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

Danny Vaca

Índice

1. Objetivo general

Realizar consultas agrupando los registros de la base de datos

2. Objetivo específico

- Realizar una conexión desde NodeJS hasta mongo
- Realizar consultas agrupadas de la base de datos

3. Desarrollo

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

3.1. Agrupar registros

En ocasiones es necesario poder contar o realizar alguna operación con los registros, pero de forma agrupada, por ejemplo, de nuestro dataset de futbolistas, queremos ver cuantos futbolistas tenemos por nacionalidad o cual es la edad promedio de los futbolistas por nacionalidad, esto puede ser útil para hacer gráficos estadísticos de la distribución de datos.

3.1.1. Conteo de datos

Para lograr agrupar datos respecto a un campo y contar la cantidad de datos que coinciden, tenemos la función **aggregate** que entre otras cosas nos permite agrupar los datos de una consulta.

La función **agregate** recibe como primer argumento un array de condiciones, entras las cuales vamos a usar el operador **$group** el cual será un objeto el cual tendrá dos propiedades, el **_id** que será la clave respecto a la cual se realizará la búsqueda, para este caso se usará la nacionalidad de los jugadores es decir **country** y como segunda propiedad recibirá otro objeto llamado **count** el cual tendrá la propiedad **$sum** igual a **1** para indicar que debemos sumar de uno en uno cada coincidencia en el campo especificado como **_id**.

```
const url = 'mongodb://localhost:27017/testdatabase'

MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
    if (error) throw error
    const database = connection.db()
    database.collection('players').aggregate([
        {
            $group: {
                _id: '$country',
                count: {
                    $sum: 1
                }
            }
        }
    ]).toArray(function(error, player) {
        if (error) throw error
        console.log(player)
        connection.close()
    })
})

```

Al resultado de esta función **aggregate** se le llama la función **toArray** para poder obtener los datos en un array.

```
[ { _id: 'Croacia', count: 1 },
  { _id: 'Uruguay', count: 1 },
  { _id: 'Egipto', count: 1 },
  { _id: 'Belgica', count: 1 },
  { _id: 'Francia', count: 3 },
  { _id: 'Inglaterra', count: 1 },
  { _id: 'Urugay', count: 1 },
  { _id: 'Brazil', count: 1 },
  { _id: 'Portugal', count: 1 },
  { _id: 'Argentina', count: 2 } ]
```

3.1.2. Promedio de datos

Para lograr agrupar datos respecto a un campo y promediar los datos que coinciden respecto al algún valor, usamos la misma función **aggregate** pero ahora en lugar de usar el operador **$sum** para sumar, vamos a usar el operador **$avg** para promediar el campo **$age** es decir la edad de los jugadores.

```
MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
    if (error) throw error
    const database = connection.db()
    database.collection('players').aggregate([
        {
            $group: {
                _id: '$country',
                count: {
                    $avg: '$age'
                }
            }
        }
    ]).toArray(function(error, player) {
        if (error) throw error
        console.log(player)
        connection.close()
    })
})
```

Esto nos regresará en el campo **count** el valor promedio de la edad ya no la cantidad de futbolistas por país.

```
MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
    if (error) throw error
    const database = connection.db()
    database.collection('players').aggregate([
        {
            $group: {
                _id: '$country',
                count: {
                    $avg: '$age'
                }
            }
        }
    ]).toArray(function(error, player) {
        if (error) throw error
        console.log(player)
        connection.close()
    })
})
```

3.1.3. Promedio y conteo de datos

Se puede obtener la cantidad de datos y el promedio a la vez, ya que el campo **count** no es un campo único, sino un campo que definimos nosotros., es decir podemos definir tantos como nos haga falta con el nombre que deseemos, y cada campo puede usar un operador diferente.

```
MongoClient.connect(url, { useNewUrlParser: true }, function(error, connection) {
    if (error) throw error
    const database = connection.db()
    database.collection('players').aggregate([
        {
            $group: {
                _id: '$country',
                total: {
                    $sum: 1
                },
                average: {
                    $avg: '$age'
                }
            }
        }
    ]).toArray(function(error, player) {
        if (error) throw error
        console.log(player)
        connection.close()
    })
})
```

```
[ { _id: 'Croacia', total: 1, average: 31 },
  { _id: 'Uruguay', total: 1, average: 31 },
  { _id: 'Egipto', total: 1, average: 26 },
  { _id: 'Belgica', total: 1, average: 29 },
  { _id: 'Francia', total: 3, average: 24.333333333333332 },
  { _id: 'Inglaterra', total: 1, average: 26 },
  { _id: 'Urugay', total: 1, average: 32 },
  { _id: 'Brazil', total: 1, average: 26 },
  { _id: 'Portugal', total: 1, average: 33 },
  { _id: 'Argentina', total: 2, average: 27.5 } ]
```