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

Danny Vaca

Índice

1. Objetivo general

Insertar y leer datos de MongoDB

2. Objetivo específico

- Realizar una conexión desde NodeJS hasta mongo
- Insertar datos en las colecciones
- Realizar consultas de datos

3. Desarrollo

La presente práctica se desarrollará a partir de la práctica 1.

3.1. Inserción de varios documentos a la vez

Vamos a renombrar el archivo **index.js** y lo vamos a llamar **crerate.js** ya que en este archivo vamos a crear los registros sobre los cuales vamos a hacer las posteriores consultas.

La función **inserOne** me permite insertar un solo objeto, si deseo inserta varios objetos la solución obvia es llamar varias veces a esta función, pero esto puede ser altamente ineficiente ya que se deben ejecutar varios comandos en la base de datos, mongo ofrece una función para insertar varios registros con un solo comando, esta función se llama **insertMany** la cual recibe un arreglo de objetos y cada uno será insertado en la colección.

```
JS create.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       console.log('Database created!')
8       const database = connection.db()
9       database.createCollection('players', function(error, collection) {
10          if (error) throw error
11          console.log('Collection created!')
12          const players = [
13              {name: 'Lionel Messi', country:'Argentina', age: 31 },
14              {name: 'Cristiano Ronaldo', country:'Portugal', age: 33 },
15              {name: 'Neymar Jr', country:'Brazil', age: 26 },
16              {name: 'Kilyan Mbappe', country:'Francia', age: 19 },
17              {name: 'Harry Kane', country:'Inglaterra', age: 26 },
18              {name: 'N\'Golo Kante', country:'Francia', age: 27 },
19              {name: 'Edison Cavani', country:'Urugay', age: 32 },
20              {name: 'Luis Suarez', country:'Uruguay', age: 31 },
21              {name: 'Paolo Dyabala', country:'Argentina', age: 24 },
22              {name: 'Antonie Griezmann', country:'Francia', age: 27 },
23              {name: 'Mohamed Salah', country:'Egipto', age: 26 },
24              {name: 'Eden Hazard', country:'Belgica', age: 29 },
25              {name: 'Luka Modric', country:'Croacia', age: 31 },
26          ]
27          collection.insertMany(players, function(error, commandResult) {
28              if (error) throw error
29              console.log('Players created!')
30              connection.close()
31          })
32      })
33  })
```

Si ejecutamos este archivo con el comando **node create.js** los datos serán insertados en la colección **players**

```
PS C:\Users\dvaca\Downloads\mongo\practica2> node .\create.js
Database created!
Collection created!
Players created!
PS C:\Users\dvaca\Downloads\mongo\practica2>
```

3.2. Extraer todos los registros de la base de datos

Para extraer datos de la base de datos necesitamos tener referencia a la colección de la cual tenemos que extraer los datos, podemos hacerlo con la función **createCollection** la cual si

ya existe la colección simplemente nos devuelve la referencia a la misma, pero también se puede extraer la referencia, con la función **collection** del objeto **database**.

Esta función nos devuelve la referencia a la colección y a partir de ella podemos llamar a la función **find** para realizar una consulta sobre la base de datos, esta función recibe como argumento las condiciones de la consulta y retorna un objeto sobre el cual debemos llamar a la función **toArray** para extraer los datos en forma de un array.

Esta función nos devolverá todos las coincidencias en un array.

```
{ _id: 5be072ad84a05829a4a85524,
  name: 'Luis Suarez',
  country: 'Uruguay',
  age: 31 },
{ _id: 5be072ad84a05829a4a85525,
  name: 'Paolo Dyabala',
  country: 'Argentina',
  age: 24 },
{ _id: 5be072ad84a05829a4a85526,
  name: 'Antonie Griezmann',
  country: 'Francia',
  age: 27 },
{ _id: 5be072ad84a05829a4a85527,
  name: 'Mohamed Salah',
  country: 'Egipto',
  age: 26 },
{ _id: 5be072ad84a05829a4a85528,
  name: 'Eden Hazard',
  country: 'Belgica',
  age: 29 },
{ _id: 5be072ad84a05829a4a85529,
  name: 'Luka Modric',
  country: 'Croacia',
  age: 31 } ]
```

3.3. Búsquedas por condiciones

3.3.1. Coincidencias exactas

La función **find** puede recibir como argumento las condiciones con las que se debe ahcer las búsquedas, estas condiciones son enviadas en un objeto y puede haber tantas condiciones como se necesiten, en este caso vamos a buscar todos los **playars** cuyo **country** sea **Argentina.**

create.js index.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 = {country: 'Argentina'}
9      database.collection('players').find(conditions).toArray(function(error, players) {
10         if (error) throw error
11         console.log(players)
12         connection.close()
13     })
14 })
15
```

```
Database created!
[ { _id: 5be072ad84a05829a4a8551d,
    name: 'Lionel Messi',
    country: 'Argentina',
    age: 31 },
  { _id: 5be072ad84a05829a4a85525,
    name: 'Paolo Dyabala',
    country: 'Argentina',
    age: 24 } ]
PS C:\Users\dvaca\Downloads\mongo\practica2>
```

3.3.2. Rangos

Si se desean buscar por rangos, es decir que sea mayor o menor que cierto valor debemos usar los comparadores

$gt para mayor que

$gte para mayor o igual que

$lt para menor que

$lte para menor o igual que

$eq igual que

$ne no es igual que

$in dentro de un rango

$nin no está dentro de un rango

```
JS create.js    JS index.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: {$gt: 32}}
9       database.collection('players').find(conditions).toArray(function(error, players) {
10          if (error) throw error
11          console.log(players)
12          connection.close()
13      })
14  })
```

3.3.3. Extraer ciertos campos

La función **find** puede recibir un segundo argumento en el cual se indique que campos se desea extraer ya que, si el documento tiene 80 elementos, no es lo más eficiente traer todos los registros.

```
JS create.js    JS index.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: {$gt: 31}}
9       const fields = {projection: {name: 1, age: 1, _id:0}}
10      database.collection('players').find(conditions, fields).toArray(function(error, player) {
11          if (error) throw error
12          console.log(player)
13          connection.close()
14      })
15  })
16
PS C:\Users\dvaca\Downloads\mongo\practica2> node .\index.js
[ { name: 'Cristiano Ronaldo', age: 33 },
  { name: 'Edison Cavani', age: 32 } ]
PS C:\Users\dvaca\Downloads\mongo\practica2>
```

3.4. Búsquedas de un solo registro

Si la búsqueda nos va a devolver un solo registro o si nos deseamos quedar solo con la primera coincidencia podríamos tomar el primer elemento del array que es devuelto por **find** o podría usar la función **findOne** para obtener solo el primer registro.

De la misma forma que la función **find** recibe las condiciones de la consulta, pero además recibe el callback que se va a ejecutar cuando se obtengan los registros.

```
JS create.js   JS index.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: {$gt: 32}}
9       database.collection('players').findOne(conditions, function(error, player) {
10          if (error) throw error
11          console.log(player)
12          connection.close()
13      })
14  })
15
```

Ya no hay necesidad de llamar a la función **toArray** ya que **findOne** automáticamente nso devuelve un solo registro en forma de objeto.

```
PS C:\Users\dvaca\Downloads\mongo\practica2> node .\index.js
{ _id: 5be072ad84a05829a4a8551e,
  name: 'Cristiano Ronaldo',
  country: 'Portugal',
  age: 33 }
PS C:\Users\dvaca\Downloads\mongo\practica2>
```