EMILIO SÁNCHEZ
23-11-20

TEORÍA PROYECTO

➢ .find()

El find() se utiliza para hacer búsquedas en JavaScript para las bases de datos que estamos tratando. Dentro del find() se deberá colocar unos corchetes "{}", y dentro de ellos todas las búsquedas que queramos hacer. Si lo dejamos en blanco nos devolverá todos los documentos de la base de datos.

Se utiliza de la siguiente forma:

db.coleccion.find({<query>})

Donde **db** especifica que se trata de una base de datos (que debemos seleccionar previamente en el terminal) y **colección** es donde irá la colección de nuestra base de datos en la que trabajaremos.

➢ $eq

La cláusula $eq se utiliza para igualar un campo a un valor. Dentro del find se utiliza de esta manera:

<campo>: {$eq: <valor>}

➢ $gt, $gte, $lt, $lte

Las cláusulas $gt, $gte, $lt y $lte sirven para hacer búsquedas en cuanto a cantidades, mayores, menores o iguales. Los significados de cada una son:

- **$gt:** mayor que (greater than)
- **$gte:** mayor o igual que (greater than equal)
- **$lt:** menor que (less than)
- **$lte:** menor o igual que (greater than equal)

Se utilizan los cuatro de la siguienta manera:

<campo>: {$gt: <valor>}

TEORÍA PROYECTO

➢ $in, $nin

La cláusula $in se utiliza para incluir varios valores en una misma búsqueda. De esta forma, todo lo que vaya dentro de un array posterior al $in irá incluido en el campo:

<campo>: {$in: [<valor1>, <valor2>,…]}

La cláusula $nin es la opuesta a $in. Se escribe igual que $in pero sustituyendo $in por $nin, usándose igualmente un array. La búsqueda seleccionará todos los documentos que no contengan los valores del array.

➢ $ne

La cláusula $ne es contraria a $eq. Se buscarán todos los documentos cuyo valor del campo no sea igual al escrito tras el $ne. Se utiliza similar al $eq:

<campo>: {$ne: <valor>}

➢ $not

La cláusula $not se utiliza para seleccionar los documentos que no tengan el valor escrito en el campo seleccionado. Se escribe de la siguiente manera:

<campo>: {$not: <valor>}

Dicho <valor> puede contener varias cláusulas dentro, no tiene porque ser únicamente un valor.

➢ $regex

La cláusula $regex es un tanto diferente a las vistas anteriormente. Se utiliza para incluir una secuencia de caracteres de uno o más valores. Se utiliza de la siguiente forma:

<campo>: {$regex: /<secuencia>/}

De esta forma, si se sustituye la <secuencia> por una letra como la “m”, buscará todas las palabras que contengan una m. Se pueden añadir secuencias más largas, como por ejemplo “mar”, buscando entonces todas los documentos que tengan esa secuencia.

TEORÍA PROYECTO

➢ $and

La cláusula $and se utiliza antes de colocar cada campo del que haremos la selección. Usando el $and especificaremos que pondremos varias búsquedas dentro y que todas deben cumplirse.
Si no se utilizase, con poner una coma entre cada cláusula funcionaría como un $and, pero debido a ciertos problemas de incongruencia al usar distintas cláusulas para un mismo campo, es necesario usar el $and para especificarlo.

Se utiliza de la siguiente manera:

.find({$and: [{<query1>}, {<query2>}] })

➢ $or, $nor

La cláusula $or se utiliza igual que la cláusula $and, pero en vez de poner una serie de búsquedas que deben cumplirse, pone varias búsquedas, pero con que se cumpla una se mostrará ese documento. Funciona de forma semejante a un "o" (**o** una cosa **o** la otra **o** las dos).
Se escribe de forma similar:

.find({$or: [{<query1>}, {<query2>}] })

➢ .pretty(), .count()

El pretty() y el count() se utilizan de forma semejante al find. Mientras que en el find() se escriben todas las cláusulas de la búsqueda que vamos a hacer, pretty y count especifican como queremos ver esos datos.

Con pretty() podremos ver los datos de forma ordenado, ya que de predeterminadamente se nos muestran todos en una misma fila. Con pretty() separará cada documento y cada valor.
Con count() contará el número de documentos que nos devuelvo. No nos mostrará cada documento, sino un número tan solo.