

HITO 2 DEL 3º TRIMESTRE DE BASE DE DATOS

Alejandro Cortés Díaz

Índice

1. Redacta la introducción y la definición del problema de software propuesto.	2
Introducción:	2
Problema:	2
2. Dibuja un árbol jerárquico que represente la estructura de documentos de la base de datos MongoDB que vais a crear. Justifica la conveniencia de utilizar dicha estructura.	2
Justificación:	3
3. Crea la base de datos en MongoDB que resuelva el problema de software expuesto anteriormente.	3
4. Realiza las cuatro operaciones CRUD sobre los datos de alguna de las colecciones.	4
5. Realiza al menos 12 consultas utilizando find o findOne cumpliendo los siguientes requisitos:	5
o Se ha utilizado proyección en al menos 3 consultas.	5
o Se han aprovechado las referencias (asociación entre documentos) para recuperar información en al menos 3 consultas.	5
o Se han utilizado operadores especiales en al menos 3 consultas.	5
6. Desarrolla 3 ejemplos utilizando la función forEach().	7
7. Redacta un pequeño tutorial sobre el uso de MongoDB Compass.	11
¿Qué es MongoDB Compass?	11
8. Para finalizar, debes investigar por tu cuenta para subir la base de datos a la nube con MongoDB Atlas indicando paso a paso cómo lo habéis hecho y aportando recortes de pantalla.	13
Enlace a GitHub	20
Bibliografía	21

1. Redacta la introducción y la definición del problema de software propuesto.

Introducción:

En una biblioteca digital moderna, es fundamental llevar un registro eficiente de los libros disponibles y de los usuarios que los toman en préstamo. Esto implica almacenar información sobre los libros, sus autores, el estado de disponibilidad, y los préstamos realizados por los usuarios.

Problema:

- Se necesita una biblioteca digital que permita tener un catálogo de libros digitales, un registro de usuarios que puedan reservar libros, y datos sobre los libros pertinentes. Además de una relación entre colecciones a través de alguna referencia

2. Dibuja un árbol jerárquico que represente la estructura de documentos de la base de datos MongoDB que vais a crear. Justifica la conveniencia de utilizar dicha estructura.

bibliotecaDigital (base de datos)

├─ libros (colección)

| └─ documento

| ├─ _id

| ├─ titulo

| ├─ autor

| ├─ genero

| ├─ disponible (boolean)

├─ usuarios (colección)

| └─ documento

| └─ _id
| └─ nombre
| └─ correo
| └─ prestamos (array de IDs de libros)

Justificación:

- Permite gestionar libros y usuarios por separado.
- Relaciona usuarios con los libros que han tomado en préstamo, utilizando referencias (_id en mi caso).

3. Crea la base de datos en MongoDB que resuelva el problema de software expuesto anteriormente.

```
db = db.getSiblingDB("bibliotecaDigital");  
// Crear la base de datos y las inserciones iniciales  
db.libros.insertMany([  
  {  
    _id: 1,  
    titulo: "1984",  
    autor: "George Orwell",  
    genero: "Distopía",  
    disponible: true  
  },  
  {  
    _id: 2,  
    titulo: "Cien años de soledad",  
    autor: "Gabriel García Márquez",  
    genero: "Realismo mágico",  
    disponible: false  
  }  
]);
```

```
// Tambien con los usuarios
db.usuarios.insertMany([
  {
    _id: 101,
    nombre: "Luis Pérez",
    correo: "luisp@correo.com",
    prestamos: [2]
  },
  {
    _id: 102,
    nombre: "María Gómez",
    correo: "mariag@correo.com",
    prestamos: []
  }
]);
```

4. Realiza las cuatro operaciones CRUD sobre los datos de alguna de las colecciones.

```
//Inserto un nuevo usuario para eliminarlo luego en el CRUD
use("bibliotecaDigital")
db.usuarios.insertOne({
  _id: 103,
  nombre: "Carlos Díaz",
  correo: "carlosd@correo.com",
  prestamos: []
})
```

```
//COMIENZO DE CRUD
```

```
// Encontrar los libros disponibles  
use("bibliotecaDigital")  
db.libros.find({disponible: true})
```

```
//Cambiar el estado de un libro a no disponible  
use("bibliotecaDigital")  
db.libros.updateOne(  
  { _id: 1 },  
  { $set: { disponible: false } }  
)
```

```
// Eliminar un usuario  
use("bibliotecaDigital")  
db.usuarios.deleteOne({ _id: 103 })
```

5. Realiza al menos 12 consultas utilizando find o findOne cumpliendo los siguientes requisitos:

o Se ha utilizado proyección en al menos 3 consultas.

o Se han aprovechado las referencias (asociación entre documentos) para recuperar información en al menos 3 consultas.

o Se han utilizado operadores especiales en al menos 3 consultas.

//CONSULTAS FIND O FINDONE

```
//Encontrar a todos los usuarios mostrando solo su nombre y prestamos  
use("bibliotecaDigital")  
db.usuarios.find({}, {nombre: 1, prestamos: 1})
```

```
// Encontrar todos los libros mostrando solo su título y autor sin el _id
```

```

use("bibliotecaDigital")
db.libros.find({}, {titulo: 1, autor: 1, _id: 0})

// Mostrar el nombre del usuario con ID 101 pero sin el _id
use("bibliotecaDigital")
db.usuarios.find({_id: 101}, {_id: 0, nombre: 1})

// Encontrar todos los libros prestados por el usuario con ID 101
use("bibliotecaDigital")
const usuario = db.usuarios.findOne({_id: 101})
db.libros.find({_id: {$in: usuario.prestamos}})

//Encontrar los prestamos de ese libro en particular
use("bibliotecaDigital")
const libro = db.libros.findOne({titulo: "Cien años de soledad"})
db.usuarios.find({prestamos: libro._id})

// Encontrar los prestamos junto con el título de "Luis Pérez"
use("bibliotecaDigital")
const usuario = db.usuarios.findOne({nombre: "Luis Pérez"})
db.libros.find({_id: {$in: usuario.prestamos}}, {titulo: 1})

//Busco cualquier libro que contenga la palabra "mágico" en su genero
use("bibliotecaDigital")
db.libros.find({genero: {$regex: /mágico/}})

//Busco usuarios que no tengan prestamos
use("bibliotecaDigital")
db.usuarios.find({prestamos: {$size: 0}})

//Devuelvo todos los libros que tengan el campo "disponible"

```

```

use("bibliotecaDigital")
db.libros.find({disponible: {$exists: true}})

//Busco los datos de María Gómez
use("bibliotecaDigital")
db.usuarios.find({nombre: "María Gómez"})

// Busco todos los libros de George Orwell
use("bibliotecaDigital")
db.libros.find({autor: "George Orwell"})

// Busco los libros no disponibles
use("bibliotecaDigital")
db.libros.find({disponible: false})

```

6. Desarrolla 3 ejemplos utilizando la función forEach().

```

// CONSULTAS CON FOREACH

//Recorre los documentos de usuarios y muestra el nombre de cada uno
use("bibliotecaDigital")
db.usuarios.find().forEach(u => print("Usuario: " + u.nombre));

// Recorre libros y muestra el titulo y el genero de cada uno
use("bibliotecaDigital")
db.libros.find().forEach(l => print(`Libro: ${l.titulo} (${l.genero})`));

// Comprueba si los usuarios tienen libros prestados, en cuyo caso,
// Imprime por consola
use("bibliotecaDigital")
db.usuarios.find().forEach(u => {

```



```

if (u.prestamos.length > 0) {

    print(`${u.nombre} tiene libros prestados.`);

}

});

```

```

Connected to localhost:27017 | ✨ Generate query with MongoDB Copilot
1 db = db.getSiblingDB("bibliotecaDigital");
2 // Crear la base de datos y las inserciones iniciales
3 db.libros.insertMany([
4   {
5     _id: 1,
6     titulo: "1984",
7     autor: "George Orwell",
8     genero: "Distopía",
9     disponible: true
10  },
11  {
12    _id: 2,
13    titulo: "Cien años de soledad",
14    autor: "Gabriel García Márquez",
15    genero: "Realismo mágico",
16    disponible: false
17  }
18  ]);
19 // También con los usuarios
20 db.usuarios.insertMany([
21   {
22     _id: 101,
23     nombre: "Luis Pérez",
24     correo: "luisp@correo.com",
25     prestamos: [2]
26   },
27   {
28     _id: 102,
29     nombre: "María Gómez",
30     correo: "mariag@correo.com",
31     prestamos: []
32   }
33  ]);
34
35 // Inserto un nuevo usuario para eliminarlo luego en el CRUD
36 use("bibliotecaDigital")
37 db.usuarios.insertOne({
38   _id: 103,
39   nombre: "Carlos Díaz",
40   correo: "carlosd@correo.com",
41   prestamos: []

```

```

//COMIENZO DE CRUD

// Encontrar los libros disponibles
use("bibliotecaDigital")
db.libros.find({disponible: true})

//Cambiar el estado de un libro a no disponible
use("bibliotecaDigital")
db.libros.updateOne(
  { _id: 1 },
  { $set: { disponible: false } }
)

// Eliminar un usuario
use("bibliotecaDigital")
db.usuarios.deleteOne({ _id: 103 })

//CONSULTAS FIND O FINDONE

//Encontrar a todos los usuarios mostrando solo su nombre y prestamos
use("bibliotecaDigital")
db.usuarios.find({}, {nombre: 1, prestamos: 1})

// Encontrar todos los libros mostrando solo su título y autor sin el _id
use("bibliotecaDigital")
db.libros.find({}, {titulo: 1, autor: 1, _id: 0})

// Mostrar el nombre del usuario con ID 101 pero sin el _id
use("bibliotecaDigital")
db.usuarios.find({_id: 101}, {_id: 0, nombre: 1})

// Encontrar todos los libros prestados por el usuario con ID 101
use("bibliotecaDigital")
const usuario = db.usuarios.findOne({_id: 101})
db.libros.find({_id: {$in: usuario.prestamos}})

//Encontrar los prestamos de ese libro en particular
use("bibliotecaDigital")
const libro = db.libros.findOne({titulo: "Cien años de soledad"})
db.usuarios.find({prestamos: libro._id})

```

```

// Encontrar los prestamos junto con el título de "Lius Perez"
use("bibliotecaDigital")
const usuario = db.usuarios.findOne({nombre: "Luis Pérez"})
db.libros.find({_id: {$in: usuario.prestamos}}, {titulo: 1})

//Busco cualquier libro que contenga la palabra "mágico" en su genero
use("bibliotecaDigital")
db.libros.find({genero: {$regex: /mágico/}})

//Busco usuarios que no tengan prestamos
use("bibliotecaDigital")
db.usuarios.find({prestamos: {$size: 0}})

//Devuelvo todos los libros que tengan el campo "disponible"
use("bibliotecaDigital")
db.libros.find({disponible: {$exists: true}})

//Busco los datos de María Gómez
use("bibliotecaDigital")
db.usuarios.find({nombre: "María Gómez"})

// Busco todos los libros de George Orwell
use("bibliotecaDigital")
db.libros.find({autor: "George Orwell"})

// Busco los libros no disponibles
use("bibliotecaDigital")
db.libros.find({disponible: false})

✓ // CONSULTAS CON FOREACH

//Recorre los documentos de usuarios y muestra el nombre de cada uno
use("bibliotecaDigital")
✓ db.usuarios.find().forEach(u => print("Usuario: " + u.nombre));

// Recorre libros y muestra el título y el genero de cada uno
use("bibliotecaDigital")
db.libros.find().forEach(l => print(`Libro: ${l.titulo} (${l.genero})`));

// Comprueba si los usuarios tienen libros prestados, en cuyo caso,
✓ // Imprime por consola

```

```

// Comprueba si los usuarios tienen libros prestados, en cuyo caso,
// Imprime por consola
use("bibliotecaDigital")
db.usuarios.find().forEach(u => {
  if (u.prestamos.length > 0) {
    print(`${u.nombre} tiene libros prestados.`);
  }
});

```

7. Redacta un pequeño tutorial sobre el uso de MongoDB Compass.

¿Qué es MongoDB Compass?

MongoDB Compass es la interfaz gráfica oficial de MongoDB. Permite visualizar, consultar, modificar y administrar bases de datos sin escribir comandos en consola, facilitando el trabajo.

- Para conectarnos al servidor tenemos que escribir, en “conexión” la URI “mongodb://localhost:27017” , luego en “Connect” para conectar.

Una vez conectado:

- En la columna izquierda aparecen todas las bases de datos disponibles.
 - Clickando en una base de datos se pueden ver sus colecciones.
 - Clickando en una colección se pueden ver los documentos que contiene.
-

Para realizar consultas;

- En la parte superior de la vista de colección hay una caja de texto llamada “Filter”. Se pueden escribir consultas similares a las del shell
 - Puedes escribir consultas similares a las de la shell. Ejemplos:
-

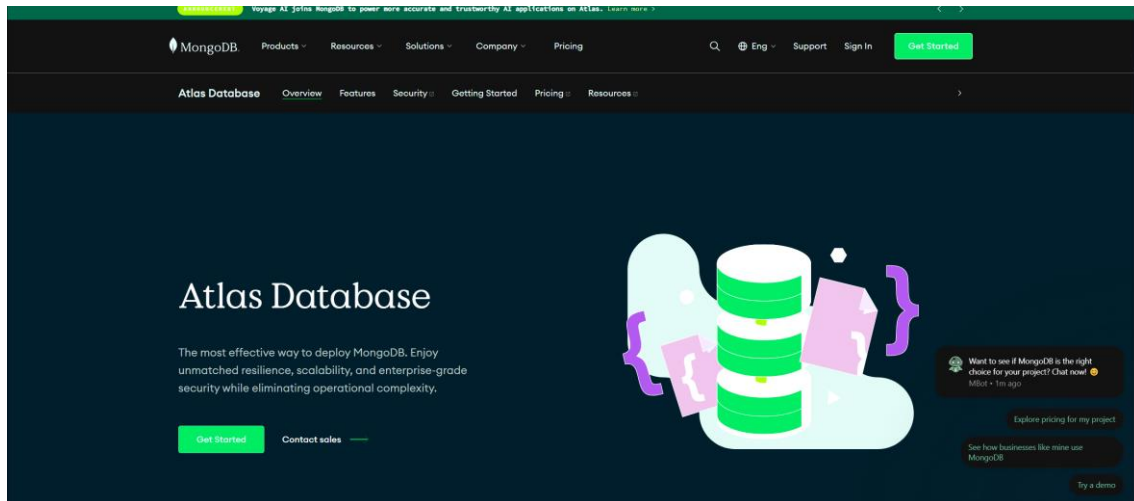
Para Insertar, editar y eliminar documentos;

- En “ADD DATA” e “Insert Document” se puede agregar un nuevo documento.
- En el botón lápiz junto a un documento se puede editar.
- Y en la papelera para eliminarlo.

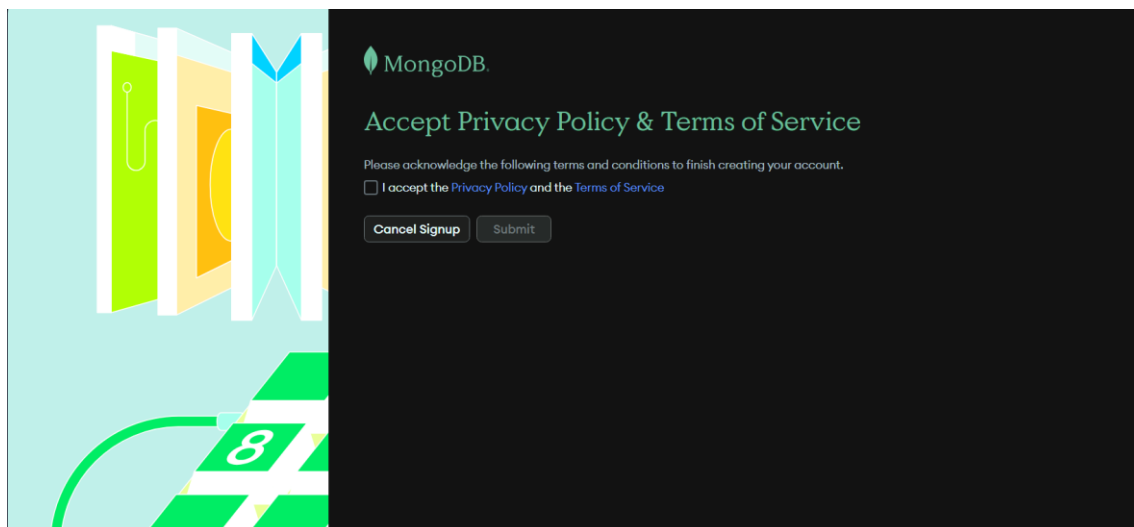
-
- Cabe destacar que con Compass podemos;
 - Crear bases de datos y colecciones.
 - Insertar documentos.
 - Actualizar registros.
 - Borrar documentos.
-
- Básicamente un CRUD
-

8. Para finalizar, debes investigar por tu cuenta para subir la base de datos a la nube con MongoDB Atlas indicando paso a paso cómo lo habéis hecho y aportando recortes de pantalla.

1. Accedo a <https://www.mongodb.com/cloud/atlas>



2. Cree una cuenta y acepté la política de privacidad



3. Continuo con las especificaciones de la web



Welcome to Atlas. Let's build something great.

Provide your goals, coding language preferences, and project details to get specific tools and guidance for your project needs.

GETTING TO KNOW YOU

What is your primary goal?

Learn MongoDB

How long have you been developing software with MongoDB?

I've never developed software with MongoDB before

GETTING TO KNOW YOUR PROJECT

What programming language are you primarily building on MongoDB with?

Not sure

What type(s) of data will your project use?

You can choose as many as you want

Not sure...

Will your application include any of the following architectural models?

You can choose as many as you want

Not sure...

[Skip personalization](#)

Finish

4. Selecciono el cluster gratuito

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10

\$0.09/hour

Dedicated cluster for development environments and low-traffic applications.

STORAGE

10 GB

RAM

2 GB

vCPU

2 vCPUs

Flex

From \$0.011/hour
Up to \$30/month

For application development and testing, with on-demand burst capacity for unpredictable traffic.

STORAGE

5 GB

RAM

Shared

vCPU

Shared

Free

For learning and exploring MongoDB in a cloud environment.

STORAGE

512 MB

RAM

Shared

vCPU

Shared

Pay-as-you-go! You will be billed hourly and can terminate your cluster anytime. Excludes variable data transfer, backup, and taxes.

Configurations

Name
You cannot change the name once the cluster is created.

Cluster0

Provider

aws

Google Cloud

Azure

Region

Spain (eu-south-2)

★

🌿

★ Recommended ⓘ 🌿 Low carbon emissions ⓘ

Quick setup

☐ Preload sample dataset ⓘ

I'll do this later

Go to Advanced Configuration

Create Deployment

5. Especifico nombre de usuario y contraseña

Connect to Cluster0

1 Set up connection security 2 Choose a connection method 3 Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

✓ Your current IP address (90.166.249.171) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access](#).

2. Create a database user

This first user will have [atlasAdmin](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

i You'll need your database user's credentials in the next step. Copy the database user password.

Username Password

alejandrocortesdiaz24 gAldcL5lDpaf8F98 HIDE Copy

Create Database User

Close Choose a connection method

6. Continuo con el método de conexión, elijo, Compass

Connect to Cluster0

✓

✓

3

Set up connection securityChoose a connection methodConnect

Connecting with MongoDB Compass

I don't have MongoDB Compass installed

I have MongoDB Compass installed

1. Choose your version of Compass

1.38 or later

See your Compass version in "About Compass"

2. Copy the connection string, then open MongoDB Compass

Show Password ⓘ

Use this connection string in your application

mongodb+srv://alejandrocortesdiaz24:gAldcL5lDpaf8F98@cluster0.81sggsj.mongodb.net/

The password for **alejandrocortesdiaz24** is included in the connection string for your first time setup. **This password will not be available again after exiting this connect flow.**

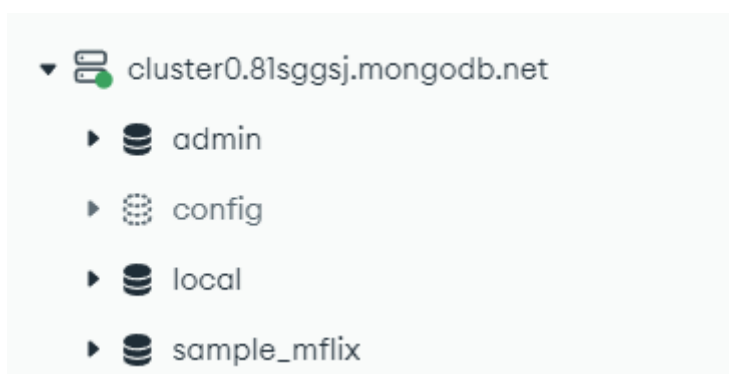
RESOURCES

[Connect with Compass](#)[Import and Export Data](#)[Access your Database Users](#)[Troubleshoot Connections](#)

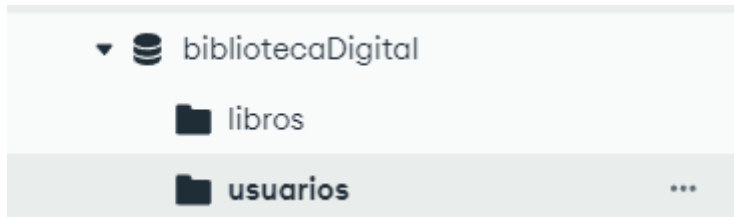
Go Back

Done

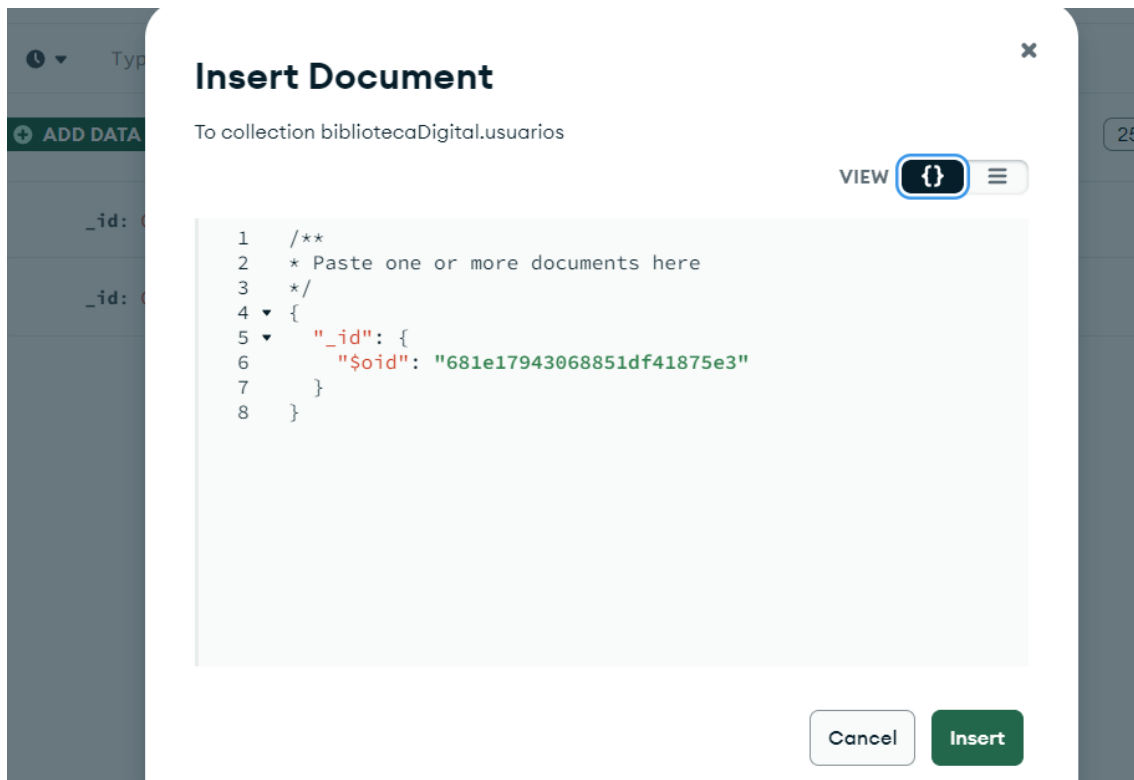
7. Copio el URI en Compass y se accede a través de la aplicación



8. Volví a crear la base de datos desde compass e introduje manualmente las colecciones



9. Dentro de cada una, introduje los datos de esta forma



- En "Paste one or more documents here" introduce cada inserción de cada libro y cada usuario, posteriormente se insertan. Terminando así;

cluster0.81sggsj.mongodb.net > bibliotecaDigital > libros Open MongoDB shell

Documents 2 Aggregations Schema Indexes 1 Validation

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

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 2 of 2 ↺ ↻ ↷ ☰ { } ⌂

`_id: ObjectId('681e16f53968851df41875de')`

`_id: ObjectId('681e17263968851df41875df')`

cluster0.81sggsj.mongodb.net > bibliotecaDigital > usuarios Open MongoDB shell

Documents 3 Aggregations Schema Indexes 1 Validation

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

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 3 of 3 ↺ ↻ ↷ ☰ { } ⌂

`_id: ObjectId('681e17363968851df41875e1')`

`_id: ObjectId('681e174b3968851df41875e2')`

`_id: ObjectId('681e17c63968851df41875e4')`

Enlace a GitHub

<https://github.com/Cortes-cmd/BBDD.git>

Bibliografía

Db.Collection.insertOne(). (s/f). Mongodb.com. Recuperado el 9 de mayo de 2025, de
<https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/>

Moisset, D. (s/f). Insertar documentos mediante los métodos insertOne e insertMany de una colección. Tutorialesprogramacionya.com. Recuperado el 9 de mayo de 2025, de
<https://www.tutorialesprogramacionya.com/mongodbya/detalleconcepto.php?punto=4&codigo=4&inicio=0>

Mongodb cursor's forEach does not execute. (2018, junio 22). Meteor Forum. https://forums.meteor.com/t/mongodb-cursors-foreach-does-not-execute/44204

MongoDB insertOne. (2020, agosto 10). MongoDB Tutorial. https://www.mongodbtutorial.org/mongodb-crud/mongodb-insertone/

MongoDB mongosh insert. (s/f). W3schools.com. Recuperado el 9 de mayo de 2025, de
https://www.w3schools.com/mongodb/mongodb_mongosh_insert.php

Yatin. (2018, febrero 28). MongoDB forEach() example. Examples Java Code Geeks; Exelixis Media P.C. https://examples.javacodegeeks.com/software-development/mongodb/mongodb-foreach-example/

(S/f). Stackoverflow.com. Recuperado el 9 de mayo de 2025, de
<https://stackoverflow.com/questions/22656517/update-in-foreach-on-mongodb-shell>