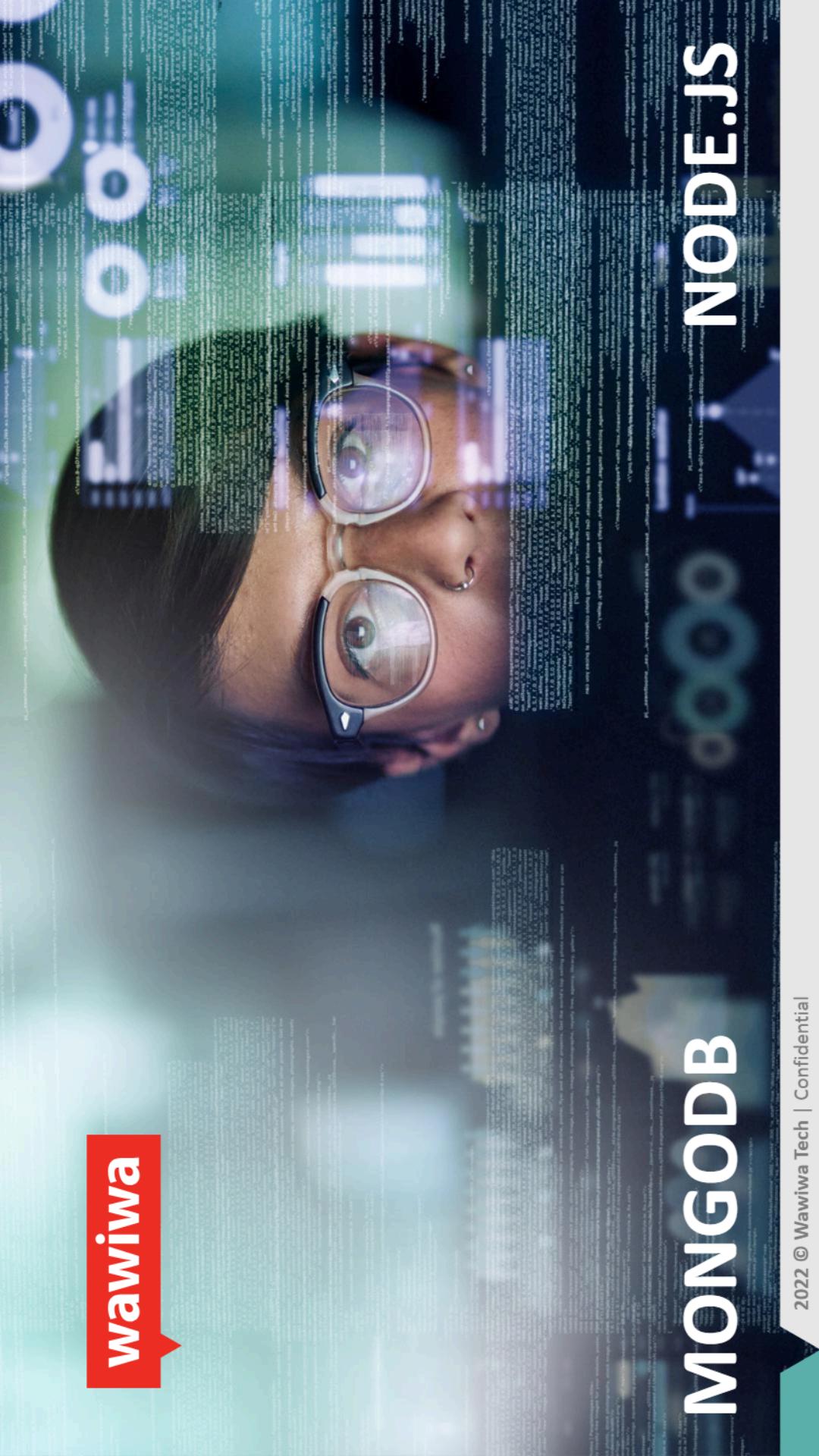


NODE.JS

MONGODB

wawiwa



¿Qué es mongodB?

MongoDB Atlas es un servicio de base de datos multinube creado por las mismas personas que crean MongoDB. Atlas simplifica la implementación y administración de sus bases de datos y, al mismo tiempo, ofrece la versatilidad que necesita para crear aplicaciones globales resilientes y de alto rendimiento en los proveedores de nube de su elección.

Database, collection, document

Database (base de datos)

Database es un contenedor físico para colecciones. Cada base de dato obtiene su conjunto de archivos en el sistema de archivos. Un único servidor MongoDB normalmente tiene varias bases de datos.

Collection (colección)

Collection es un grupo de documentos de MongoDB. Una colección existe en una sola base de datos. Los documentos en una colección pueden tener diferentes campos. MongoDB almacena documentos en colecciones. Las colecciones son análogas a las tablas en bases de datos relacionales.

Document (documento)

Un documento es un conjunto de pares clave-valor. Los documentos tienen un esquema dinámico. El esquema dinámico significa que los documentos de la misma colección no necesitan tener el mismo conjunto de campos o estructura y los campos comunes en los documentos de una colección pueden contener diferentes tipos de datos. MongoDB almacena registros de datos como documentos BSON. BSON es una representación binaria de documentos JSON, aunque contiene más tipos de datos que [JSON](#). Para conocer las especificaciones BSON, consulte bsontspec.org Consulta también [BSON Types](#).

MongoDB - Registrarse

1. Ve a <https://docs.atlas.mongodb.com/tutorial/create-atlas-account/> y crea una cuenta
2. Implementa el clúster gratuito
<https://docs.atlas.mongodb.com/tutorial/deploy-free-tier-cluster/>
3. Agrega tu dirección de ip a la lista de accesos
<https://docs.atlas.mongodb.com/security/add-ip-address-to-list/>

Estructura del Documento

```
{  
    field1: value1,  
    field2: value2,  
    field3: value3,  
    ...  
    fieldN: valueN  
}
```

Ejemplo de la estructura del Documento

```
const mydoc = {  
    id: ObjectId("5099803df3f4948bd2f98391"),  
    name: { first: "Alan", last: "Turing" },  
    birth: new Date('Jun 23, 1912'),  
    death: new Date('Jun 07, 1954'),  
    contribs: [ "Turing machine", "Turing test", "Turingery" ],  
    views : NumberLong(1250000)  
};
```

Los campos de arriba tienen los siguientes tipos de información :

- id** tiene un ObjectId – id es un número hexadecimal de 12 bytes que asegura la unicidad de cada documento.
- name** tiene un documento incorporado que contiene los campos first y last.
- birth** y **death** tienen valores de tipo Date.
- contribs** tiene un arreglo de strings.
- views** tiene un valor de tipo NumberLong.

Campos de arreglos de documentos

Para especificar o acceder un elemento de un arreglo por la posición del índice de base cero, concatena nombre del arreglo con un punto (.) y posición del índice de base cero y encerrar entre comillas: "`<array>. <index>`"

Por ejemplo, dado el siguiente campo en un documento:

```
{  
    ...  
    contribs: [ "Turing machine", "Turing test", "Turingery" ],  
    ...  
}
```

Documentos incorporados

Para especificar o acceder a un campo de un documento incrustado con notación de puntos, concatene el nombre del documento incrustado con el punto (.) y el nombre del campo encerrado en comillas: "`<embedded document>.<field>`"

Por ejemplo, dado el siguiente campo en un documento:

```
{  
  ...  
  name: { first: "Alan", last: "Turing" },  
  contact: { phone: { type: "cell", number: "111-222-3333" } },  
  ...  
}
```

- Para especificar el campo llamado "last" en el campo "name", usa la notación de punto "name.last".
- Para especificar el número en el documento "phone" en el campo "contact", usa la notación de punto "contact.phone.number".

Introducción de CRUD en MongoDB

CRUD operaciones *create, read, update y delete*

Las operaciones para crear o insertar agregan nuevos documentos a la colección. Si la colección no existe actualmente, las operaciones de insert van a crear la colección.

Colecciones y bases de datos

MongoDB almacena registros de datos como documentos (específicamente documentos BSON) que están reunidos en colecciones. Una base de datos almacena una o más colecciones de documentos.

Crea una base de datos

Crea una base de datos

Si una base de datos no existe, MongoDB crea la base de datos cuando por primera vez almacenas información para esa base de datos. Como tal, puedes cambiar a una base de datos no existente y ejecutar la siguiente operación en [mongosh](#)

Puedes navegar en

<https://docs.mongodb.com/manual/tutorial/getting-started/>

Para entrenar en línea: puedes: cambiar de base de datos, insertar, encontrar y filtrar datos

Click inside the shell to connect. Once connected, you can run the examples in the shell above.

1. Switch Database
2. Insert
3. Find All
4. Filter Data
5. Project Fields
6. Aggregate

wawiwa

CREA UNA DATABASE EN MONGODB MÁS DE MONGODB - [LINK](#)

Después de que te registres

Lista de usuarios y
pass que
tienen
acceso a la
base de
datos

Lista de ip's que tienen acceso a la base de datos

Project 0

DEPLOYMENT

Databases

Data Lake

DATA SERVICES

Triggers

Data API

PREVIEW

SECURITY

Database Access

Network Access

Advanced

BRAINWOP > PROJECT 0

Database Deployments

Find a database deployment...

Clusterio Connect View Monitoring Browse Collections ...

Atlas

Realm

Charts

+ Create

FREE SHARED

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrades

VERSION 4.4.10

REGION AWS / N. Virginia (us-east-1)

CLUSTER TIER Mo Sandbox (General)

TYPE Replica Set - 3 nodes

BACKUPS Inactive

LINKED REALM APP None Linked

ATLAS SEARCH Create Index

Después de que te registres - Crea

Haz click en
connect

The screenshot shows the MongoDB Atlas interface. At the top, there are tabs for Project 0, Atlas, Realm, and Charts. Below these are buttons for Create, Refresh, and a bell icon. The main navigation bar includes PROJECT > PROJECT 0, DEPLOYMENT, Databases, and PREVIEW. Under Databases, there are sections for DATA SERVICES (Triggers, Data API), SECURITY (Database Access, Network Access, Advanced), and a preview section for Cluster 0. The preview section shows connection metrics (0 R, 0 W, Last 16 minutes, 100.0%), a graph of throughput (In: 0.0 B/s, Out: 0.0 B/s, Last 11 minutes, 6.9 B/s), and a graph of data size (512.0 MB). There are also tabs for Connect, View Monitoring, Browse Collections, and an ellipsis. A green button labeled 'Upgrade' is visible. The right side of the interface shows a summary table:

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
4.4.10	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None	Create Index

After signup create

We will connect with
nodes

The screenshot shows a software window titled 'Cluster Connection' with a progress bar at the top. The main area is titled 'Choose a connection method'. It includes a sub-section 'Choose a connection method' with a 'View documentation' link. Below this, a note says 'Get your pre-formatted connection string by selecting your tool below.' Three options are listed: 'Connect with the MongoDB Shell' (selected), 'Connect your application' (disabled), and 'Connect using MongoDB Compass' (disabled). A 'Go Back' button is at the bottom right.

Connect to Cluster0

✓ Setup connection security > Choose a connection method > Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.

Connect with the MongoDB Shell

Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application

Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI

Go Back

Close

Después de que te registres - Crea

Nos vamos a conectar con nodejs

Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

1 Select your driver and version

DRIVER	VERSION
Node.js	4.0 or later
Node.js	into your application code
Perl	sample
PHP	example
Python	password>@cluster0,hqfhv.mongodb.net/myFirstDatabase?
Ruby	majority

2

Replace “`password`” with the password for the `ssaf` user. Replace `myFirstDatabase` with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

Después de que te registres - Crea

Copia el código

✓ Setup connection security > ✓ Choose a connection method > Connect

① Select your driver and version

DRIVER: Node.js VERSION: 4.0 or later

② Add your connection string into your application code

Include full driver code example

```
const { MongoClient } = require('mongodb');
const uri = "mongodb+srv://asaf:<password>@cluster0.hqfhv.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });
client.connect(err => {
  const collection = client.db("test").collection("devices");
  // perform actions on the collection object
  client.close();
});
```

Replace `<password>` with the password for the `asaf` user. Replace `myFirstDatabase` with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? [View our troubleshooting documentation](#)

Crea a1.js y pega el código

Navega los datos

The screenshot shows the MongoDB Atlas interface. At the top, there are navigation icons for Project 0, a dropdown menu, and a search bar. Below the search bar, there are tabs for DEPLOYMENT (selected), DATA SERVICES, and SECURITY. The DEPLOYMENT tab shows a deployment named "BRAINS.ORG - 2021-12-19 > PROJECT 0". The "Database Deployments" section contains a table with one row. The table columns are: VERSION (4.4.10), REGION (AWS / N. Virginia (us-east-1)), CLUSTER TIER (No Sandbox (General)), TYPE (Replica Set - 3 nodes), BACKUPS (Inactive), LINKED REALM APP (None linked), and ATLAS SEARCH (Create Index). A green button labeled "+ Create" is located at the top right of the deployment card. A green box highlights the text "Navega los datos" in the bottom left corner of the screenshot.

Crea a1.js y pega el código

The screenshot shows the MongoDB Atlas Cluster0 interface. At the top, there are tabs for Project, Deployment, Databases, Data Services, Triggers, Data API, Preview, Security, Database Access, Network Access, and Advanced. The 'Collections' tab is currently selected. Below the tabs, it says 'DATABASES: 0 COLLECTIONS: 0'. To the right, there's a sidebar with 'REGION AWS N. Virginia (us-east-1)', 'VERSION 4.4.10', 'Performance Advisor', 'Online Archive', 'Command', and a 'REFRESH' button. A 'VISUALIZE YOUR DATA' button is also present. On the far right, there's a 'Learn more in Docs and Tutorials' link with a 'next' icon. At the bottom right, there are buttons for 'Load a Sample Dataset' and 'Add My Own Data'.

Project ▾ : Atlas Realm Charts

DEPLOYMENT

BRAINS.ORG - 2021-12-19 > PROJECT 0 > DATABASES

Databases Cluster0

DATA SERVICES

Triggers

Data API PREVIEW

SECURITY SECURITY

Database Access

Network Access

Advanced

REGION AWS N. Virginia (us-east-1)

VERSION 4.4.10

Performance Advisor

Online Archive

Command

REFRESH

VISUALIZE YOUR DATA

Add My Own Data

Load a Sample Dataset

Learn more in Docs and Tutorials

Crea a1.js y pega el código

Crea una nueva carpeta llamada mongo y corre **npm init en la terminal y después **npm install mongodb**. Y agrega este código.**

```
const { MongoClient } = require('mongodb');
const uri =
  "mongodb+srv://asaf:<password>@cluster0.hqfhv.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });
client.connect(err => {
  const collection = client.db("test").collection("devices");
  // perform actions on the collection object
  client.close();
});
```

String de conexión

Agrega tu contraseña al string de conexión

Collection name

Db name

wawiwa Crea a1.js y pega el código

```
const { MongoClient } = require('mongodb');

const uri =
  "mongodb+srv://asaf@cluster0.hqfhv.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"

const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

client.connect(async (err) => {
  if (err) console.error(err);
  const db = client.db("test");
  // Usa la colección "people"
  const col = db.collection("people");

  // Construye un document
  const personDocument = {
    "name": { "first": "Alan", "last": "Turing" },
    "birth": new Date(1912, 5, 23), // June 23, 1912
    "death": new Date(1954, 5, 7), // June 7, 1954
    "contribs": [ "Turing machine", "Turing test", "Turingery" ],
    "views": 1250000
  };
  const p = await col.insertOne(personDocument);

  client.close();
});
```

Crea una **db** llamada "test"

Crea una **collection** llamada "people"

Crea el **document** : personDocument

Insert to people **collection**

Crea a1.js y pega el código

The screenshot shows the MongoDB Atlas Cluster0 dashboard. At the top, there are tabs for Project 0, Deployment, Databases, Data Services, and Data API. The Databases tab is active, showing a list of databases: Brain's Org (2021-12-19), Project 0, and Databases. Below this, there are tabs for Overview, Real Time, Metrics, Collections (which is underlined in green), Profiler, Search, Performance Advisor, Online Archive, and Command. On the left, there are sections for Data Lake, Data Services, SECURITY (with PREVIEW), and Database Access (with Network Access and Advanced). The main area is titled "Explore Your Data" with a magnifying glass icon. It shows a table with columns: _id, name, type, and size. There are 0 rows listed. At the bottom right of the dashboard, there is a "REFRESH" button. A large red arrow points from the "Explore Your Data" section towards the "REFRESH" button.

Project 0

Deployment

Databases

Data Services

Data API

SECURITY

Database Access

Network Access

Advanced

Atlas

Brain's Org - 2021-12-19 > PROJECT 0 > DATABASES

Cluster0

Overview

Real Time

Metrics

Collections

Profiler

Search

Performance Advisor

Online Archive

Command

REFRESH

Explore Your Data

_id	name	type	size
0			

VISUALIZE YOUR DATA

Add My Own Data

Load a Sample Dataset

REGION AWS N. Virginia (us-east-1)

VERSION 4.4.10

COMMAND

REFRESH

Learn more in Docs and Tutorials ↗

Crea a1.js y pega el código

The screenshot shows the Wawiwa web application interface. At the top, there's a navigation bar with Project 0, Overview, Real Time, Metrics, Collections (selected), Search, Profiler, Performance Advisor, Online Archive, and Command Line Tools. There are also refresh, export, and help icons.

In the main area, under the DEPLOYMENT tab, the Databases section is active. It shows 1 COLLECTION: test. A 'Create Database' button is available. Below it, the DATA SERVICES section includes Triggers, Data API (with a PREVIEW button), SECURITY (with a people section), Database Access, Network Access, and Advanced.

The Collections section shows 1 COLLECTION: test.people. It displays the following details:

- COLLECTION SIZE: 180B
- TOTAL DOCUMENTS: 1
- INDEXES TOTAL SIZE: 4kB
- Indexes
- Schema Anti-Patterns: 0
- Aggregation
- Search Indexes: 0

Below this, there's a search bar with 'test' and a 'Find' button. To the right, there are buttons for 'OPTIONS', 'Apply', and 'Reset'. An 'INSERT DOCUMENT' button is also present.

A query builder section titled 'test.people' contains the following code:

```
filter: { field: 'value' }
```

Buttons for 'OPTIONS', 'Apply', and 'Reset' are located to the right of the filter input. A large text area below shows the query results:

QUERY RESULTS 1-1 OF 1

```
_id:ObjectId("61be40c675231917402174")
> name: Object
> birth: 1912-06-22T21:39:28.000+00:00
> death: 1954-06-06T22:00:00.000+00:00
> contribs: Array
views: 1250000
```

At the bottom right, there's a status bar with 'System Status: All Good', '©2021 MongoDB, Inc.', and links for Status, Terms, Privacy, Alias Blog, and Contact Sales. A small circular icon with a speech bubble and a checkmark is also visible.

Crea a1.js y pega el código

```
FILTER { field: 'value' }
```

QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("61bef0c87525319a1740b174")
> name: Object
  birth: 1912-06-22T21:39:20.000+00:00
  death: 1954-06-06T22:00:00.000+00:00
> contribs: Array
  views: 1250000
```

_id único

Los campos de
“person”

Código final con try y catch

```
const { MongoClient } = require('mongodb');
const uri = "mongodb+srv://asaf@cluster0.hqfhv.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

client.connect(async (err) => {
  if (err) console.error(err);
  try {
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");

    // Construye un documento
    const personDocument = {
      "name": { "first": "Alan", "last": "Turing" },
      "birth": new Date(1912, 5, 23), // June 23, 1912
      "death": new Date(1954, 5, 7), // June 7, 1954
      "contribs": [ "Turing machine", "Turing test", "Turingery" ],
      "views": 1250000
    };

    const p = await col.insertOne(personDocument);
  } catch (err) { console.log(err.stack); }
  finally { await client.close(); }
});
```

Código final con try y catch

```
try {
    const db = client.db("test");

    // Usa la colección "people"
    const col = db.collection("people");

    // Construye un documento
    const personDocument = {
        "name": { "first": "Alan", "last": "Turing" },
        "birth": new Date(1912, 5, 23), // June 23, 1912
        "death": new Date(1954, 5, 7), // June 7, 1954
        "contribs": [ "Turing machine", "Turing test", "Turingry" ],
        "views": 1250000
    };

    const p = await col.insertOne(personDocument);

    // Encuentra un documento
    const myDoc = await col.findOne();

    // Imprime en la consola
    console.log(myDoc);
}
```

```
{
    _id: new ObjectId("61bef30d9126e644fe102cfb"),
    name: { first: 'Alan', last: 'Turing' },
    birth: '1912-06-22T21:39:20.000Z',
    death: '1954-06-06T22:00:00.000Z',
    contribs: [ 'Turing machine', 'Turing test', 'Turingry' ],
    views: 1250000
}
```

Insertemos 2 "persons"

```
const personDocument1 = {  
  "name": { "first": "Mike", "last": "Lewis" },  
  "birth": new Date(1942, 5, 22),  
  "death": new Date(1980, 6, 8),  
  "contribs": [ "Turingery" ],  
  "views": 130000  
};
```

```
const personDocument2 = {  
  "name": { "first": "Lisa", "last": "Mine" },  
  "birth": new Date(1971, 8, 22),  
  "death": new Date(1967, 9, 9),  
  "contribs": [ "Turing machine" ],  
  "views": 145000  
};
```

Después de insertar

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

- Database:** test
- Collection:** people
- Query:** `{ field: 'value' }`
- Results:** 3 documents found.

Document 1:

```
_id: ObjectId("61be130d9126e64fe02cfb")  
  name: Object  
    first: "Alan"  
    last: "Ullman"  
    birth: 1912-06-22T21:39:20.000+00:00  
    death: 1954-06-06T22:00:00.000+00:00  
    contribs: Array  
      views: 1250000
```

Document 2:

```
_id: ObjectId("61bf1133176f649e321250e0")  
  name: Object  
    first: "Mike"  
    last: "Lewis"  
    birth: 1942-06-21T21:00:00.000+00:00  
    death: 1989-07-07T22:00:00.000+00:00  
    contribs: Array  
      views: 130000
```

Document 3:

```
_id: ObjectId("61bf1145651d23d771b52c")  
  name: Object  
    first: "Lisa"  
    last: "Mile"  
    birth: 1971-09-21T22:00:00.000+00:00  
    death: 1987-10-08T22:00:00.000+00:00  
    contribs: Array  
      views: 145000
```

Insert

insertOne Insertar en la colección un documento

```
const p = await col.insertOne(personDocument);
```

insertOne vs insertMany

El método `insert()` está obsoleto en el controlador principal, por lo que deberías usar el método `.insertOne()` cuando quieras insertar un solo documento en tu colección y `.insertMany` cuando quieras insertar múltiples documentos en tu colección.

Más sobre `insertOne` e `insertMany`

`insertOne` - [link](#)

`insertMany` - [link](#)

Hazlo tú mismo 1

La secuencia disponible en las siguientes diapositivas (hasta el fin del capítulo) solamente va a funcionar si ejecutamos todos los pasos descritos en las diapositivas previas - conexión a la base de datos, try, catch, etc.

1. Abre una cuenta en mongoDB
2. Despues de registrarte, verifica que tienes un usuario en la lista de usuarios que tiene acceso a la cuenta
3. Verifica que tu ip está en la lista de ips y si no, agregala
4. Crea un archivo js llamado b1.js
5. Haz click en connect y luego conecta tu aplicación y selecciona nodejs versión 4. Copia el código y pegalo en b1.js.
- Corre el archivo de node b1

Hazlo tú mismo 2

Crea un objeto "product" con los siguientes atributos en el archivo `obj1.js`:

```
const product = {  
    "title": "ball",  
    "description": "Big blue ball",  
    "tags": ["circle", "toy", "kids"],  
    "age": 12,  
    "price": 20  
};
```

Agrega los comandos correctos para poner este "product" en mongoDB usando el comando "insert"

Corre el archivo `obj1.js` y asegurate que un "product" se registra

Hazlo tú mismo 3

Crea 3 objetos adicionales que describa "products" diferentes en el archivo b1.js

Agrega los comandos correctos para insertar en mongoDB usando el comando insert
Corre el archivo b1 y asegurate que los "products" se registran

find()

```
find({})
```

Retorna todos los documentos

```
try {
```

```
const db = client.db("test");
```

```
// Usa la colección "people"
```

```
const col = db.collection("people");
```

```
// Encuentra un documento
```

```
const myDoc = await col.find().toArray();
```

```
// Imprime en la consola
```

```
console.log(myDoc);
```

```
}
```

```
[ {  
    _id: new ObjectId("61bef30d9126664fe102cfb"),  
    name: { first: 'Alan', last: 'Turing' },  
    birth: 1912-06-23T21:39:20.000Z,  
    death: 1954-06-06T22:00:00.000Z,  
    contribs: [ 'Turing machine', 'Turing test', 'Turingery' ],  
    views: 1250000  
},  
{  
    _id: new ObjectId("61bf1133176f849e321250d0"),  
    name: { first: 'Mike', last: 'Lewis' },  
    birth: 1942-06-21T21:00:00.000Z,  
    death: 1980-07-07T22:00:00.000Z,  
    contribs: [ 'Turingery' ],  
    views: 130000  
},  
{  
    _id: new ObjectId("61bf1145651d237d271b528c"),  
    name: { first: 'Lisa', last: 'Hine' },  
    birth: 1971-09-21T22:00:00.000Z,  
    death: 1967-10-08T22:00:00.000Z,  
    contribs: [ 'Turing machine' ],  
    views: 145000  
}]
```

Hazlo tú mismo 4

Abre un archivo llamado `b4.js` y escribe un código adecuado para que imprima en la consola todos los registros en mongoDB
Ayúdate del comando:

```
find() .toArray();
```

Find y filter ({})

```
find({views:130000})  
Retorna el documento que sus "views" sean 130000 (Mike Lewis)  
  
try {  
  const db = client.db("test");  
  
  // Usa la colección "people"  
  const col = db.collection("people");  
  
  const myDoc = await col.find({ "views":130000 }).toArray();
```

```
// Encuentra un documento  
// Imprimir a la consola  
console.log(myDoc);
```

Solamente "people" 130,000
"views"

```
{  
  id: new ObjectId("61bf1133176f849e321250d0"),  
  name: { first: 'Mike', last: 'Lewis' },  
  birth: '1942-06-21T21:00:00.000Z',  
  death: '1980-07-07T22:00:00.000Z',  
  contribs: [ 'Turingery' ],  
  views: 130000  
}
```

Hazlo tú mismo 5

Abre un archivo llamado b5.js y escribe un código adecuado que imprima en la consola todos os registros que están en mongodb que su "cost" sea 20.

Ayudate del comando:

```
find() .toArray();
```

Operadores populares - enlaces para más operadores

<u><code>\$eq</code></u>	Coincide los valores que son iguales a un valor específico.
<u><code>\$gt</code></u>	Coincide los valores que son mayores a un valor específico.
<u><code>\$gte</code></u>	Coincide los valores que son mayores o igual a un valor específico.
<u><code>\$in</code></u>	Coincida cualquier valor de un arreglo específico.
<u><code>\$lt</code></u>	Coincide los valores que son menores a un valor específico.
<u><code>\$lte</code></u>	Coincide todos los valores que son menores o igual a un valor específico.
<u><code>\$ne</code></u>	Coincide con ninguno de los valores especificados en un arreglo

Encontrar y filtrar ({}) - \$gt

La siguiente operación usa el operador **\$gt** que retorna todos los documentos de la colección **people** donde **views** es mayor a 135000

```
try {  
    const db = client.db("test");  
  
    // Usa la colección "people"  
    const col = db.collection("people");  
  
    // Encuentra un documento  
    const myDoc = await col.find({ "views": { $gt: 135000 } }).  
    // Imprime en la consola  
    console.log(myDoc);  
}  
]  
  
[  
  {  
    id: new ObjectId("61bf30d9126e644fe102cfb"),  
    name: { first: "Alan", last: "Turing" },  
    birth: 1912-06-23T21:39:20.000Z,  
    death: 1954-06-06T22:00:00.000Z,  
    contribs: [ "turing machine", "turing test", "turingry" ],  
    views: 1250000  
  },  
  {  
    id: new ObjectId("61bf1145651d237d271b528c"),  
    name: { first: "Lisa", last: "Mine" },  
    birth: 1971-09-21T22:00:00.000Z,  
    death: 1967-10-08T22:00:00.000Z,  
    contribs: [ "turing machine" ],  
    views: 145000  
  }  
]
```

Hazlo tú mismo 6

Abre un archivo llamado b6.js y escribe un código adecuado que imprima a la consola todos los registros que están en mongoDB que su "cost" sea de más de 20.

Ayúdate del comando:

```
find() .toArray();
```

Encontrar y filtrar ({}) - \$lt

La siguiente operación usa el operador **\$lt** que retorna todos los documentos de la colección **people** donde **views** es menor a 135000

```
try {
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");

    // Encuentra un documento
    const myDoc = await col.find({ "views": { $lt: 135000 } }).toArray();

    // Imprime en la consola
    console.log(myDoc);
}
```

Hazlo tú mismo 7

Abre un archivo llamado b7.js y escribe un código adecuado que imprima a la consola todos los registros que están en mongodb que su "cost" sea de menos de 20.

Ayúdate del comando:

```
find() .toArray();
```

Encontrar y Filtrar ({}) - entre

Combina operadores de comparación para especificar específicos rangos para un campo. La siguiente operación retorna de la colección “people” los documentos donde las “views” son **entre** 100,000 y 130,000

```
try {  
    const db = client.db("test");  
  
    // Usa la colección "people"  
    const col = db.collection("people");  
  
    // Encuentra un documento  
    const myDoc = await col.find({ "views": { $gt:100000, $lt:150000 } }).toArray();  
  
    // Imprime en la consola  
    console.log(myDoc);  
}
```

mayor menor

Hazlo tú mismo 8

Abre un archivo llamado b8.js y escribe un código adecuado que imprima a la consola todos los registros que están en mongodb que su "cost" sea entre 20 y 40.

Ayúdate del comando:

```
find() .toArray();
```

Query de condiciones múltiples

La siguiente operación retorna todos los documentos de la colección “people” que el campo “views” es mayor a 100000 y el campo “birth” es mayor a 01/01/1920:

```
try {  
    const db = client.db("test");  
  
    // Usa la colección "people"  
    const col = db.collection("people");  
  
    // Encuentra un documento  
    const myDoc = await col.find({  
        "views": {$gt:100000},  
        "birth": {$gt: new Date('1920-01-01') }  
    }) .toArray();  
  
    // Imprime en la consola  
    console.log(myDoc);  
}
```

Múltiples condiciones



```
// Encuentra un documento  
const myDoc = await col.find(  
    "views": {$gt:100000},  
    "birth": {$gt: new Date('1920-01-01') }  
) .toArray();
```

```
// Imprime en la consola  
console.log(myDoc);
```

Hazlo tú mismo 9

Abre un archivo llamado b9.js y escribe un código adecuado que imprima a la consola todos los registros que están en mongodb que su "cost" mayores de 20 años y también son adecuados para edades de 12 años.

Ayúdate del comando:

```
find() .toArray();
```

Consultar documentos incorporados

La siguiente operación retorna documentos en la colección people donde el documento incorporado es exactamente { first: "Lisa" , last: "Mine" }, incluyendo el orden:

```
try {
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");

    // Encuentra un documento
    const myDoc = await col.find({ name: { first: "Lisa" , last: "Mine" }}).toArray();

    // Imprime en la consola
    console.log(myDoc);
}
```

Query incorporado

Campos de consulta de documentos incrustados

La siguiente operación devuelve documentos en la colección donde el “name” del documento incrustado contiene un campo primero con el valor “Lisa” y un campo al final con el valor “Mine”. La consulta utiliza **notación de puntos para acceder a los campos de un documento incorporado**:

```
try {  
    const db = client.db("test");  
  
    // Usa la colección "people"  
    const col = db.collection("people");  
  
    // Encuentra un documento  
    const myDoc = await col.find({  
        "name.first": "Lisa",  
        "name.last": "Mine"  
    }).toArray();  
  
    // Imprime en la consola  
    console.log(myDoc);  
}
```

Consulta arreglos

La siguiente operación retorna documentos en la colección “people” donde el campo “contribs” contiene el elemento **Turing test**:

```
try {
  const db = client.db("test");
  [
    {
      _id: new ObjectId("61bef3009126e644fe102cfb"),
      name: { first: 'Alan', last: 'Turing' },
      birth: '1912-06-23T21:39:20.000Z',
      death: '1954-06-06T22:00:00.000Z',
      contribs: [ 'Turing machine', 'Turing test', 'Turingery' ],
      views: 1230000
    }
  ]
}

// Encuentra un documento
const myDoc = await col.find(
  { "contribs": "Turing test" }).toArray();

// Imprime en la consola
console.log(myDoc);
}
```

Consulta arreglos

La siguiente operación retorna documentos en la colección “people” donde el campo del arreglo de “contribs” contiene el elemento “Turingery” or “Lisp”:

```
try {
    const db = client.db("test");
    const col = db.collection("people");
    // Usa la colección "people"
    // Encuentra un documento
    const myDoc = await col.find(
        {"contribs": "Turingery"}).toArray();
    // Imprime en la consola
    console.log(myDoc);
}
```

```
[{"_id": new ObjectId("61bef30dd9126e644fe102cfb"),
  "name": {"first": "Alan", "last": "Turing"},
  "birth": "1912-06-23T21:39:20.000Z",
  "death": "1954-06-07T22:00:00.000Z",
  "contribs": ["Turing machine", "Turing test", "Turingery"],
  "views": 1250000},
 {"_id": new ObjectId("61bf1133176f849e32125000"),
  "name": {"first": "Mike", "last": "Lewis"},
  "birth": "1942-06-23T21:00:00.000Z",
  "death": "1980-07-07T22:00:00.000Z",
  "contribs": ["Turingery"],
  "views": 130000}]
```

Consulta arreglos

La siguiente operación usa el operador `$all` para retornar los documentos en la colección donde el campo de arreglo “`contribs`” contiene los dos elementos “`Turingery`” y “`Turing test`”:

```
try
{
  const db = client.db("test");
  // Usa la colección "people"
  const col = db.collection("people");
  // Encuentra un documento
  const myDoc = await col.find(
    { contribs: [ "$all: [ "Turingery" , "Turing test" ] } ).toArray();
  console.log(myDoc);
}
```

Consulta arreglos

La siguiente operación usa el operador `$size` para retornar documentos en la colección “bios” donde el tamaño del arreglo “contribs” es 3:

```
try
{
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");
    // Encuentra un documento
    const myDoc = await col.find(
        { contribs: { $size: 3} }).toArray();
    // Imprime en la consola
    console.log(myDoc);
}
```

Sort

El método sort() ordena los documentos en un conjunto del resultado.

La siguiente operación retorna documentos en la colección “people” ordenados de forma ascendente por el campo name.first:

ascending (1) vs descending (-1) order

```
[ {  
    _id: new ObjectId("61bef38d9126e644fe102cfb"),  
    name: { first: "Alan", last: "Turing" },  
    birth: "1912-06-23T21:39:28.000Z",  
    death: "1954-06-07T22:08:08.000Z",  
    contribs: [ "Turing machine", "Turing test", "Turing theory" ],  
    views: 1250000  
,  
    _id: new ObjectId("61bf114565ad37d71b528c"),  
    name: { first: "Lisa", last: "Mine" },  
    birth: "1971-09-20T22:08:08.000Z",  
    death: "1967-10-06T22:08:08.000Z",  
    contribs: [ "Turing machine" ],  
    views: 150000  
,  
    _id: new ObjectId("61bf113317f649e32125000"),  
    name: { first: "Mike", last: "Lewis" },  
    birth: "1942-06-21T21:08:08.000Z",  
    death: "1980-07-07T22:08:08.000Z",  
    contribs: [ "Turing theory" ],  
    views: 130000  
}  
,
```

```
try {  
    const db = client.db("test");  
    // Usa la colección "people"  
    const col = db.collection("people");  
    // Encuentra un documento  
    const myDoc = await col.find().sort({ 'name.first': 1 }).toArray();  
    // Imprime en la consola  
    console.log(myDoc);  
}  
}
```

Limit

El método `limit()` limita el número de documentos en el conjunto de resultados. La siguiente operación retorna máximo 2 documentos en la colección:

```
try {
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");
    // Encuentra un documento
    const myDoc = await col.find().limit(2).toArray();
    // Imprime en la consola
    console.log(myDoc);
}
```

```
[{"id": new ObjectId("61bef30d9126e64fe102cfb"),
  name: { first: 'Alan', last: 'Turing' },
  birth: "1912-06-23T21:39:20.000Z",
  death: "1954-06-06T22:00:00.000Z",
  contribs: [ 'Turing machine', 'Turing test', 'Turingery' ],
  views: 1250000},
 {"id": new ObjectId("61bf1133176f849e321250d0"),
  name: { first: 'Mike', last: 'Lewis' },
  birth: "1942-06-21T21:00:00.000Z",
  death: "1980-07-07T22:00:00.000Z",
  contribs: [ 'Turingery' ],
  views: 130000}]
```

Hazlo tú mismo 10

Abre un archivo llamado 10.js y escribe un código adecuado que imprima a la consola todos los registros, limita el número de registros a 2.

Ayúdate del comando:

```
find() .toArray();
```

Delete - enlace

El método `deleteOne()`:

```
try {
```

```
    const db = client.db("test");
    // Usa la colección "people"
    const col = db.collection("people");
    // Encuentra un documento
    const myDoc = await
        col.deleteOne({ _id:ObjectId("61bef30d9126e644fe102cfb")});
    // Imprime en la consola
    console.log(myDoc);
    await client.close();
}
```

Antes

QUERY RESULTS 1-3 OF 3

```
1: {_id: ObjectId("61bef30d9126e644fe102cfb")}
2: { name: Object
3:   birth: 1912-06-22T23:31:20.000+00:00
4:   death: 1954-06-05T22:00:00.000+00:00
5:   contribs: Array[0]
6:   views: 1250000 }
```

```
1: {_id: ObjectId("61bf1331f6f649a31125405f")}
2: { name: Object
3:   birth: 1942-06-21T21:00:00.000+00:00
4:   death: 1988-07-07T22:00:00.000+00:00
5:   contribs: Array[0]
6:   views: 130000 }
```

```
1: {_id: ObjectId("61bf1331f6f649a31125405f")}
2: { name: Object
3:   birth: 1971-09-21T22:00:00.000+00:00
4:   death: 1987-10-01T22:00:00.000+00:00
5:   contribs: Array[0]
6:   views: 145000 }
```

delete

After

QUERY RESULTS 1-2 OF 2

```
_id: ObjectId('61bf1133176f849e321250c0')
```

```
> name: Object  
> birth: 1942-06-21T21:00:00.000+00:00  
> death: 1988-07-07T22:00:00.000+00:00  
> contribs: Array  
views: 130000
```

```
_id: ObjectId('61bf1145651d2373721b528c')
```

```
> name: Object  
> birth: 1971-09-21T22:00:00.000+00:00  
> death: 1967-10-08T22:00:00.000+00:00  
> contribs: Array  
views: 145000
```

```
try {  
    const db = client.db("test");  
    // Usa la colección "people"  
    const col = db.collection("people");  
    // Encuentra un documento  
    const myDoc = await col.deleteOne({ _id:ObjectId("61bef30d9126e644fe102cf0") })  
    // Imprime en la consola  
    console.log(myDoc);  
    await client.close();  
}
```

Update síncrono

antes

```
_id: ObjectId("61bf1133176f849e321250d0")
  > name: Object
    birth: 1942-06-21T21:00:00.000+00:00
    death: 1980-07-07T22:00:00.000+00:00
    contribs: Array
      views: 130000
```

El método update:

```
try
```

```
{  
  const db = client.db("test");  
  // Usa la colección "people"  
  const col = db.collection("people");  
  // Encuentra un documento  
  const myquery = { "views": 130000 };  
  const newvalues = { $set: { views: 172123 } }  
  await col.updateOne(myquery, newvalues)  
  // Imprime en la consola  
  console.log("updated");  
  await client.close();  
}
```

después

```
_id: ObjectId("61bf1145651d237d271b528c")
  > name: Object
    birth: 1971-09-21T22:00:00.000+00:00
    death: 1967-10-08T22:00:00.000+00:00
    > contribs: Array
      views: 145000
```

```
_id: ObjectId("61bf1133176f849e321250d0")
  > name: Object
    birth: 1942-06-21T21:00:00.000+00:00
    death: 1980-07-07T22:00:00.000+00:00
    > contribs: Array
      views: 173123
```

Funcióñ Async - Recordatorio de js - enlace

Una función `async` es una función declarada con las claves “`async`” y la clave “`await`” está permitido en ella. Las claves “`async`” y “`await`” permite comportamiento asíncrono a base de promesas en una manera más limpia, evitando la necesidad de configurar cadenas de promesas.

Update asíncrono

before

```

client.connect(err => {
  if(err) console.error(err);

  const db = client.db("test");

  // Usa la colección "people"
  const col = db.collection("people");

  // Encuentra un documento
  const myQuery = { "views": 130000 };
  const newvalues = { $set: { views: 172123 } };

  col.updateOne(myQuery, newvalues, (err, res) => {
    if (err) throw err;
    console.log("1 document updated");
    client.close();
  });
}

// Imprime en la consola
};


```

After

```

_id:ObjectId("61bf1133176f849e321250d0")
  > name: Object
    birth: 1942-06-21T21:00:00.000+00:00
    death: 1980-07-07T22:00:00.000+00:00
    > contribs: Array
      views: 130000

_id:ObjectId("61bf1145651d237d271b528c")
  > name: Object
    birth: 1967-09-21T22:00:00.000+00:00
    death: 1987-10-08T22:00:00.000+00:00
    > contribs: Array
      views: 145000

_id:ObjectId("61bf1133176f849e321250d0")
  > name: Object
    birth: 1942-06-21T21:00:00.000+00:00
    death: 1980-07-07T22:00:00.000+00:00
    > contribs: Array
      views: 172123

_id:ObjectId("61bf1145651d237d271b528c")
  > name: Object
    birth: 1971-09-21T22:00:00.000+00:00
    death: 1987-10-08T22:00:00.000+00:00
    > contribs: Array
      views: 145000

```

Hazlo tú mismo 11

Abre un archivo llamado b11.js y escribe un código adecuado que actualice los registros que su "cost" sea de 20 a 30 .
Ayúdate de with updateMany or updateOne

wawiwa

¿Preguntas?