





3. Conexión con base de datos Mongo

 Date	Empty
 Status	Not started
 Type	Empty
Units	 9. Base de datos

Bases de datos no relacionales

Configuraciones previas

Crear la base de datos

Conexión con la base de datos

Conexión desde shell

Operaciones mongo

Inserción

Actualizaciones

Borrados

Búsquedas

Conectar Mongo con Java

Operaciones

Inserción

Borrado

Actualización

Selección

Bases de datos no relacionales

En los últimos años, han surgido un nuevo tipo de base de datos que permite una comunicación e interacción con los datos de una forma muy rápida, sencilla y segura. Se trata de las bases de datos no relacionales, que tal y como su nombre indica son aquellas que no

utilizan un esquema de tablas con filas y columnas, sino que utilizan un esquema de colecciones, donde en cada una de ellas se pueden insertar documentos con par clave – valor, tal y como lo hace un documento JSON. Al tratarse de base de datos no relacionales, el lenguaje de base de datos utilizado en los motores relacionales no es válido, por lo que en este tipo de base de datos se utiliza una sintaxis diferente. Las principales características de estas bases de datos son las siguientes:

- Escalabilidad: Se trata de bases de datos que pueden ser utilizadas en clúster (servidores remotos), por lo que si se necesitan más recursos para los datos es tan sencillo como contratar características hw adicionales para soportar los cambios
- Flexibilidad: Se trata de bases de datos que no están sujetos a un esquema fijo, sino que los documentos pueden añadir o quitar datos que otros documentos de la misma colección no tiene
- Alto rendimiento: Se trata de bases de cuya organización interna está optimizada para el tratamiento de datos, lo que hace que este sea muy rápido y eficiente. Del mismo modo, las bases de datos no relacionales permiten un acceso mediante APIs con un modelo de datos específico para las colecciones y documentos que están guardadas.

Existen diferentes tipos de bases de datos no relacionales, en este apartado nos vamos a centrar en un tipo concreto que es la de los colecciones y documentos. En concreto para este tema vamos a utilizar las bases de datos documentales MongoDB, las cuales tienen el siguiente esquema dentro de su estructura interna

- MongoDB
 - Collection
 - Document

Un ejemplo sería el siguiente

- Database: concesionario
- Collection: coches
- Document:

```
{ marca: "Mercedes", modelo: "c220", precio: 40000, combustio  
n: "gasoil" cc:30000, cv: 250 }
```

Como se puede ver, cada uno de los documentos son representados por un JSON, pudiendo haber tantos como sean necesarios cuya estructura puede que no sea la misma dentro de la misma colección

Configuraciones previas

Antes de empezar a trabajar con la base de datos, su conexión y la manipulación de la información, es necesario configurar la base de datos para que esta pueda existir y así conectarnos a ella. Para eso existen dos posibilidades: trabajar en local o utilizar un clúster accesible mediante web que permita la conexión y trabajo desde cualquier punto. Evidentemente el segundo caso es bastante más eficiente ya que independizamos el acceso físico y podemos ejecutar nuestro código desde cualquier punto. Para ello se va a utilizar MongoAtlas, cluster de mongoDB que ofrece un plan gratuito. Para poder hacer esto es necesario realizar los siguientes pasos:

1. Accedemos a la página web <https://www.mongodb.com/es/atlas/database> e indicamos que queremos una prueba gratuita con el botón superior. Se puede iniciar con una cuenta de Google, por lo que no sería necesario crear una cuenta
2. Una vez iniciada sesión y tras aceptar los términos de uso y responder a una serie de preguntas sobre el uso de los desarrollos que se van a realizar, es necesario seleccionar el plan sobre el cual trabajaremos en nuestro clúster. En nuestro caso utilizaremos un plan Shared, el cual nos da una infraestructura básica pero funcional sin necesidad de pagar dinero
3. El siguiente paso nos pedirá una serie de configuraciones de donde queremos que se aloje nuestro clúster, el tipo de HW que tendrá y alguna configuración adicional. No es necesario que cambiemos nada de los valores que vienen por defecto, poniendo especial atención al nombre del clúster otorgado, que por regla general será Cluster0

Con estas configuraciones ya tenemos preparado el cluter para empezar a crear bases de datos con colecciones y documentos.

Crear la base de datos

Con el clúster creado, si lo seleccionamos podremos crear una base de datos. Para este ejemplo crearemos una base de datos llamada academia y una colección llamada alumnos.

A partir de este momento se pueden incluir datos, crear colecciones y sobre ellas realizar inserciones de documentos, búsquedas, modificaciones, etc... Una vez realizado esto podremos conectarnos a dicha base de datos

Conexión con la base de datos

Para conectarnos con la base de datos, MongoDB ofrece diferentes posibilidades como son la conexión desde shell o la conexión desde driver. Vamos a explicar ambas:

Conexión desde shell

En el caso de querer conectar de forma puntual con nuestra base de datos desde la terminal, tendremos que previamente instalar un paquete dependiendo de cuál es el sistema operativo desde el que queremos hacerlo. Para ello se ofrece una herramienta que indica cuál es la descarga necesaria. Basta con irnos a la pestaña de Cmd Line Tools y seleccionar Connect to Your Cluster. Esto abrirá un cuadro de diálogo donde seleccionaremos shell, siguiendo las instrucciones para cada uno de los sistemas operativos que seleccionemos

Es importante tener en cuenta que para windows es necesario configurar las variables de entorno

Una vez se han seguido todos los pasos podremos conectar con la base de datos desde la terminal introduciendo el siguiente comando

```
mongosh "mongodb+srv://cluster0.atboh0.mongodb.net" --apiVersion 1 --username root
```

Por defecto esta conexión se realiza sobre una base de datos llamada test, pero en el caso de querer conectarnos a la base de datos que hemos creado pondremos el comando

```
use academia
```

Todo lo que hagamos sobre la línea de comandos se realizará sobre la

...es lo que hagamos sobre la forma de comandos se realizará sobre la base de datos seleccionada.

Operaciones mongo

A la hora de hacer cualquier acción dentro de mongo, indicar que se puede realizar de forma individual o de forma masiva.

Inserción

Se utilizan los métodos insertOne o insertMany

```
db.alumnos.insertOne({nombre:"Borja", apellidos: "Martin Herrera",telefono: 12345, intereses: ["deporte","tecnología"]})
```

```
db.alumnos.insertMany([{"nombre:"Juan", apellidos: "Herrera Martín",telefono: 12345, intereses: ["musica","cine"]}, {"nombre:"Luis", apellidos: "Herrera Castillo",telefono: 12345, intereses: ["cine"]}])
```

Actualizaciones

Se utilizan los métodos updateOne o updateMany. La sintaxis de los comandos es la siguiente

```
updateOne(Criterio, Operacion)
```

Un ejemplo sobre uno de los alumnos creados en el punto anterior sería

```
db.alumnos.updateOne({nombre: "Borja"}, {$set: {telefono: 2345}})
```

En el caso de utilizar updateMany la sintaxis es la misma con la diferencia que actualizará todos aquellos elementos que cumplan la condición de búsqueda. Los modificadores más comunes que se utilizan son:

- set: asigna un valor nuevo

- unset: elimina un campo. En este caso hay que poner un booleano para indicar el borrado
- inc: incrementa el valor del campo
- push: añade elementos a un vector
- pull: elimina elemento de un vector
- pop: elimina elementos de un vector tratandolo como una pila (primero o último)

Borrados

Se utilizan los métodos `deleteOne`, `deleteMany` o `findOneAndDelete`. La diferencia entre ellos es que el último además de borrar el documento lo devuelve. La sintaxis del método es la siguiente:

```
deleteOne(Criterio)
```

Búsquedas

Para realizar las búsquedas se utiliza el método `find` o `findOne`, indicando el criterio de búsqueda en formato JSON

```
db.alumnos.find({nombre: "Borja"})
```

Adicionalmente se puede utilizar comparadores para realizar queries más exactas. Estos comparadores son los siguientes:

- lt: menor que
- lte: menor igual que
- gt: mayor que
- gte: mayor igual que
- in: comprendido entre (hay que indicar un array)
- nin: no comprendido entre (hay que indicar un array)
- eq: igual que
- \$or: anidación de o

Conectar Mongo con Java

Operaciones

Inserción

Los pasos a ejecutar son los siguientes:

1. Crear un documento con par clave - valor con los datos del objeto que se quiere agregar

```
Document document = new Document(); document.append("marca",  
c.getMarca()); document.append("modelo",c.getModelo()); docum  
ent.append("precio",c.getPrecio()); document.append("cv",c.ge  
tCc()); document.append("cc",c.getCc()); document.append("pes  
o",c.getPeso()); document.append("tipo_coche",c.getTipo());
```

1. Ejecutar la sentencia de inserción

```
collection.insertOne(document);
```

Borrado

Los pasos serán los siguientes

1. Crear un objeto con la condición de búsqueda

```
Document = new Document(); document.put("marca", "audi");
```

1. Ejecutar la sentencia de borrado

```
collection.deleteOne(document);
```

Actualización

Los pasos serán los siguientes:

1. Crear un objeto con los criterios de búsqueda

```
// objeto que se quiere actualizar collection = mongoDatabase.  
getCollection("coches"); Document objeto = new Document();  
objeto.put("modelo", "Fiesta");
```

1. Crear un objeto con los datos que se quieran cambiar

```
// actualización sobre el objeto Document objetoNuevo = new Document();  
objetoNuevo.put("cv", 100);
```

1. Crear un objeto con los modificadores a crear

```
// modificación de actualización Document updateObject = new Document();  
updateObject.put("$set", objetoNuevo);
```

1. Ejecutar la sentencia de actualización

```
collection.updateOne(objeto, updateObject);
```

Selección

Los pasos serán los siguientes:

1. Crear un objeto de tipo FindIterable mediante el método find

```
FindIterable<Document> findIterable = collection.find();
```

1. Obtener un iterador para poder recorrer el documento

```
MongoCursor<Document> iterator = findIterable.iterator();
```

1. Iterar el documento obtenido

```
while (iterator.hasNext()){ Document document = iterator.next();  
System.out.println(document.get("marca")); System.out.println();
```



```
println(document.get("modelo")); System.out.println(document.get  
("cv")); System.out.println("-----"); }
```

En el caso de querer obtener elementos con una condición concreta, será necesario pasar al método find un documento con la condición de búsqueda

```
Document = new Document("cv", new Document().append("$lt",16  
0)); FindIterable<Document> documents = collection.find(docum  
ent); MongoClient<Document> iterador = documents.iterator();  
while (iterador.hasNext()){ Document item = iterador.next();  
System.out.println(item.get("modelo")); }
```