



ACTIVIDAD DE PROYECTO

Construir La Base de Datos



www.sena.edu.co

GUARDAR DATOS

Para guardar datos usamos las siguientes instrucciones:

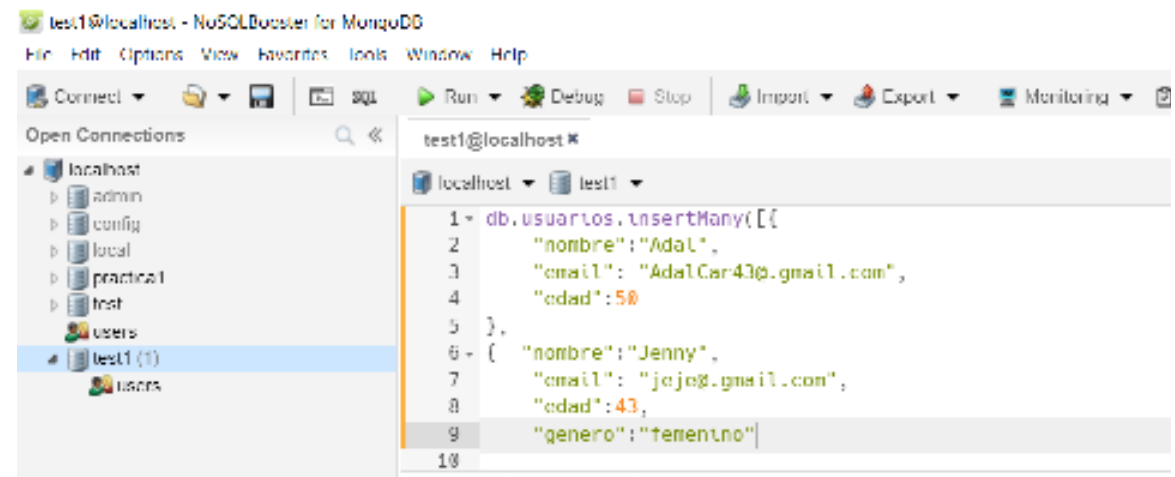
`db.usuarios.insertOne({})` “Para un solo dato”

`db.usuarios.insertMany([{}])` “Para varios datos”

En Mongo Compass , en el Shell lo trabajaríamos así:

```
>_MONGOOSH
> db.publicaciones.insertOne({"titulo":"curso de Python","duracion":200})
< {
  acknowledged: true,
  insertedId: ObjectId("64bd9e591f0a045b05a7b1cb")
}
test> |
```

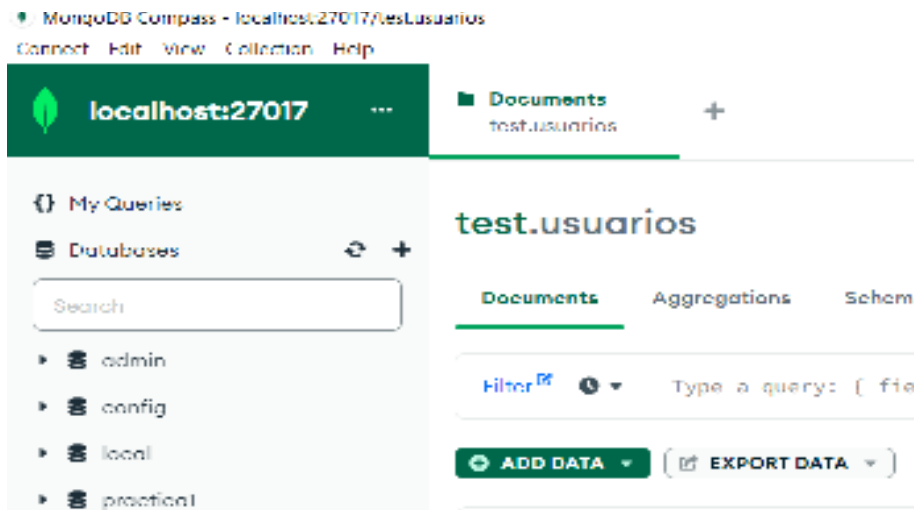
En NoSQLBooster así:



```
test1@localhost - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect Open Connections SQL Run Debug Stop Import Export Monitoring
localhost test1@localhost *
localhost test1
1 db.usuarios.insertMany([
2   {
3     "nombre": "Adal",
4     "email": "AdalCar43@gmail.com",
5     "edad": 50
6   },
7   { "nombre": "Jenny",
8     "email": "jojeg@gmail.com",
9     "edad": 43,
10    "genero": "femenino"
11  }
12 ])
```

ObjectId y LISTAR Documentos

En add data, podemos generar de manera automática un id, de tipo objectId propio de la base de datos noSQL. Y se crea de forma dinámica de tal manera que si quieres volver adicionar un dato se generará otro id diferente.



Para mirar una colección en una base de datos usamos `Show collections` y para listar el documento usamos la instrucción `db.nombre_de_la_colección.find()`

```
>_MONGOSH
> use test1
< switched to db test1
> show collections
< usuarios
> db.usuarios.find()
< {
  _id: ObjectId("64bd96deac1490ff49fd01eb"),
  nombre: 'Carlos',
  email: 'Car43@gmail.com',
  edad: 26
}
```



LISTAR UN DOCUMENTO CON PARÁMETRO

```
>_MONGOSH  
  
> show collections  
< usuarios  
> db.usuarios.find({"nombre":"Adal"})  
< {  
  _id: ObjectId("64bd97caac1490ff49fd01ec"),  
  nombre: 'Adal',  
  email: 'AdalCar43@gmail.com',  
  edad: 50  
}  
test1> |
```

BUSQUEDA AVANZADA

```
db.usuarios.find({"edad":{$gte:40}})
```

Esta instrucción es una búsqueda avanzada utilizando un operador de comparación \$gte significa mayor o igual.

Esto se usa en caso de realizar una consulta con un dato que desconozco.

Los operadores relacionales en nosql son:

\$eq - equal - igual que.

\$ne - not equal – distinto de,

\$lt - low than - menor que

\$lte - low than equal - menor o igual que

\$gt - greater than - mayor que

\$gte - greater than equal - mayor o igual que

\$in - dentro de (Valores que se cuentren en un array de elementos)

\$nin - not in - no dentro de



BUSQUEDA AVANZADA

Podemos filtrar campos si existen

```
db.usuarios.find({"edad":{"$exists:true}})
```

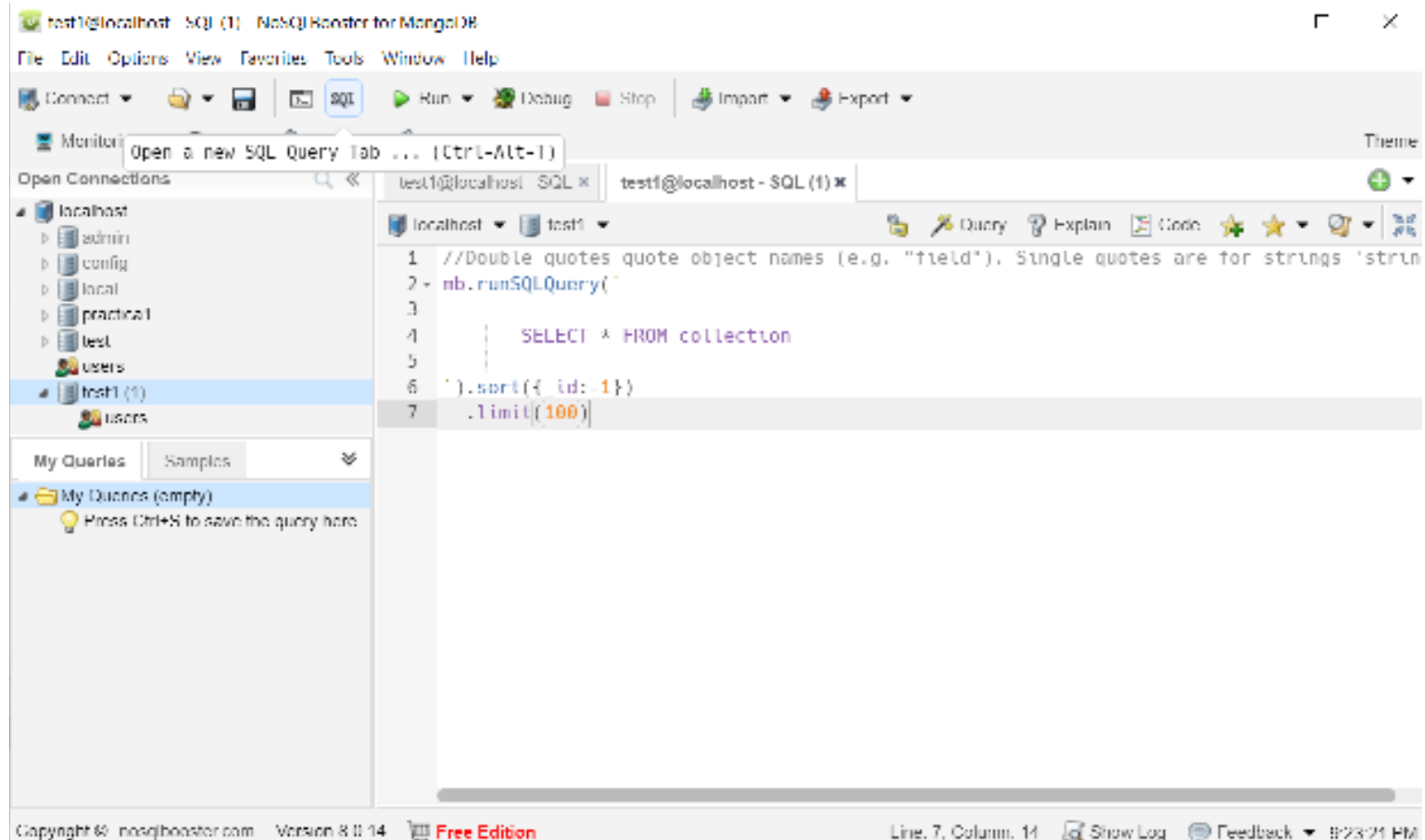
Podemos usar operadores lógicos
cómo or, and.. ejemplo:

```
db.usuarios.find({$and:[{edad:25},{nombre:"Susana"}]})
```

```
db.usuarios.find({$or:[{edad:27},{nombre:"Susana"}]})
```

SQL en NoSQL

Para pasar unas sentencias sql a lenguaje mql podemos hacerlo usando la herramienta NoSQLBooster



Clasula orden by



The screenshot shows the NoSQLBooster for MongoDB application interface. The window title is "test1@localhost - SQL - NoSQLBooster for MongoDB". The menu bar includes File, Edit, Options, View, Favorites, Tools, Window, and Help. The toolbar contains icons for Connect, Open, SQL, Run, Debug, Stop, Import, and Export. Below the toolbar are tabs for Monitoring, Tasks, DataGen, and Schema. The "Open Connections" panel on the left shows a tree view of the database structure, with "test1 (1)" selected. The main editor area displays a SQL query: `SELECT * FROM usuarios order by nombre`. Below the SQL query, the "Query Code" tab shows the equivalent MongoDB Shell code: `use test1; db.usuarios.aggregate([{$sort: {nombre: 1}}]);`. The status bar at the bottom indicates the language is "MongoDB Shell", provides a "Copy to Clipboard" button, and shows the version "8.0.14" and "Free Edition".

test1@localhost - SQL - NoSQLBooster for MongoDB

File Edit Options View Favorites Tools Window Help

Connect Open SQL Run Debug Stop Import Export

Monitoring Tasks DataGen Schema Theme

Open Connections

- localhost
 - admin
 - config
 - local
 - practica1
 - test
 - users
 - test1 (1)
 - users

My Queries Samples

My Queries (empty)

Press Ctrl+S to save the query here

test1@localhost - SQL x test1@localhost - SQL (1) x test1@localhost x

localhost test1

Query Explain Code

```
1 //Double quotes quote object names (e.g. "field"). Single quotes are for strings 'string'
2 mb.runSQLQuery(`
3
4 SELECT * FROM usuarios order by nombre
5
6 `)
```

SQL (Aggregate) x Query Code x

Language: MongoDB Shell

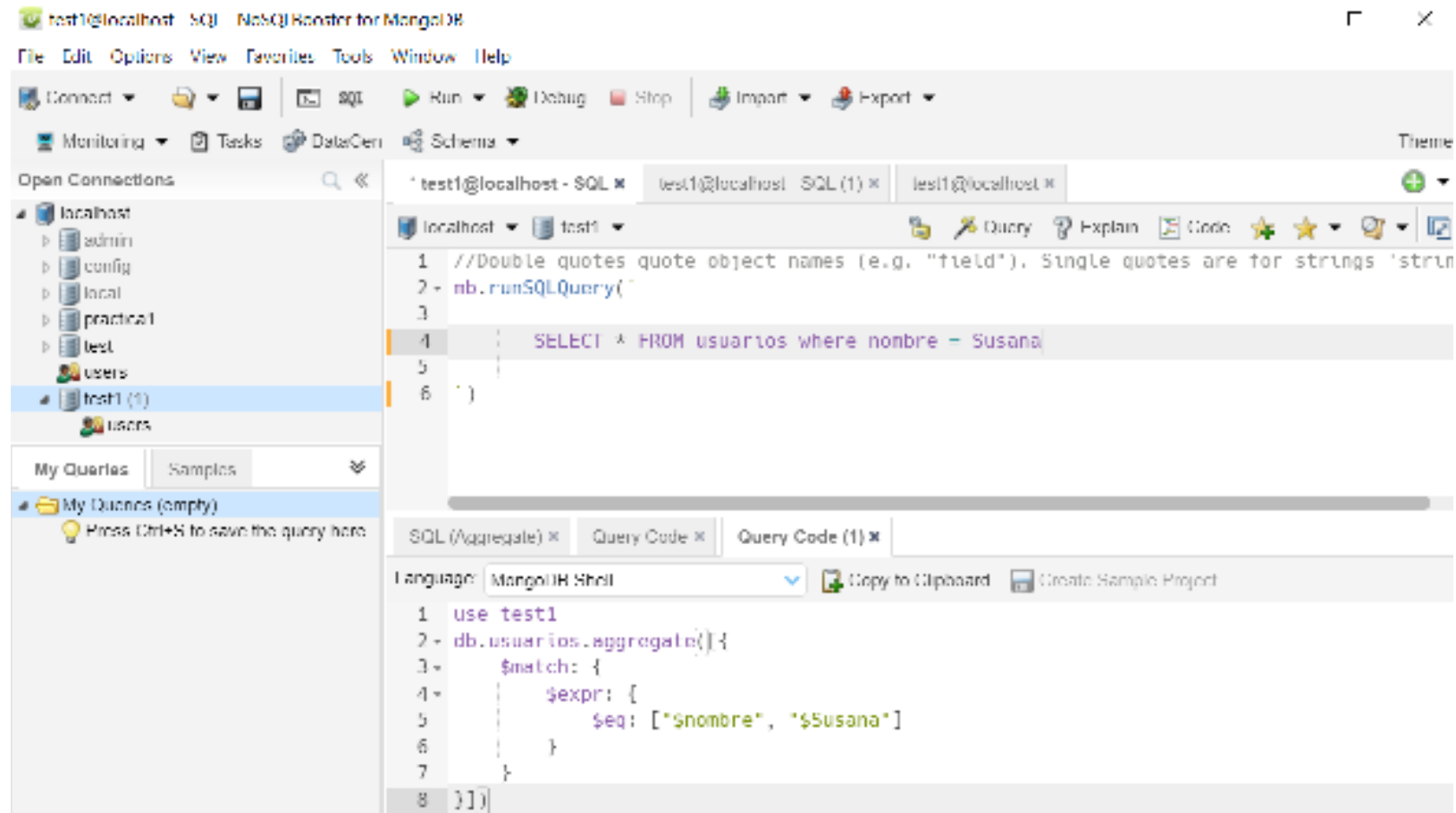
Copy to Clipboard Create Sample Project

```
1 use test1
2 db.usuarios.aggregate([
3   $sort: {
4     nombre: 1
5   }
6 ]])
```

Copyright © nosqlbooster.com Version 8.0.14 Free Edition

Line: 4, Column: 46 Show Log Feedback 9:41:34 PM

SQL en NoSQL



The screenshot shows the NoSQL Ronter for MongoDB application interface. The top menu bar includes File, Edit, Options, View, Favorites, Tools, Window, and Help. Below the menu is a toolbar with icons for Connect, Import, Export, Run, Debug, Stop, and other functions. The left sidebar displays a tree view of the database structure, including collections like admin, config, local, practical1, test, users, and test1 (1). The main editor area shows a SQL query being entered: `SELECT * FROM usuarios where nombre = Susana`. The bottom panel shows the MongoDB Shell code for the same query: `use test1; db.usuarios.aggregate({$match: {$expr: {$eq: ['$nombre', '$Susana']}}});`

```
1 //Double quotes quote object names (e.g. "field"). Single quotes are for strings 'string'
2 - nb.runSQLQuery(
3
4     SELECT * FROM usuarios where nombre = Susana
5
6 )
```

```
1 use test1
2 - db.usuarios.aggregate({
3     $match: {
4         $expr: {
5             $eq: ['$nombre', '$Susana']
6         }
7     }
8 })
```

Actualizar un Documento



The screenshot shows the NoSQLBooster for MongoDB interface. The left sidebar displays the database structure with 'test1 (1)' selected. The main editor shows a MongoDB query to find a user and update their email. The bottom panel shows the result of the query, a document for a user named Carlos with email 'car@gmail.com' and age 28.

```
1 db.usuarios.find( )
2
3 db.usuarios.updateOne(
4   _id:ObjectId( "64bd96deac1490ff49fd01eb" )
5
6 ), { $set: { email: "car@gmail.com" } })
```

Key	Value	Type
(1) 64bd96deac1490ff49fd01eb	{ nombre: "Carlos", email: "car@gmail.com", edad: 28 } (4 fields)	Document
id	64bd96deac1490ff49fd01eb	ObjectId
nombre	Carlos	String
email	car@gmail.com	String
edad	28	Int32

Usamos db.usuarios.
updateMany

Para modificar varios.



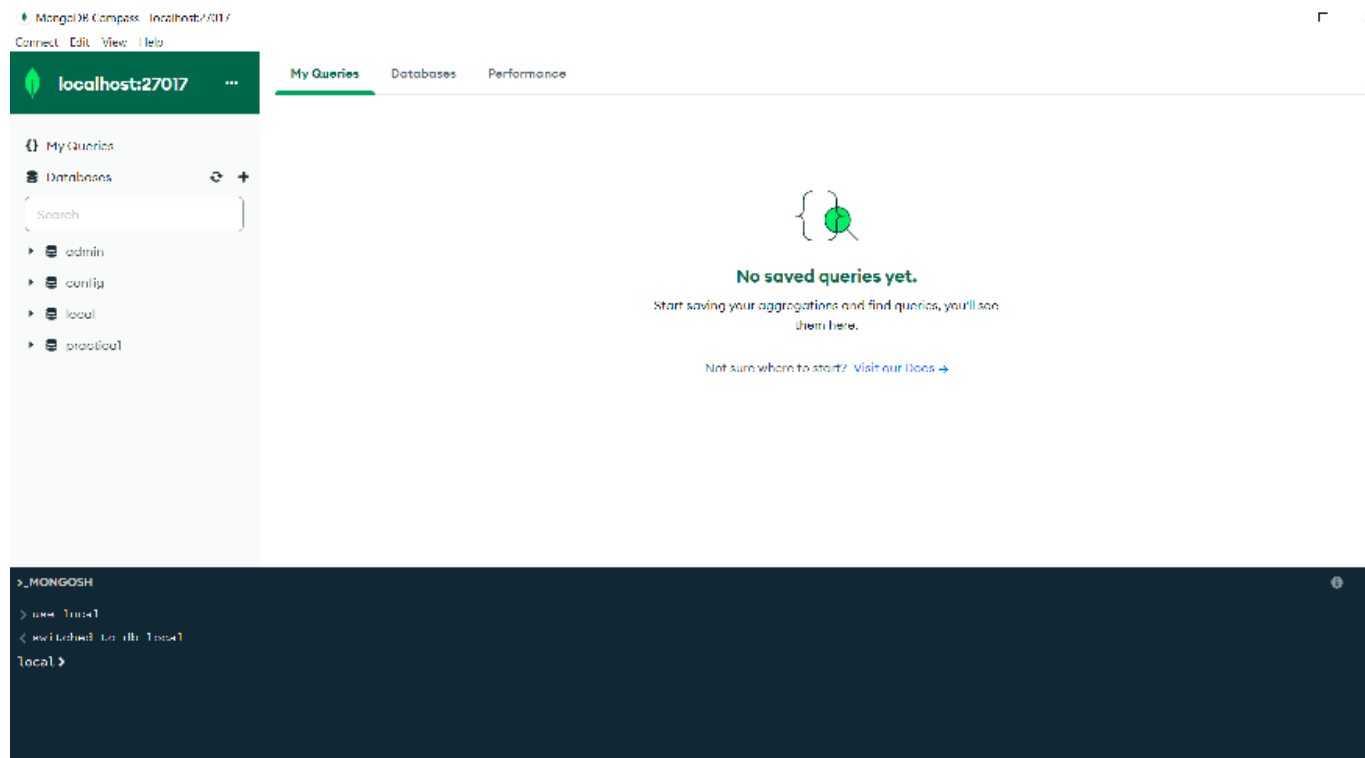
ELIMINAR DATOS

```
db.usuarios.deleteOne({id_ObjectId("64bd97caac1490ff49fd01ed")})
```

```
db.usuarios.deleteOne({email:car@Gmail.com})})
```

CREAR BASE DE DATOS

Para pasar de una base de datos a otra usamos la palabra reservada use.



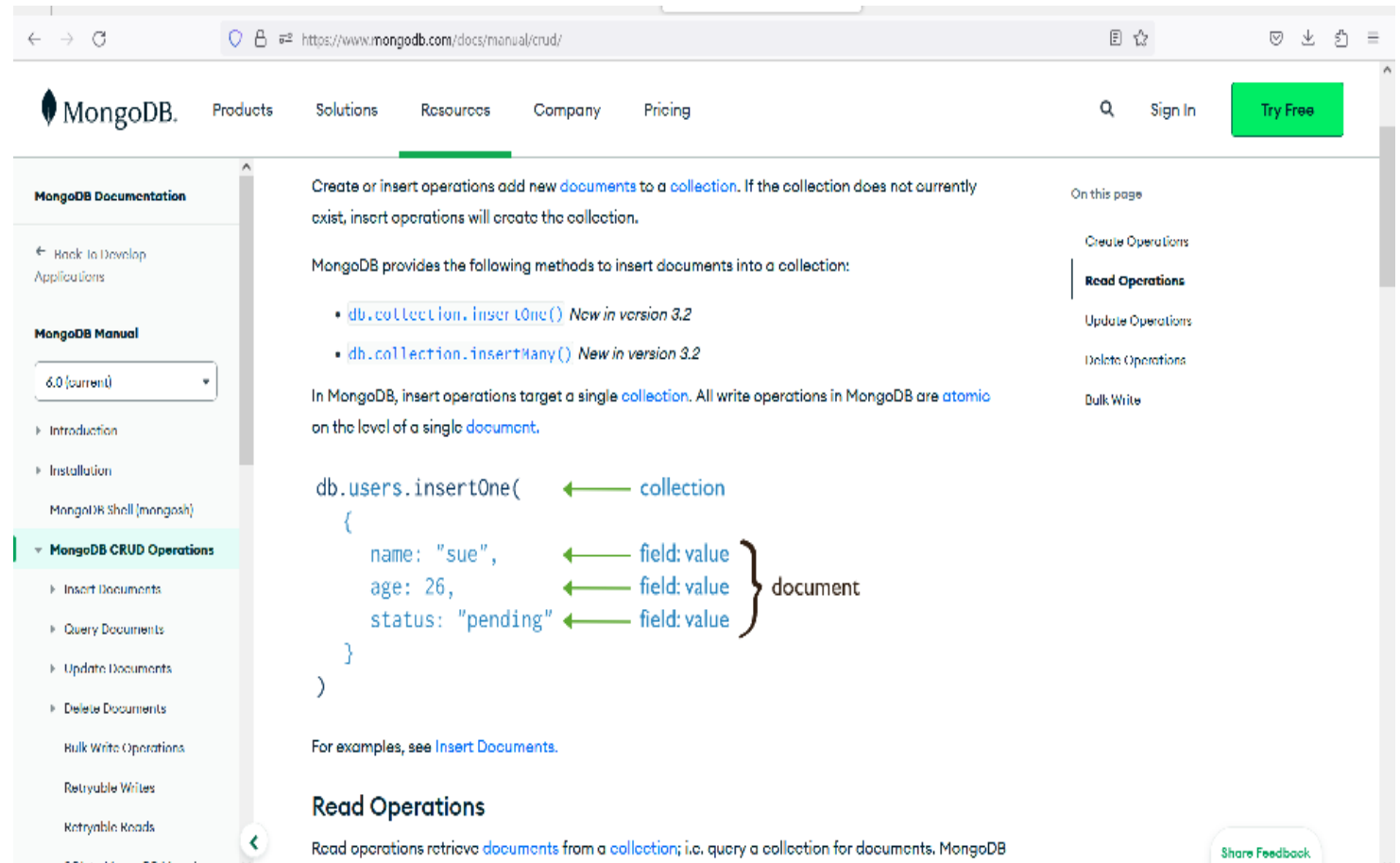
Para limpiar el entorno usamos la palabra reservada `use`.

CREAR COLECCIONES

Para crear una colección
la podemos hacer de varias
formas:

```
db.collection.insertOne()  
db.collection.insertMany()
```

El esquema es muy flexible.



The screenshot shows the MongoDB documentation page for CRUD operations. The left sidebar contains a navigation menu with sections like 'MongoDB Documentation', 'MongoDB Manual', and 'MongoDB CRUD Operations'. The main content area explains that create or insert operations add new documents to a collection and lists methods like `db.collection.insertOne()` and `db.collection.insertMany()`. It also shows an example of an `insertOne` operation with a document structure and labels for its components: 'collection' for the database name, 'field: value' for each key-value pair, and 'document' for the entire object. The right sidebar lists 'On this page' links for Create, Read, Update, Delete, and Bulk Write operations.

MongoDB Documentation

Products Solutions Resources Company Pricing

Search Sign In Try Free

MongoDB Documentation

Back to Develop Applications

MongoDB Manual

6.0 (current)

Introduction

Installation

MongoDB Shell (mongosh)

MongoDB CRUD Operations

Insert Documents

Query Documents

Update Documents

Delete Documents

Bulk Write Operations

Retryable Writes

Retryable Reads

6.0.1 to MongoDB Manual

Create or insert operations add new **documents** to a **collection**. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- `db.collection.insertOne()` *New in version 3.2*
- `db.collection.insertMany()` *New in version 3.2*

In MongoDB, insert operations target a single **collection**. All write operations in MongoDB are **atomic** on the level of a single **document**.

```
db.users.insertOne(  
  {  
    name: "sue",  
    age: 26,  
    status: "pending"  
  }  
)
```

← collection

← field: value

← field: value

← field: value

} document

For examples, see [Insert Documents](#).

Read Operations

Read operations retrieve **documents** from a **collection**; i.e. query a collection for documents. MongoDB

On this page

- Create Operations
- Read Operations**
- Update Operations
- Delete Operations
- Bulk Write

Share Feedback

CREAR COLECCIONES

Para ver las colecciones que tenemos usamos la palabra reservada :
Show collections

Para ver las bases de datos usamos la palabra reservada show databases.

```
>_MONGOSH
```

```
> db.createCollection("users")
```

```
< { ok: 1 }
```

```
> show collections
```

```
< users
```

```
> show databases
```

```
< admin      40.00 KiB
```

```
  config    108.00 KiB
```

```
  local      72.00 KiB
```

```
  system.js  0.00 KiB
```

INSERTAR DATOS EN UNA COLECCIÓN

Otra manera de crear colecciones es insertando un dato.

```
db.publicaciones.insertOne({})
```

(Aquí me dirá que inserté un valor desconocido)

```
db.publicaciones.insertOne({"titulo":"Clase se MongoDB",  
"duracion":"un trimestre"})
```

MOSTRAR LA INFORMACIÓN

Para mostrar la información agregada usamos la función `(.find)`
Pueden haber documentos distintos en una colección.

Si me desconecto y vuelvo a ingresar va a aparecer la base de datos test.
Porque ya tiene una colección.