educación IT
CODE YOUR FUTURE

Bootcamp
Full Stack Engineer

Fase 3 - Desarrollo Back End

Módulo 35

Operaciones CRUD en MongoDB

¿Qué es CRUD?

El concepto *CRUD* está estrechamente vinculado a la **gestión de datos digitales**. CRUD hace referencia a un acrónimo en el que se reúnen las primeras letras de **las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos**:

- **Create**: crear registros.
- **Read** / **Retrieve**: leer registros.
- **Update**: actualizar registros.
- **Delete** / **Destroy**: borrar registros.

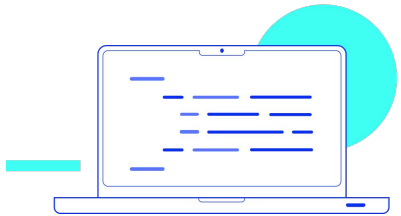
CRUD resume las funciones requeridas por un usuario para **crear** y **gestionar datos**. Varios procesos de gestión de datos están basados en en este sistema donde las operaciones están adaptadas a **los requisitos del sistema y del usuario**, tanto para **la gestión de bases de datos** como para **el uso de aplicaciones**.

Para un experto, las operaciones son las herramientas de acceso claves para comprobar, por ejemplo, **los problemas de la base de datos**. En cambio, para un usuario, CRUD significa **crear una cuenta** (*create*), **usarla** (*read*), **actualizarla** (*update*) o **borrarla** (*delete*).

En MongoDB, las operaciones CRUD se implementan a través de comandos específicos que, en nuestro caso, ejecutaremos a través de la consola **Mongo Shell**.

Iniciando el entorno

1. Para testear Mongo, levantaremos el servidor de base de datos, ejecutando **`Mongod`** en la consola.
2. Luego, **en otra ventana**, ejecutamos **el cliente Mongo Shell** con **Mongo** y verificamos que la conexión se haya realizado **correctamente** (ambas consolas, la del servidor y la del cliente **deben quedar bloqueadas**, a la espera de comandos).
3. Los comandos los escribiremos en la consola de **Mongo Shell**.



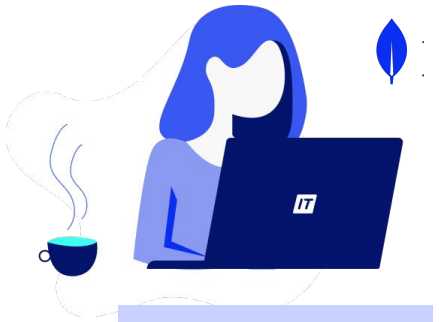
Creando los primeros documentos

Mongo es un gestor de bases de datos no relacionales, estas están compuestas por **colecciones**, que a su vez están integradas por **documentos**.

Vamos a crear una base de datos llamada **“Test”** y una colección ***Inventory*** de documentos.

Para crear una base de datos nueva o cambiar a una existente, usamos el comando **`Use`**. De esta forma, para crear una base de datos llamada **“Test”,** escribiremos en la consola de Mongo **`use test`**.

En la diapositiva siguiente, veremos la imagen de la consola de Mongo.



```
switched to db test 1
> show databases
admin   0.000GB
config  0.000GB      2
local   0.000GB
> db.inventory.insertMany([ 3
...     // MongoDB adds the _id field with an ObjectId if _id is not present
...     { item: "journal", qty: 25, status: "A",
...         size: { h: 14, w: 21, uom: "cm" }, tags: [ "blank", "red" ] },
...     { item: "notebook", qty: 50, status: "A",
...         size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank" ] },
...     { item: "paper", qty: 100, status: "D",
...         size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank", "plain" ] },
...     { item: "planner", qty: 75, status: "D",
...         size: { h: 22.85, w: 30, uom: "cm" }, tags: [ "blank", "red" ] },
...     { item: "postcard", qty: 45, status: "A",
...         size: { h: 10, w: 15.25, uom: "cm" }, tags: [ "blue" ] }
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5c5071283f08894e4d46a59d"),
                ObjectId("5c5071283f08894e4d46a59e"),
                ObjectId("5c5071283f08894e4d46a59f"),      4
                ObjectId("5c5071283f08894e4d46a5a0"),
                ObjectId("5c5071283f08894e4d46a5a1")
        ]
}
> show collections 5
inventory
> show databases  6
admin   0.000GB
config  0.000GB
local   0.000GB
test    0.000GB
```

Con la imagen anterior, extraer las siguientes conclusiones:

1. El comando para **cambiar de base de datos** y para **crear una nueva** es **`Use`**.
2. Para **mostrar las bases de datos** usamos el comando **`Show`** seguido por el subcomando **`Databases`**. No se muestran las bases de datos que **no tengan colecciones en su interior**, como la que acabamos de crear.
3. Para insertar un conjunto de documentos en una colección **no es necesario crear previamente la colección** (aunque también podríamos crear la colección vacía).
4. Tras el comando ejecutado con éxito,veremos **un resumen de los documentos creados y el id asignado por Mongo a cada uno de ellos**.
5. Si queremos ver las colecciones dentro de la base de datos que estamos usando, utilizaremos el comando **`Show`** seguido del subcomando **`Collections`**.
6. Al tener una colección, la base de datos creada inicialmente se nos muestra **al ejecutar el comando correspondiente**.

> El script inicial para insertar documentos está sacado de la documentación oficial de Mongo: Getting Started — MongoDB Manual.

Operaciones CRUD

CRUD es el acrónimo de las palabras inglesas ***Create***, ***Read***, ***Update*** y ***Delete***.

A continuación, vamos a ver la sintaxis básica de cada una de estas operaciones en Mongo.

Create

Podemos usar el procedimiento **`insertMany()`** en la creación de **colecciones y documentos**. A continuación, veremos qué debemos hacer si deseamos **insertar solo un documento y guardar un documento en una variable e introducirla más adelante en la colección**.

Los documentos de Mongo siguen una estructura determinada influenciada por JSON, donde un documento es visto como una sucesión de pares **clave-valor**. Mientras que el valor puede ser **cualquier tipo de datos BSON**, incluso otros documentos o *arrays*, todas las claves son de tipo ***String***.

Veamos un ejemplo de introducción de un único documento. Como Mongo permite **registros / documentos duplicados** (le asigna a cada uno un `ID` distinto internamente, el cual actúa como una clave primaria), usaremos el primer documento introducido en el paso anterior para ejemplificar el proceso.

En la diapositiva siguiente, veremos el ejemplo.

Notemos cómo podemos crear **una variable a la que asignarle un documento** que introduciremos luego en una colección (pese a previamente haberlo introducido). Mongo se encargará de asignarle un **`ObjectId`** distinto a cada uno.

```
> var document1 = { item: "journal", qty: 25, status: "A",
...         size: { h: 14, w: 21, uom: "cm" }, tags: [ "blank", "red" ] }1
> db.inventory.insertOne(document1);2
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5c5079103f08894e4d46a5a2")
}
> db.inventory.find();
{ "_id" : ObjectId("5c5071283f08894e4d46a59d"), "item" : "journal", "qty" : 25, "status" : "A", "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "tags" : [ "blank", "red" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a59e"), "item" : "notebook", "qty" : 50, "status" : "A", "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "tags" : [ "red", "blank" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a59f"), "item" : "paper", "qty" : 100, "status" : "D", "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "tags" : [ "red", "blank", "plain" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a5a0"), "item" : "planner", "qty" : 75, "status" : "D", "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "tags" : [ "blank", "red" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a5a1"), "item" : "postcard", "qty" : 45, "status" : "A", "size" : { "h" : 10, "w" : 15.25, "uom" : "cm" }, "tags" : [ "blue" ] }
{ "_id" : ObjectId("5c5079103f08894e4d46a5a2"), "item" : "journal", "qty" : 25, "status" : "A", "size" : { "h" : 14, "w" : 21, "uom" : "cm" }, "tags" : [ "blank", "red" ] }
```

MongoDB.

Read

El comando **`find` devuelve los documentos en función de determinadas condiciones** y los muestra, si así se desea, de distinta manera. Por ejemplo, oculta algún par clave-valor.

La sintaxis del comando es:

```
db.<collection>.find(filter, projection);
```

Tanto **`filter`** como **`projection`** son parámetros **opcionales**. **`Filter`** es **el filtro por el que pasarán los documentos de la colección**. Un ejemplo podría ser: “muéstrame los documentos que tengan como valor de la clave «qty» el número 25”.

`Projection` muestra u oculta los pares clave-valor del documento. Si bien por defecto muestra todos, podemos usarlo para filtrar datos a través de una sentencia como la anterior: “muéstrame solo los *tags* de los documentos que tengan como valor de la clave «qty» el número 25”.

Podríamos resumir que el **query** es el **`where`** y el **`projection`** el **select** de una consulta **MySQL**.

Veamos el ejemplo en la diapositiva siguiente.

```
> db.inventory.find({"qty":25});1
{ "_id" : ObjectId("5c5071283f08894e4d46a59d"), "item" : "journal", "qty" : 25, "status" .....
{ "_id" : ObjectId("5c5079103f08894e4d46a5a2"), "item" : "journal", "qty" : 25, "status" .....
> db.inventory.find({"qty":25},{tags:1,_id:0});2
{ "tags" : [ "blank", "red" ] }
{ "tags" : [ "blank", "red" ] }
> db.inventory.find({},{tags:1,_id:0});3
{ "tags" : [ "blank", "red" ] }
{ "tags" : [ "red", "blank" ] }
{ "tags" : [ "red", "blank", "plain" ] }
{ "tags" : [ "blank", "red" ] }
{ "tags" : [ "blue" ] }
{ "tags" : [ "blank", "red" ] }
> _
```

Consideraciones

Los parámetros de las funciones de Mongo se colocan **entre llaves** y **separados por comas**. Si deseamos obviar un parámetro alcanza con poner **dos llaves que no encierran nada**.

- Seleccionamos los documentos con la clave **`qty`** igual a **25**.
- Seleccionamos los documentos con **`qty`** igual a **25** y mostramos **solo sus tags**. Como por defecto aparece el ID de cada documento, tenemos que explicitar que no deseamos mostrarlo.
- Si **no queremos filtrar** pero sí mostrar **los tags de los documentos**, dejamos el primer parámetro **vacío**.
- Podemos ubicar las llaves tanto **entre comillas como sin comillas**.

Por otro lado, veamos qué sucede si precisamos agregar **una condición de filtro más compleja**.

Comencemos por una condición de filtro que indique **mayor o igual que un valor**. Antes, colocábamos dos puntos y el valor que deseábamos filtrar, pero ahora, ese valor deja de ser único e implica la creación de una condición más compleja que un ***equals***.

Debemos encapsular **entre llaves** las condiciones más complejas, como un objeto cuya clave **es el operador** y su valor **el de la condición**.

La sintaxis sería:

```
db.<collection>.find({key:{$operator:value}}, projection);
```

En la documentación oficial: https://docs.mongodb.com/manual/reference/operator/query/

podemos conocer los operadores usados para **filtrar y proyectar/seleccionar**.

Cómo añadir condiciones de filtro

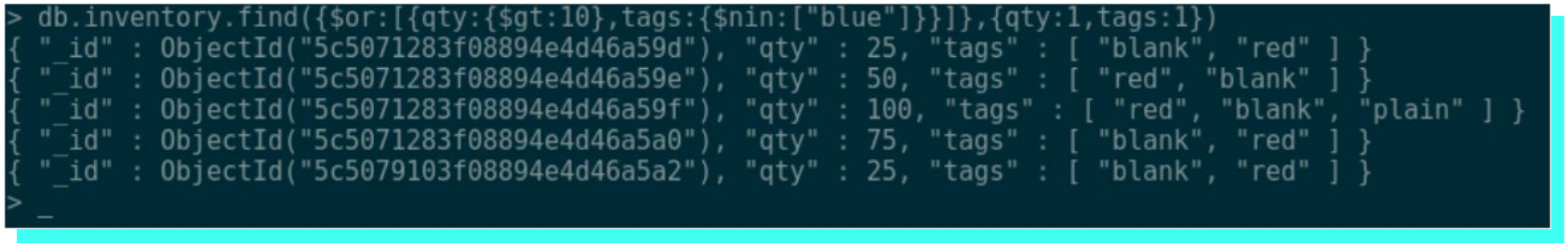
Imaginemos que deseamos, por ejemplo, que el campo **qty** sea mayor a 10 o no tenga el *tag «blue»*.

El parámetro ***query*** se puede expresar también como un objeto que tiene como clave un operador lógico y, como valor, un array de objetos que contienen los campos y sus condiciones.

Se auto explica con la sintaxis y un ejemplo:

```
db.<collection>.find({$logical_op:[{key:{$operator
:value},key:{$operator:value}}]}, projection);
```

Veamos la imagen con el ejemplo, debajo:



En la *shell* de Mongo resulta complicado ejecutar consultas complejas, sobre todo las primeras veces debido a lo poco legible que resulta el formato JSON en una única línea.

Si usamos la tecla ***Enter*** podemos comenzar a tener un código más legible.

```
> db.inventory.find(
...    {$or:
...      [
...      {qty:{$gt:10}},
...      {tags:{$nin:["blue"]}}
...      ]
...    },
...    {qty:1,tags:1})
{ "_id" : ObjectId("5c5071283f08894e4d46a59d"), "qty" : 25, "tags" : [ "blank", "red" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a59e"), "qty" : 50, "tags" : [ "red", "blank" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a59f"), "qty" : 100, "tags" : [ "red", "blank", "plain" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a5a0"), "qty" : 75, "tags" : [ "blank", "red" ] }
{ "_id" : ObjectId("5c5071283f08894e4d46a5a1"), "qty" : 45, "tags" : [ "blue" ] }
{ "_id" : ObjectId("5c5079103f08894e4d46a5a2"), "qty" : 25, "tags" : [ "blank", "red" ] }
```

Nota: el operador lógico **and** se puede obviar al ser el operador por defecto. Simplemente al poner en el parámetro *query* **el conjunto de campos y los valores por los que filtrarlos**, obtenemos el mismo resultado que haciendo explícito al operador **and**.

Estas dos expresiones son equivalentes:

```
db.inventory.find( { $and: [ {qty:{$gt:10}}, {tags:{$in:[«blue»]}}
]});
```

```
db.inventory.find( {qty:{$gt:10},tags:{$in:[«blue»]}});
```

Update

Al momento de actualizar documentos, utilizamos **`updateMany()`**, **`updateOne()`** o **`replaceOne()`**.

La sintaxis de todos ellos es la siguiente:

```
db.<collection>.replaceOne(<filter>, <update>,
<options>);
```

- El parámetro **`filter`** funciona como en la *query*, filtrando los documentos en función de su **campos y valores**.
- El parámetro **`update`** es el encargado de cambiar el valor de una serie de campos por otros valores.

- El parámetro **`options`** es una serie de pares clave-valor en un objeto con propósitos muy diversos en los que, por el momento, no ahondaremos.

Para ejecutar cualquiera de estas acciones, debemos pasarles como **primer parámetro** un filtro de documentos que ya aprendimos a realizar en el apartado correspondiente a la consulta de documentos. **El segundo parámetro** (encargado de la **actualización**) es el más interesante en
este caso.

Podemos ver todos los **operadores de actualización** con una breve descripción en este enlace de la documentación oficial: Update Operators MongoDB.

Veamos un ejemplo:

Si deseamos actualizar un documento, al quitar el campo **tags** y establecer el campo **cantidad** a 0 en todos los documentos con una cantidad inferior a 30 unidades, podríamos escribir este comando en consola:

```
db.inventory.updateMany({qty:{$lte:30}},{$unset:{tags:»»}, $set:{qty:0}})
```

Delete

En caso de necesitar suprimir un documento en una colección, de la misma manera que al momento de crearlos, contamos con dos **procedimientos** según la cantidad de documentos que queramos eliminar:

- Para un documento, **`deleteOne()`**:

```
db.collection.deleteOne(filter, option)
```

- Para varios documentos, **`deleteMany()`**:

```
db.collection.deleteMany(filter, option)
```

El procedimiento tiene dos parámetros:

- El primero es el parámetro **`filter`** que ya hemos visto en los dos tipos de operaciones anteriores.
- En el segundo parámetro, en función de los documentos filtrados, podemos especificar **una serie de opciones para el procedimiento**, siendo este, nuevamente opcional.

No vamos a incidir más por tanto en este procedimiento ya que ya hemos visto cómo funciona el parámetro **`filter`**.

En este link de la documentación oficial tendremos más detalle de estos comandos [Delete Documents - MongoDB Manual](#).

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!