



# TALLER DE CREACIÓN DE UNA RED SOCIAL CON REACT, REDUX, NODEJS, EXPRESS Y MONGODB

Danny Vaca

## Índice

1.	Objetivo general.....	2
2.	Objetivo específico.....	2
3.	Desarrollo.....	2
3.1.	Eliminar un registro .....	2
3.2.	Eliminar múltiples registros .....	3
3.3.	Actualizar un registro.....	4
3.4.	Actualizar múltiples registros .....	5
3.4.1.	Agregar campos a un documento .....	5
3.4.2.	Eliminar campos de un documento .....	6

## 1. Objetivo general

Actualizar y eliminar registros de la base de datos

## 2. Objetivo específico

- Realizar una conexión desde NodeJS hasta mongo
- Eliminar registros de la base de datos

## 3. Desarrollo

La presente práctica se desarrollará a partir de la práctica 2.

### 3.1. Eliminar un registro

Similar a la función **findOne** la cual nos devolvía el primer registro que coincidía con los parámetros de búsqueda tenemos la función **deleteOne** con la gran diferencia que esta función borra el primer registro que encuentra.

De forma idéntica a la función **findOne**, **deleteOne** recibe como primer argumento el objeto de parámetros de búsqueda y como segundo argumento la función callback que se ejecutará cuando primero documento encontrado sea borrado.

```
JS index.js x
1  const mongodb = require('mongodb')
2  const MongoClient = mongodb.MongoClient
3  const url = 'mongodb://localhost:27017/testdatabase'
4
5  MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
6    if (error) throw error
7    const database = connection.db()
8    const conditions = {name: 'Lionel Messi'}
9    database.collection('players').deleteOne(conditions, function(error, commandResult) {
10     if (error) throw error
11     database.collection('players').find(conditions).toArray(function(error, players) {
12       if (error) throw error
13       console.log(players)
14       connection.close()
15     })
16   })
17 })
18
```

Dentro del resultado de la función **deleteOne** hemos ubicado un **find** para buscar el elemento que se supone ya debería estar eliminado, y como resultado obviamente tenemos un array vacío ya que no existe un elemento que cumpla con esas condiciones.

```
PS C:\Users\dvaca\Downloads\mongo\practica3> node .\index.js
[]
PS C:\Users\dvaca\Downloads\mongo\practica3>
```

La función **deleteOne** borrará el primer registro que coincida con los parámetros, por lo tanto, si existe más de un registro la función **find** aún podría retornar datos, pese a que la función **deleteOne** se haya ejecutado correctamente.

### 3.2. Eliminar múltiples registros

La función que se usa para eliminar múltiples documentos es la función **deleteMany** la cuál funciona exactamente igual que la función **deleteOne** recibiendo como primer argumento las condiciones y como segundo argumento el callback a ejecutarse cuando los registros sean eliminados.

Para este caso la condición establecida es borrar todos los documentos cuyo **age** sea menor que **30**, usando el comparador **\$lt** dentro de las condiciones.



Dentro del callback de la función de eliminar igualmente hemos creado un **find** para ver los resultados, en los cuales no debe haber un solo registro mayor de 30 años.

```
JS delete.js x JS update.js
1  const mongodb = require('mongodb')
2  const MongoClient = mongodb.MongoClient
3  const url = 'mongodb://localhost:27017/testdatabase'
4
5  MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
6    if (error) throw error
7    const database = connection.db()
8    const conditions = {age:{$lt: 30}}
9    database.collection('players').deleteMany(conditions, function(error, commandResult) {
10     if (error) throw error
11     database.collection('players').find({}).toArray(function(error, players) {
12       if (error) throw error
13       console.log(players)
14       connection.close()
15     })
16   })
17 })
18
```

Como se puede apreciar en el resultado de la función **find** no hay un solo documento que tenga **age** menor que **30**

```
PS C:\Users\dvaca\Downloads\mongo\practica3> node .\delete.js
[ { _id: 5be072ad84a05829a4a8551e,
  name: 'Cristiano Ronaldo',
  country: 'Portugal',
  age: 33 },
  { _id: 5be072ad84a05829a4a85523,
  name: 'Edison Cavani',
  country: 'Uruguay',
  age: 32 },
  { _id: 5be072ad84a05829a4a85524,
  name: 'Luis Suarez',
  country: 'Uruguay',
  age: 31 },
  { _id: 5be072ad84a05829a4a85529,
  name: 'Luka Modric',
  country: 'Croacia',
  age: 31 } ]
PS C:\Users\dvaca\Downloads\mongo\practica3>
```

### 3.3. Actualizar un registro

Para actualizar un documento, tenemos la función **updateOne**, la cual recibe como primer argumento las condiciones para realizar la búsqueda, como segundo argumento los cambios a

Elaborado por:  
Ing. Danny Vaca

Fecha de revisión:  
5 noviembre del 2018



aplicar al documento, ya que mongo no solo nos permite realizar actualizaciones de un solo valor por uno nuevo, sino nos permite ejecutar diferentes acciones sobre el documento y como tercer valor se recibe el callback a ejecutarse cuando se haya realizado la acción de actualización del documento.

```
JS delete.js JS update.js x
1  const mongodb = require('mongodb')
2  const MongoClient = mongodb.MongoClient
3  const url = 'mongodb://localhost:27017/testdatabase'
4
5  MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
6    if (error) throw error
7    const database = connection.db()
8    const conditions = {name: 'Cristiano Ronaldo'}
9    const newValues = {$set:{name: 'CR7', balondor: 5}}
10   database.collection('players').updateOne(conditions, newValues, function(error, commandResult) {
11     if (error) throw error
12     database.collection('players').findOne(conditions, function(error, players) {
13       if (error) throw error
14       console.log(players)
15       connection.close()
16     })
17   })
18 })
19 |
```

En el interior del callback del **updateOne** se ha ubicado un **findOne** con el cual se espera ver que los cambios se hayan realizado correctamente.

```
PS C:\Users\dvaca\Downloads\mongo\practica3> node .\update.js
{ _id: 5be072ad84a05829a4a8551e,
  name: 'CR7',
  country: 'Portugal',
  age: 33,
  balondor: 5 }
PS C:\Users\dvaca\Downloads\mongo\practica3> |
```

### 3.4. Actualizar múltiples registros

#### 3.4.1. Agregar campos a un documento

Si deseamos actualizar varios documentos a la vez tenemos la función **updateMany** la cual funciona exactamente igual que la función **updateOne** con la diferencia que ahora se

**Elaborado por:**

Ing. Danny Vaca

**Fecha de revisión:**

5 noviembre del 2018

actualizan todos los registros que cumplan las condiciones no solo el primero que cumpla las condiciones.

Como mongo es una base de datos no estructurada, no hay una estructura previamente definida, por lo que se pueden agregar o eliminar campos sin mayor problema, cuando un campo no existe y tratamos de acceder a él, este tiene un valor de **null**.

```
JS delete.js JS update.js x
1  const mongodb = require('mongodb')
2  const MongoClient = mongodb.MongoClient
3  const url = 'mongodb://localhost:27017/testdatabase'
4
5  MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
6    if (error) throw error
7    const database = connection.db()
8    const conditions = {balondor: null}
9    const newValues = {$set:{balondor: 0}}
10   database.collection('players').updateMany(conditions, newValues, function(error, commandResult) {
11     if (error) throw error
12     database.collection('players').find({}).toArray(function(error, players) {
13       if (error) throw error
14       console.log(players)
15       connection.close()
16     })
17   })
18 })
19
```

Podemos observar como todos los documentos que antes no tenía un valor de **balondor** ahora tiene un valor de 0.

```
country: 'Uruguay',
age: 31,
balondor: 0 },
{ _id: 5be072ad84a05829a4a85529,
  name: 'Luka Modric',
  country: 'Croacia',
  age: 31,
  balondor: 0 } ]
PS C:\Users\dvaca\Downloads\mongo\practica3>
```

### 3.4.2. Eliminar campos de un documento

Para eliminar un campo de un documento podemos usar la función **\$unset** dentro del objeto de nuevos valores que tendrán los documentos.

<b>Elaborado por:</b> Ing. Danny Vaca	<b>Fecha de revisión:</b> 5 noviembre del 2018
--	---

```
JS delete.js JS update.js x
1  const mongodb = require('mongodb')
2  const MongoClient = mongodb.MongoClient
3  const url = 'mongodb://localhost:27017/testdatabase'
4
5  MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
6    if (error) throw error
7    const database = connection.db()
8    const conditions = {}
9    const newValues = {$unset:{country: 1}}
10   database.collection('players').updateMany(conditions, newValues, function(error, commandResult) {
11     if (error) throw error
12     database.collection('players').find({}).toArray(function(error, players) {
13       if (error) throw error
14       console.log(players)
15       connection.close()
16     })
17   })
18 })
19
```

Podemos observar como los nuevos registros no tienen el campo **country** ahora.

```
{ _id: 5be072ad84a05829a4a85524,
  name: 'Luis Suarez',
  age: 31,
  balondor: 0 },
{ _id: 5be072ad84a05829a4a85529,
  name: 'Luka Modric',
  age: 31,
  balondor: 0 } ]
PS C:\Users\dvaca\Downloads\mongo\practica3>
```