:40:34.832+00:00
```

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

```
router.get('/', async (req, res) => {
  try {
    const { precio } = req.query;
    let filtro = {};

    if (precio) {
      filtro.Precio = { $gte: Number(precio) };
    }

    const productos = await Producto.find(filtro);
    res.json(productos);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

GET http://localhost:3000/api/Producto/?Precio=100

Params • Authorization Headers (7) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

```
1 [ { _id: "682a8b139414d2b46508368c", "nombredelProducto": "Mouse 6 botnes", "Precio": 40, "categoria": "Tecnología", } ]
```

Ordena los productos por precio de manera descendente.

```
router.get('/', async (req, res) => {
  try {
    const { precio } = req.query;

    let filtro = {};
    if (precio) {
      filtro.Precio = { $gte: Number(precio) };
    }

    const productos = await Producto.find(filtro).sort({ Precio: -1 });
    res.json(productos);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

Como en mongo por consola, realizamos el sort por precio, el -1 indica orden desc

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

```
router.put('/actualizar-stock', async (req, res) => {
  try {
    const resultado = await Producto.updateMany({}, { $set: { stock: true } });
    res.json({ mensaje: 'Stock actualizado para todos los productos', resultado });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

No me sirvió, me daba "acknowledged": false, entonces lo hice por interfaz para seguir con los demás puntos 😞

The screenshot shows the Postman interface with a PUT request to `http://localhost:3000/api/Producto/actualizar-stock`. The Headers tab shows 8 items. The Body tab is selected, showing the raw JSON response:

```
1 {
  2   "mensaje": "Stock actualizado para todos los productos",
  3   "resultado": {
  4     "acknowledged": false
  5   }
  6 }
```

4. Elimina registros: • Elimina todos los productos cuyo precio sea menor a \$50.

```
router.delete('/eliminar-precio-menor-50', async (req, res) => {
  try {
    const resultado = await Producto.deleteMany({ Precio: { $lt: 50 } });
    res.json({ mensaje: 'Productos eliminados con precio menor a 50', resultado });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

DELETE

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 [{}]

Body Cookies Headers (8) Test Results

Pretty Raw | Preview JSON ↻

```
1 [
2   "mensaje": "Productos eliminados con precio menor a 50",
3   "resultado": {
4     "acknowledged": true,
5     "deletedCount": 2
6   }
7 ]
```