Apuntes MongoDB - Abel Rios - 25/04/2022

Creamos una carpeta que llamaremos MongoDB y la abrimos en el Visual Studio. Ahora abrimos una terminal e instalamos el MongoDB con el comando “npm install mongodb“. Hecho esto, empezamos nuestro archivo JS con:

const { MongoClient } = require (‘mongodb’);

Y ahora conectamos con nuestra base de datos local:

```
GIT > BasesDatos > MongoDB > ejercicios > JS mongodb.js > client
1   const { MongoClient } = require('mongodb');
2
3   const client = new MongoClient("mongodb://localhost:27017");
4
5   async function obtenerUsuarios(){
6       await client.connect();
7       await client.db("ecommerce").command({ping:1});
8
9       console.log("Conectado a MongoDB!");
10  }
11
12  obtenerUsuarios();
13
```

**Aquí, si nuestra base de datos estuviera online (url) lo pondríamos en la línea 3 en lugar de localhost.

Y conectamos, a través de la terminal en Visual Studio:

```
    at node:internal/main/run_main_module:17:47
PS C:\Users\Abel\Documents\Releevant\git\basesdatos\mongodb\ejercicios> node mongodb.js
Conectado a MongoDB!
```

Añadimos las líneas 11 y 13:

```
9       console.log("Conectado a MongoDB!");
10
11      const usuarios = client.db("ecommerce").collection("usuarios").find();
12
13      console.log(usuarios);
14  }
```

Y al ejecutar de nuevo la consola nos devolverá un cursor apuntando a la base de datos:

```
PS C:\Users\Abel\Documents\Releevant\git\BasesDatos\MongoDB\ejercicios> node mongodb.js
Conectado a MongoDB!
FindCursor {
  _events: [Object: null prototype] {},
  _eventsCount: 0,
  _maxListeners: undefined,
  [Symbol(kCapture)]: false,
  [Symbol(topology)]: Topology {
    _events: [Object: null prototype] {
      connectionPoolCreated: [Function (anonymous)],
      connectionPoolClosed: [Function (anonymous)],
      connectionCreated: [Function (anonymous)],
      connectionReady: [Function (anonymous)],
      connectionClosed: [Function (anonymous)],
      connectionCheckOutStarted: [Function (anonymous)],
      connectionCheckOutFailed: [Function (anonymous)],
      connectionCheckedOut: [Function (anonymous)],
      connectionCheckedIn: [Function (anonymous)],
      connectionPoolCleared: [Function (anonymous)],
      commandStarted: [Function (anonymous)],
      commandSucceeded: [Function (anonymous)],
      commandFailed: [Function (anonymous)],
      serverOpening: [Function (anonymous)],
      serverClosed: [Function (anonymous)],
      serverDescriptionChanged: [Function (anonymous)],
      topologyOpening: [Function (anonymous)],
      topologyClosed: [Function (anonymous)],
      topologyDescriptionChanged: [Function (anonymous)],
```

Una vez tenemos ese cursor, debemos iterar cada uno de los registros:

```
14
15      usuarios.forEach((u)=> console.log(u));
16  }
```

De esta manera por consola nos devolverá ahora el listado de usuarios (también he eliminado la línea de **console.log(usuarios)** para que no me devuelva en la consola el cursor entero, sólo me devuelva el listado de registros que le acabamos de pedir.

```
PS C:\Users\Abel\Documents\Releevant\git\BasesDatos\MongoDB\ejercicios> node mongodb.js
Conectado a MongoDB!
{
  _id: new ObjectId("6262bc1a403760c20adab527"),
  nombre: 'Jose',
  apellidos: 'Alcala',
  telefono: '1122334455',
  correo: 'jose@gmail.com',
  password: '1234',
  direccion: 'C/ Trebujena 22',
  tipoPago: 'Paypal'
}
```

Y para cerrar la conexión con la base de datos lo hacemos así:

```
12
13      await usuarios.forEach((u)=> console.log(u));
14      await client.close();
15
16      console.log("Desconectado de MongoDB!");
17  }
```

Hemos puesto el **'await' en la línea 13 para que no salte directamente a la línea 14, si no lo pusiéramos antes de mostrarnos los registros de 'usuarios' se desconectaría del servidor. Al ser promesas los métodos que estamos utilizando, mezclada con la base de datos, tenemos que usar funciones asíncronas (que no esperan el resultado). El hecho de ser asíncrono es porque no se sabe cuánto va a tardar los datos de la base de datos en volver a nuestro programa.

EJ: Obtener el listado de ventas con nombre de usuario y el tipo de pago realizado (si la colección ventas no tiene tipo de pago, incluirlo)

```
async function ventasUsuarioYPago(){
    await client.connect();
    await client.db("ecommerce").command({ping:1});

    console.log("Conectado a Ecommerce");

    const result = client.db("ecommerce").collection("ventas").find();

    await result.forEach((i)=> console.log(i.usuario.nombre,i.usuario.tipoPago));

    await client.close();

    console.log("Desconectado de MongoDB!");
}
```

** Y de esta forma hacemos el mismo filtrado en el find y no nos traemos todos los objetos enteros a la variable result, sino que nos traemos sólo las propiedades de los objetos que hemos filtrado:

```
async function ventasUsuarioYPago(){  // vamos a intentar hacer el filtrado en el FIND
    await client.connect();
    await client.db("ecommerce").command({ping:1});

    console.log("Conectado a Ecommerce");

    const result = client.db("ecommerce").collection("ventas").find().project({
        _id:0,
        "usuario.nombre":true,
        "usuario.tipoPago":true});

    await result.forEach((i)=> console.log(i));

    await client.close();

    console.log("Desconectado de MongoDB!");
}
```

EJ: Sacar por consola lista de productos donde aparezca su nombre y la cantidad vendida

```
async function productosNombreYCantidad(){
    await client.connect();
    await client.db("ecommerce").command({ping:1});

    console.log("Conectado a Ecommerce");
    const result = client.db("ecommerce").collection("ventas").find().project({
        _id:0,
        "producto.nombre":true,
        "cantidad":true
    });

    await result.forEach((i)=> console.log(i));

    await client.close();

    console.log("Desconectado de MongoDB!");
}
```

** Hacerlo con el group by:

```
async function productosNombreYCantidadAgrupados(){
    await client.connect();
    await client.db("ecommerce").command({ping:1});

    console.log("Conectado a Ecommerce");
    const ventas = client.db("ecommerce").collection("ventas").find().project({
        _id:0,
        "producto.nombre":true,
        "cantidad":true
    });

    let results=[];

    await ventas.forEach(function(v){
        let found = false;

        for (let i = 0; i < results.length; i++){
            if (results[i].producto.nombre === v.producto.nombre){
                results[i].cantidad += v.cantidad;

                found = true;

                break;
            }
        }
        }

        if(!found){
            results.push(v);
        }

    });

    console.log(results);

    await client.close();

    console.log("Desconectado de MongoDB!");
}
```

EJ: Sacar por consola Lista de productos que se muestre el nombre y el rating

```
async function productosNombreYRating(){
    await client.connect();
    await client.db("ecommerce").command({ping:1});

    console.log("Conectado a Ecommerce");
    const result = client.db("ecommerce").collection("productos").find().project({
        _id:0,
        "nombre":true,
        "rating":true
    });

    await result.forEach((i)=> console.log(i));

    await client.close();

    console.log("Desconectado de MongoDB!");
}
```

EJ: Sacar por consola cual es el producto más vendido y nombre de usuarios han comprado dicho producto

(lo tenemos en el archivo de ejercicio mongo.js)