Acceso a MongoDB desde Java

Introducción

- Para trabajar en Java con MongoDB necesitamos descargar el driver desde la URL de MongoDB https://mongodb.github.io/mongo-java-driver/
- La versión que vamos a utilizar es la **mongodb-java-driver-3.10.1.jar**
- Antes de empezar definamos **BSON:** BSON es un formato de serialización binaria, se utiliza para almacenar documentos y hacer llamadas a procedimientos en MongoDB. La especificación BSON se encuentra en **bsonspec.org**. BSON soporta los siguientes tipos de datos como valores en los documentos, cada tipo de dato tiene un número y un alias que se pueden utilizar con el operador **$type** para consultar los documentos de tipo BSON.
- Algunos de los tipos BSON son (Tipo, Numero, Alias): Double, 1, “double”; String, 2, “string”; Object, 3, “object”; Array, 4, “array”; Binary data, 5, “binData”; ObjectId, 7, “objectId”; Boolean, 8, “bool”; Date, 9, “date”; Null, 10, “null”; Symbol, 14, “symbol”; Timestamp, 17, “timestamp”

Conexión a la BD

- Para conectarnos a la bbdd creamos una instancia **MongoClient**. Por defecto crea una conexión con la base de datos local, y escucha por el puerto 27017.
- Todos los métodos relacionados con operaciones CRUD (Create, Read, Update, Delete) en Java se acceden a través de la interfaz **MongoCollection**.
- Las instancias de **MongoCollection** se pueden obtener a partir de una instancia **MongoClient** por medio de una **MongoDatabase**.
- Veamos cómo nos conectamos a la base de datos *mibasedatos* y a la colección *amigos*:

```
MongoClient  cliente = new MongoClient();
MongoDatabase db = cliente.getDatabase (“mibasedatos”);
MongoCollection <Document> coleccion = db.getCollection(“amigos”);
```

- MongoCollection es una interfaz genérica: el parámetro de tipo Document es la clase que los clientes utilizan para insertar o modificar los documentos de una colección y es el tipo predeterminado para devolver búsquedas (find).
- El método de un solo argumento **getCollection** devuelve una instancia de **MongoCollection <Document>,** y así es como podemos trabajar con instancias de la clase de documento.

Visualizar los datos de una colección

- Los datos de una colección se pueden cargar en una lista utilizando el método **find().into()** de la siguiente manera:

```
List<Document> consulta = coleccion.find().into(new ArrayList<Document> ());
for (int i =0; i < consulta.size(); i++) {
        System.out.println(“ - “ +  consulta.get(i).toString());
}
```

Insertar documentos

- Para insertar documentos, creamos un objeto Document, con el método **put** asignamos los pares *clave-valor*, donde el primer parámetro es el nombre del campo o la clave, y el segundo el valor. Y con el método **insertOne** se inserta en la colección
- El siguiente código añade un amigo a la colección:

```
Document amigo = new Document();
amigo.put(“nombre”, “Pepito”);
amigo.put(“teléfono”, 925677);
amigo.put(“curso”,”2DAM”);
amigo.put(“fecha”, new Date() );
coleccion.insertOne(amigo);
```

- También se puede insertar documentos utilizando el método **append** de **Document.**

Consultar documentos

- Anteriormente se ha visto cómo cargar los documentos en una lista utilizando el método **find().into()**.
- El método **find()** devuelve un cursor, en concreto devuelve una instancia **FindIterable**. Podemos utilizar el método **iterator()** para recorrer el cursor.
- En el ejemplo recuperamos todos los documentos de la colección y se visualizan en formato Json:

```
MongoCursor<Document> cursor = coleccion.find().iterator();
while (cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println (doc.toJson());
}
cursor.close();
```