GUÍA MONGODB CRUD

FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

LABORATORIO No. 03 – MONGODB CRUD

OBJETIVOS

- Conocer las características principales de las bases de datos documentales.
- Aprender a utilizar las operaciones CRUD para la manipulación de los datos en MongoDB.
- Aprender a utilizar el API de JAVA para establecer conectividad y acceso a una base de datos en MongoDB.
- Comprender la versatilidad y flexibilidad de la estructura de las bases de datos documentales MongoDB.

LECTURAS PREVIAS Y MATERIAL DE CONSULTA

- MongoDB en español: T1, El principio[1]

MARCO TEÓRICO

MongoDB es una base de datos documental de código abierto que proporciona un alto rendimiento, disponibilidad y escalabilidad. Los datos se almacenan en colecciones y estas en bases de datos; las colecciones almacenan datos relacionados, de igual forma contienen documentos que tienen otro nombre para los atributos. Todos los datos se almacenan y consultan en BSON obteniendo una identificación del objeto, soporte de expresiones regulares, fecha y datos binarios. Proporciona la persistencia de datos del rendimiento en particular, apoyando a los modelos de datos incrustados y consultas más rápidas [2].

BSON permite mayor flexibilidad, este fue diseñado para tener las siguientes características [2]: Ligero: Mantiene sobrecarga mínima, importante para cualquier formato de representación de datos cuando se utiliza en la red.
Atravesable: Es una propiedad fundamental en su función de representación de datos principal para MongoDB.

Eficiente: la codificación y decodificación de datos BSON se realizan muy rápidamente ya que utiliza tipos de datos C.

MATERIAL A UTILTIZAR

- Máquina virtual de Windows 7 de 64 bits con mínimo 2 GB de RAM. La máquina virtual debe contener:
  - MongoDB instalado.
  - IDE Netbeans.
  - API de MongoDB para JAVA.

PROCEDIMIENTO

1. Iniciar MongoDB.

Para iniciar MogonDB se ejecuta el archivo '`mongod.exe`'. Se abre el símbolo del sistema y se escriben los siguientes comandos:

En Windows:

Instrucción

```
> C:/mongodb/bin/mongod.exe
```

En Ubuntu:

Instrucción

```
> sudo service mongod start
```

2. Conectarse a MongoDB:

Para conectarse a MongoDB se abre otra línea de comandos y se ejecuta el archivo '`mongo.exe`', así:
En Windows:
Instrucción

```
> C:/mongodb/bin/mongo.exe
```

Si ya se desea dejar de usar MongoDB, se presiona '`Control + C`' en la terminal donde el archivo '`mongod`' está ejecutando la instancia.

En Ubuntu:
Se trabaja sobre la misma ventana de comandos.

3. Crear una base de datos y sus colecciones

Para crear una base de datos (que vamos a llamar '`prueba`') se escribe '`use`' y seguido el nombre de la base de datos, así:

1. Crear una base de datos:

Instrucción

```
> use prueba
```

Para visualizar las bases de datos existentes se utiliza la siguiente línea de código:

Instrucción

```
> show dbs
```

Resultado

```
local   0.078GB
prueba  0.078GB
```

Figura 2 Lista bds

Tener en cuenta que:

- La base de datos '`local`' es creada automáticamente por MongoDB.
- La base de datos creada sólo se mostrará si por lo menos tiene una colección, y esa colección tiene por lo menos un documento almacenado.

La misma instrucción '`use`' sirve para seleccionar la base de datos que quiero manipular. Por ejemplo, si tengo tres bases de datos creadas. Una se llama '`prueba`', la otra '`prueba2`' y la tercera '`prueba3`'. Y se desea manipular la base de datos '`prueba`', se escribe la siguiente línea en la consola.

Instrucción

```
> use prueba
```

Resultado

```
switched to db prueba
```

Figura 3 bd actual

Como se muestra en la Figura 3 bd actual en este momento ya se podría empezar a trabajar con la base de datos '`prueba`'.

2. Crear colecciones:

Se procede a crear una colección para esta nueva base de datos (que llamaremos 'ejemplo') así:

Instrucción

```
> db.ejemplo
```

Para visualizar las colecciones de la nueva base de datos, se escribe la siguiente instrucción:

Instrucción

```
> show collections
```

En la Figura 4 Lista de colecciones se mostrará el listado de las dos colecciones que contiene la base de datos.

Resultado

```
ejemplo
system.indexes
```

Figura 4 Lista de colecciones

Tener en cuenta que:

- La colección '`system.indexes`' es creada automáticamente por MongoDB.
- La colección solo se mostrará si tiene por lo menos un documento almacenado.

4. Realizar sentencias CRUD básicas en MongoDB

MongoDB provee un grupo de métodos en JavaScript para realizar CRUD en nuestra base de datos a través de la consola.

1. Create:

Para realizar la inserción de un documento utilizamos la función '`insert()`'. Si queremos crear un documento con un campo 'materia' que contenga el valor 'Programación I', la instrucción se escribiría así en la consola:

Instrucción:

```
> db.ejemplo.insert({materia: "Programación I"})
```

Si la inserción es correcta, retornara en consola lo que aparece en la Figura 5 Resultado inserción:

Resultado:

```
WriteResult({ "nInserted" : 1 })
```

Figura 5 Resultado inserción

2. Read:

Para consultar los documentos almacenados en una colección se utiliza la siguiente función '`find()`', así:

Instrucción

```
> db.ejemplo.find()
```

Aparecerán todos los documentos asociados a esta colección, para este caso solo aparecerá el registro que se acaba de insertar, como se observa en la **Figura** 6 Resultado consulta:

Resultado

Figura 6 Resultado consulta

El campo '`_id`' es un campo obligatorio de tipo ObjectId que genera un valor de 12 bytes, de los cuales

4 bytes son de timestamp (fecha y hora).
3 bytes son del identificador de la computadora.
2 bytes son del número del proceso.
3 bytes son del contador que comienza con un número aleatorio.

El campo '`_id`' se genera para identificar de forma única cada documento que se guarda en la base de datos.

- **Visualizar la consulta de manera organizada:**

Si se desea visualizar la información de forma ordenada, se utiliza la función '`pretty()`', así:

Instrucción

```
> db.ejemplo.find().pretty()
```

Aparecerán todos los documentos asociados a esta colección. Para este caso solo aparecerá el registro que se acaba de insertar, de esta forma:

Resultado:

Figura 7 Resultado pretty

3. Update:

Para actualizar un documento se utiliza la función '`update()`' que incluye dos parámetros. En el primer parámetro va el filtro (condición), y en el siguiente parámetro se escriben los campos que se quieren modificar o agregar al documento, cada uno con su respectivo valor.

Si por ejemplo se desea modificar el documento donde el valor del campo 'materia' es 'Programación I', y se desea actualizar este campo con el valor 'Diseño de algoritmos'. Se escribirá la siguiente línea:

Instrucción

```
db.ejemplo.update({materia: "Programación I"},{materia: "Diseño de algoritmos"})
```

Resultado

```
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
```

Figura 8 Resultado actualización

4. Delete:

Para eliminar un documento se utiliza la función 'remove()' que puede eliminar uno o varios documentos de forma selectiva. Si no se le pasa ningún parámetro, elimina todos los documentos de una colección. En este caso se va a indicar que se eliminará el documento que contiene dentro del campo 'materia' el valor 'Diseño de algoritmos':

Instrucción

```
> db.ejemplo.remove({ materia : "Diseño de algoritmos" })
```

Resultado

```
WriteResult({ "nRemoved" : 1 })
```

Figura 9 Resultado delete

5. Realizar sentencias CRUD en varios documentos en MongoDB

En el tema anterior se utilizaron las funciones básicas para realizar sentencias CRUD sobre un solo documento, pero existen varios tipos de funciones JavaScript y técnicas que me permiten ejecutar estas sentencias sobre varios documentos de forma simultánea, permitiendo también utilizar condiciones muy similares a las utilizadas en el lenguaje SQL. A continuación se presentan algunos ejemplos para cada tipo de sentencia.

1. Insertar múltiples documentos:

Cuando se requiere insertar varios documentos de manera simultánea se utiliza la misma función 'insert()'. Esta vez como ejemplo se asigna de forma manual el valor del campo '_id', y también se adiciona el campo 'creditos' a cada documento. La excepción es el tercer documento que no contendrá este campo.

Instrucción

```
> db.ejemplo.insert(
```

```
      [
            {
                  _id: "Documento1",
                 materia: "TGS",
                creditos: 2
            },
            {
                 _id: "Documento2",
                 materia: "Bases de datos",
                creditos: 4
            },
            {
                 _id: "Documento3",
                 materia: "Estructuras de datos"
            },
      ]
)
```

- **Inserción utilizando variables:**

Otra forma de guardar los documentos podría ser utilizando variables JavaScript y la misma función '`insert()`'.

Instrucción

```
> materia1 = {_id: "Documento1", materia: "TGS", creditos: 2 }
> materia2 = {_id: "Documento2", materia: "Bases de datos",
 creditos: 4 }
> materia3 = {_id: "Documento3", materia: "Estructuras de datos" }
>
> db.ejemplo.insert([materia1, materia2, materia3])
```

Si la inserción de los tres documentos es correcta se imprimiría un informe de inserción en consola, como se ve en la Figura 10 Resultado inserción múltiple:

Resultado

```
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Figura 10 Resultado inserción múltiple

2. Consulta de documentos aplicando condiciones:

Para consultar los documentos que se acaban de almacenar en la colección, se utiliza la misma función '`find()`':

Instrucción

```
> db.ejemplo.find()
```

Resultado

Figura 11 Consulta inserción múltiple

- **Consultar un solo documento:**

En caso de que sólo se requiera visualizar el primer documento se utiliza '`findOne()`', por ejemplo:

Instrucción

```
> db.ejemplo.findOne()
```

Resultado

Figura 12 Resultado función findOne()

- **Consultar un documento aplicando un filtro:**

Si solo se desea consultar el documento que contenga el valor "`Bases de datos`" en el campo '`materia`', se enviaría esto como parámetro en la función '`find()`', así:

Instrucción

```
> db.ejemplo.find({"materia": "Bases de datos"})
```

Resultado

Figura 13 Resultado consulta con condición

- **Omitir en el resultado de la consulta algún campo:**

Si se desea omitir en el resultado el campo '`_id`', la instrucción quedaría así:

Instrucción

```
> db.ejemplo.find({"materia": "Bases de datos"},{"_id":false})
```

Resultado

Figura 14 Resultado consulta omitiendo campo _id

Si se desea mostrar sólo el campo '`materia`', la instrucción quedaría así:

Instrucción

```
> db.ejemplo.find({"materia": "Bases de datos"},{"_id":false,
 "materia":true})
```

Resultado

Figura 15 Resultado consulta omitiendo campos

- **Cantidad de documentos en una colección:**

Si se desea saber la cantidad de documentos almacenados en nuestra colección utilizamos la función '`count()`', así:

Instrucción

```
> db.ejemplo.count()
```

Resultado

```
3
```

- **Consultas con condiciones:**

Para aplicar condiciones a una consulta se utilizan las siguientes instrucciones:

Instrucción	Significado
$gt	Mayor que
$gte	Mayor o igual que
$lt	Menor que
$lte	Menor o igual que
$ne	Negación

Tabla 16 Operadores de relación

- **Ejemplo 1 read:**

Listar las materias que tengan más de 2 créditos.
Instrucción

```
> db.ejemplo.find({creditos: {$gt:2}})
```

Resultado

Figura 16 Resultado consulta con condicional $gt

- **Ejemplo 2 read:**

Listar las materias que no tengan 4 créditos.
Instrucción

```
> db.ejemplo.find({creditos: {$ne:4}})
```

Resultado

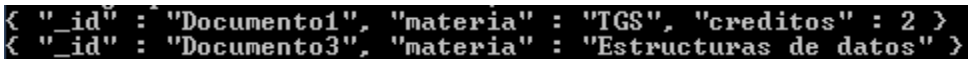


Figura 17 Resultado consulta con condicional $ne

3. Actualizar documentos y campos en específico:

Al actualizar un documento se pueden realizar cambios como adicionar, renombrar o eliminar campos y cambiar sus respectivos valores. Todas estas modificaciones se realizan utilizando la misma función '`update()`'. A continuación se realizan una serie de ejemplos donde se evidencia el uso de algunos de los operadores de la **Tabla 17** Operadores para actualización.

Instrucción	Significado
Operadores:	
$inc	Incrementa en una cantidad numérica especificada el valor del campo a en cuestión.
$rename	Renombrar campos del documento.
$set	Permite especificar los campos que van a ser modificados.
$unset	Eliminar campos del documento.
Operadores referentes a arreglos:	
$pop	Elimina el primer o último valor de un arreglo.
$pull	Elimina los valores de un arreglo que cumplan con el filtro indicado.
$pullAll	Elimina los valores especificados de un arreglo.
$push	Agrega un elemento a un arreglo.

Tabla 17 Operadores para actualización

- **Ejemplo 1 de update:**

Al documento que tiene el valor 'Documento3' en su campo '_id' adicionarle un campo numérico 'HorasSemana' con valor '3', otro campo numérico 'Semestre' con valor '3' y el campo 'credito' con valor '3'. También cambiar el valor del campo 'materia' por 'Estructuras de datos II' a '5'.[3]

Solución:
Miramos como está actualmente el documento:
Instrucción

```
> db.ejemplo.find({_id: "Documento3"})
```

Resultado

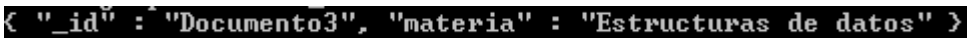


Figura 18 Resultado consulta ejemplo 1 update - registro sin modificar

Procedemos a escribir la sentencia para actualizar el documento.
Instrucción

```
> db.ejemplo.update(
   {_id: "Documento3"},
   {
```

```
      materia:"Estructuras de datos II",
      horasSemana:3,
      semestre:3,
      creditos:3
   }
)
```

Revisamos como quedó el documento:

Instrucción

```
> db.ejemplo.find({_id: "Documento3"})
```

Resultado

```
{ "_id" : "Documento3", "materia" : "Estructuras de datos II", "horasSemana" : 3
, "semestre" : 3, "creditos" : 3 }
```

Figura 19 Resultado consulta ejemplo 1 update - registro modificado

- **Ejemplo 2 de update:**

Al documento que tiene el valor 'Documento3' en su campo '_id' cambiarle el campo 'semestre' y renombrarlo con 'nivel'. Incrementar en 1 el número de créditos. Eliminar el campo 'horasSemana'.

Solución:

Instrucción

```
> db.ejemplo.update(
   {_id: "Documento3"},
   {
      $rename:{ "semestre" : "nivel"}
   }
)
```

Instrucción

```
> db.ejemplo.update(
   {_id: "Documento3"},
   {
      $inc:{creditos:1}
   }
)
```

Instrucción

```
> db.ejemplo.update(
   {_id: "Documento3"},
   {
      $unset:{horasSemana: ""}
   }
)
```

Revisamos como quedó el documento:

Instrucción

```
> db.ejemplo.find({_id: "Documento3"})
```

Resultado

```
{ "_id" : "Documento3", "materia" : "Estructuras de datos II", "creditos" : 4, "
nivel" : 3 }
```

Figura 20 Resultado consulta ejemplo 2 update - registro modificado

Se puede apreciar en la Figura 20 Resultado consulta ejemplo 2 update - registro modificado que el campo 'semestre' ahora se llama 'nivel', el campo 'creditos' aumentó a 4 el y el campo 'horasSemana' fue eliminado.

6. MongoDB desde JAVA

1. Descarga del driver de JAVA:

Para descargar el driver de JAVA para la conexión con MongoDB se van a tener en cuenta los siguientes pasos.

a. Ingresar a la página de mongoDB 'www.mongodb.org'.
b. Hacer click en 'Docs'.

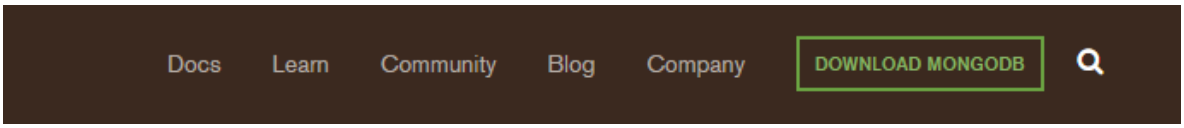


Figura 21 Página principal MongoDB

c. En la nueva ventana que aparece, hacer click en 'Drivers'.

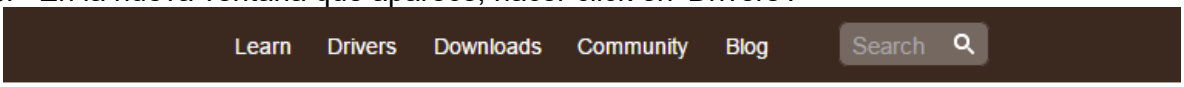
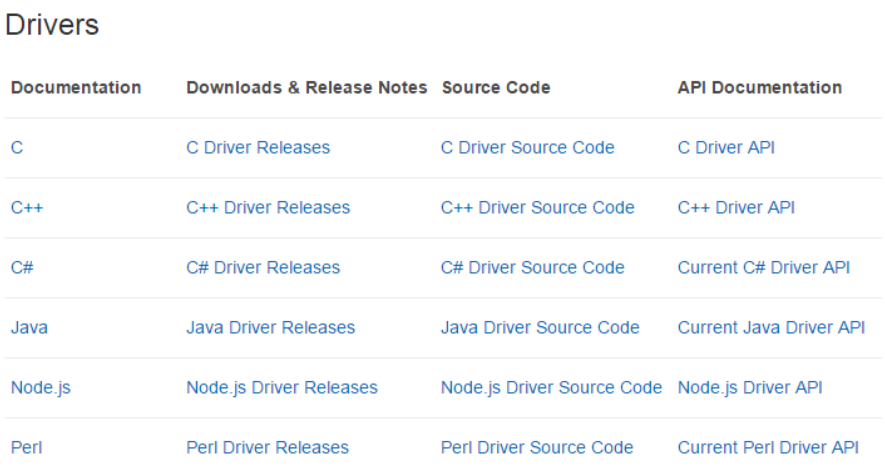


Figura 22 Página MongoDB docs

d. En la siguiente ventana que se muestra, aparece una grilla listando todos los drivers por lenguaje. Nos ubicamos en la columna 'Source Code' y hacemos click en 'Java Driver Source Code', como se ven la Figura 23 Página MongoDB Drivers.



Drivers

Documentation	Downloads & Release Notes	Source Code	API Documentation
C	C Driver Releases	C Driver Source Code	C Driver API
C++	C++ Driver Releases	C++ Driver Source Code	C++ Driver API
C#	C# Driver Releases	C# Driver Source Code	Current C# Driver API
Java	Java Driver Releases	Java Driver Source Code	Current Java Driver API
Node.js	Node.js Driver Releases	Node.js Driver Source Code	Node.js Driver API
Perl	Perl Driver Releases	Perl Driver Source Code	Current Perl Driver API

Figura 23 Página MongoDB Drivers

e. Esta opción nos redirecciona a la página de 'github.com'. Hacemos click en 'https://mongodb.github.io/mongo-java-driver'.

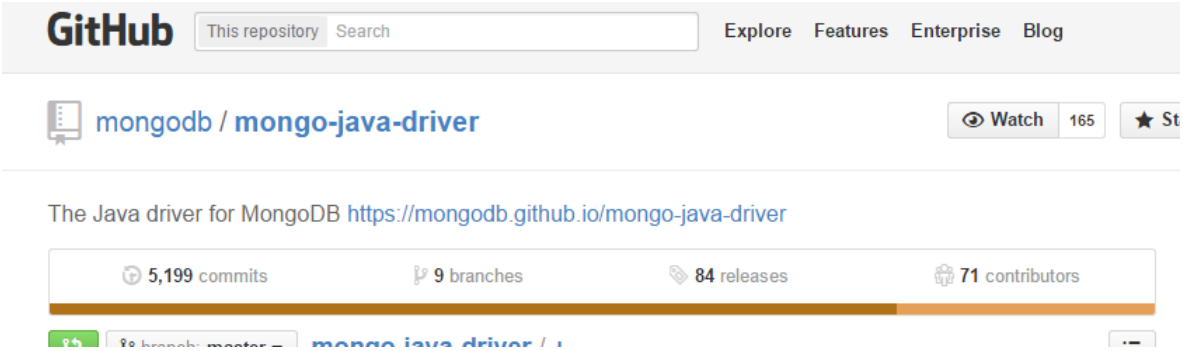


Figura 24 Página github.com

f. Como se aprecia en la siguiente ventana que se dibuja, en la parte inferior de ella nos cercioramos de que en el primer desplegable esté seleccionada la opción 'mongo-java-driver' y en el segundo desplegable esté seleccionada la opción '3.0.0'. Hacemos click en 'DOWNLOAD'.

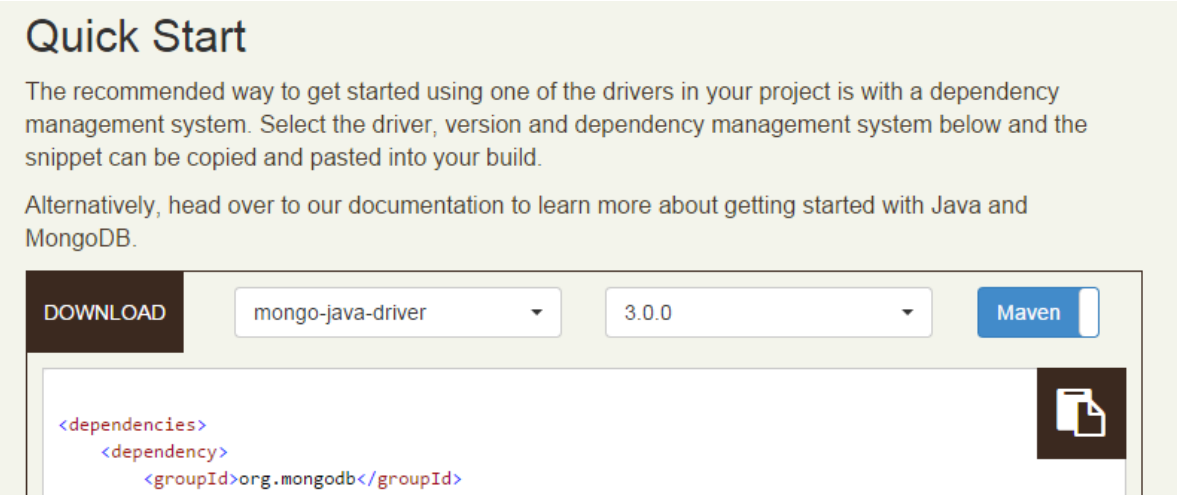


Figura 25 Página MongoDB Selección de driver

g. En la nueva ventana que se abre, se visualiza el repositorio que contiene los drivers de JAVA. Como se ve en la hacemos click en 'mongodb-driver-3.0.0.jar' e inmediatamente comienza la descarga del driver.

Index of /repositories/releases/org/mongodb/mongo

Name	Last Modified	Size	Description
Parent Directory			
mongo-java-driver-3.0.0-javadoc.jar	Tue Mar 31 14:15:30 CDT 2015	1994272	
mongo-java-driver-3.0.0-javadoc.jar.asc	Tue Mar 31 14:15:27 CDT 2015	475	
mongo-java-driver-3.0.0-javadoc.jar.asc.md5	Tue Mar 31 14:15:27 CDT 2015	32	
mongo-java-driver-3.0.0-javadoc.jar.asc.sha1	Tue Mar 31 14:15:27 CDT 2015	40	
mongo-java-driver-3.0.0-javadoc.jar.md5	Tue Mar 31 14:15:31 CDT 2015	32	
mongo-java-driver-3.0.0-javadoc.jar.sha1	Tue Mar 31 14:15:31 CDT 2015	40	
mongo-java-driver-3.0.0-sources.jar	Tue Mar 31 14:15:26 CDT 2015	919124	
mongo-java-driver-3.0.0-sources.jar.asc	Tue Mar 31 14:15:28 CDT 2015	475	
mongo-java-driver-3.0.0-sources.jar.asc.md5	Tue Mar 31 14:15:29 CDT 2015	32	
mongo-java-driver-3.0.0-sources.jar.asc.sha1	Tue Mar 31 14:15:28 CDT 2015	40	
mongo-java-driver-3.0.0-sources.jar.md5	Tue Mar 31 14:15:27 CDT 2015	32	
mongo-java-driver-3.0.0-sources.jar.sha1	Tue Mar 31 14:15:26 CDT 2015	40	
mongo-java-driver-3.0.0.jar	Tue Mar 31 14:15:22 CDT 2015	1278601	
mongo-java-driver-3.0.0.jar.asc	Tue Mar 31 14:15:24 CDT 2015	475	

Figura 26 Repositorio drivers

Este driver lo guardamos en una carpeta que llamaremos 'drivers' que va a estar ubicada dentro de la carpeta 'mongodb'.

2. Agregar la librería descargada a Netbeans

Para agregar la librería a Netbeans se van a tener en cuenta los siguientes pasos.

a. Abrimos Netbeans, hacemos click en la pestaña 'Tools' (Herramientas), después click en 'Libraries'.

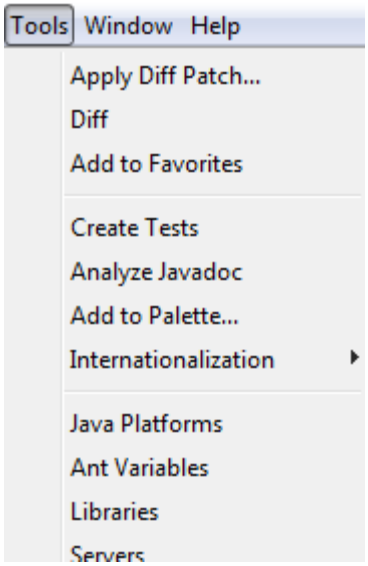


Figura 27 Pestaña herramientas

b. A continuación se abre una ventana. Hacemos click en 'New Library…' y creamos una nueva librería que llamaremos 'mongoDB'. Hacemos click en el botón 'OK'.

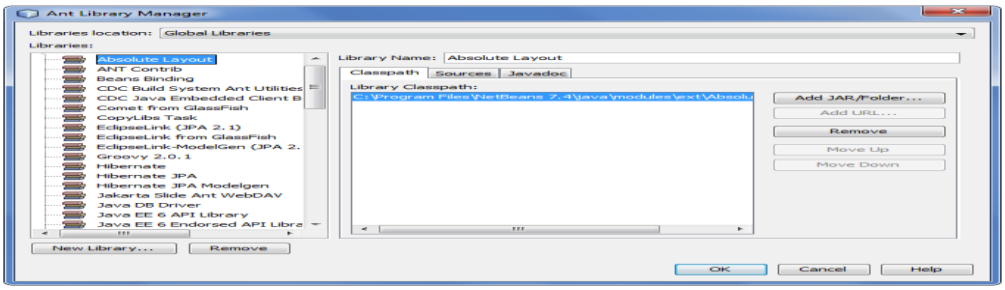


Figura 28 Ventana 'Gestión de librerías' - crear librería

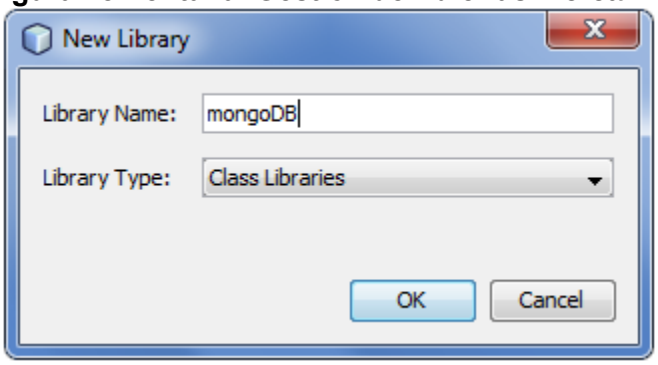
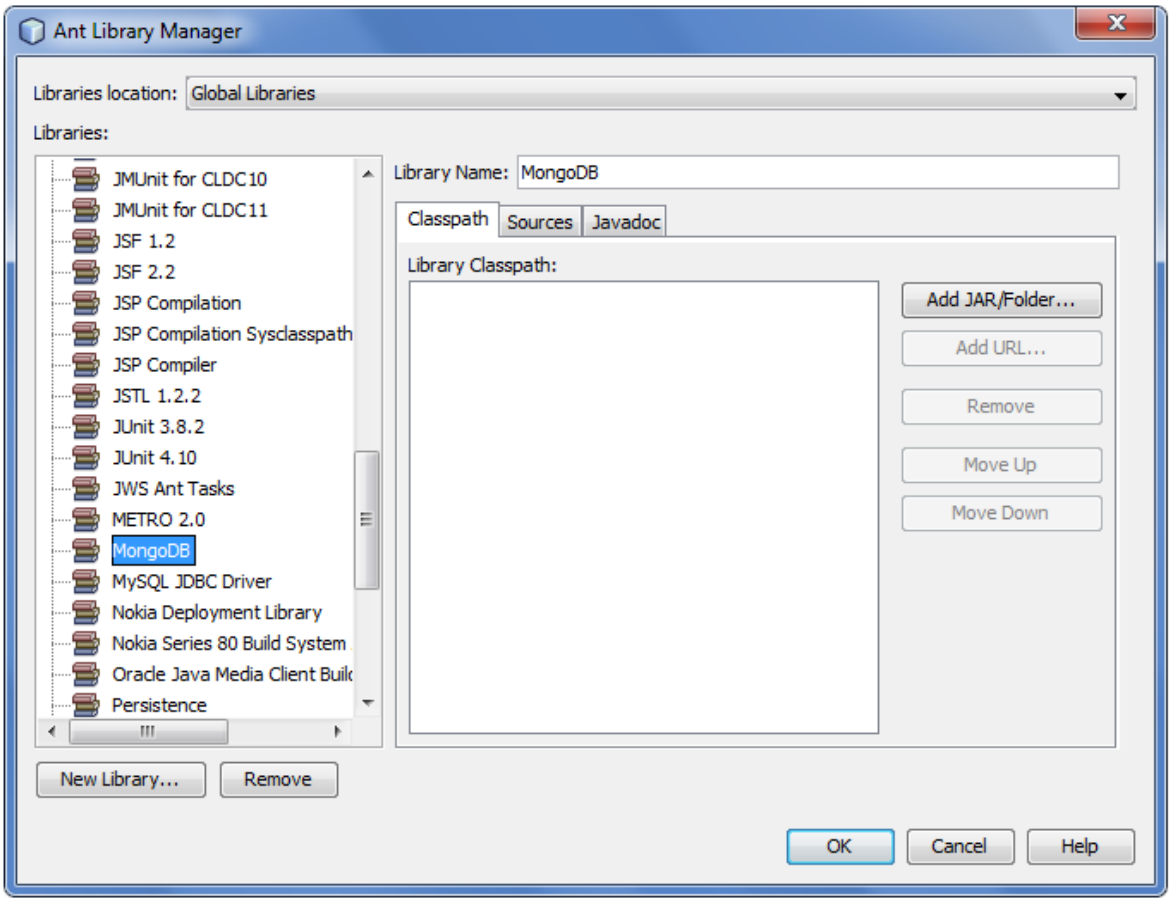
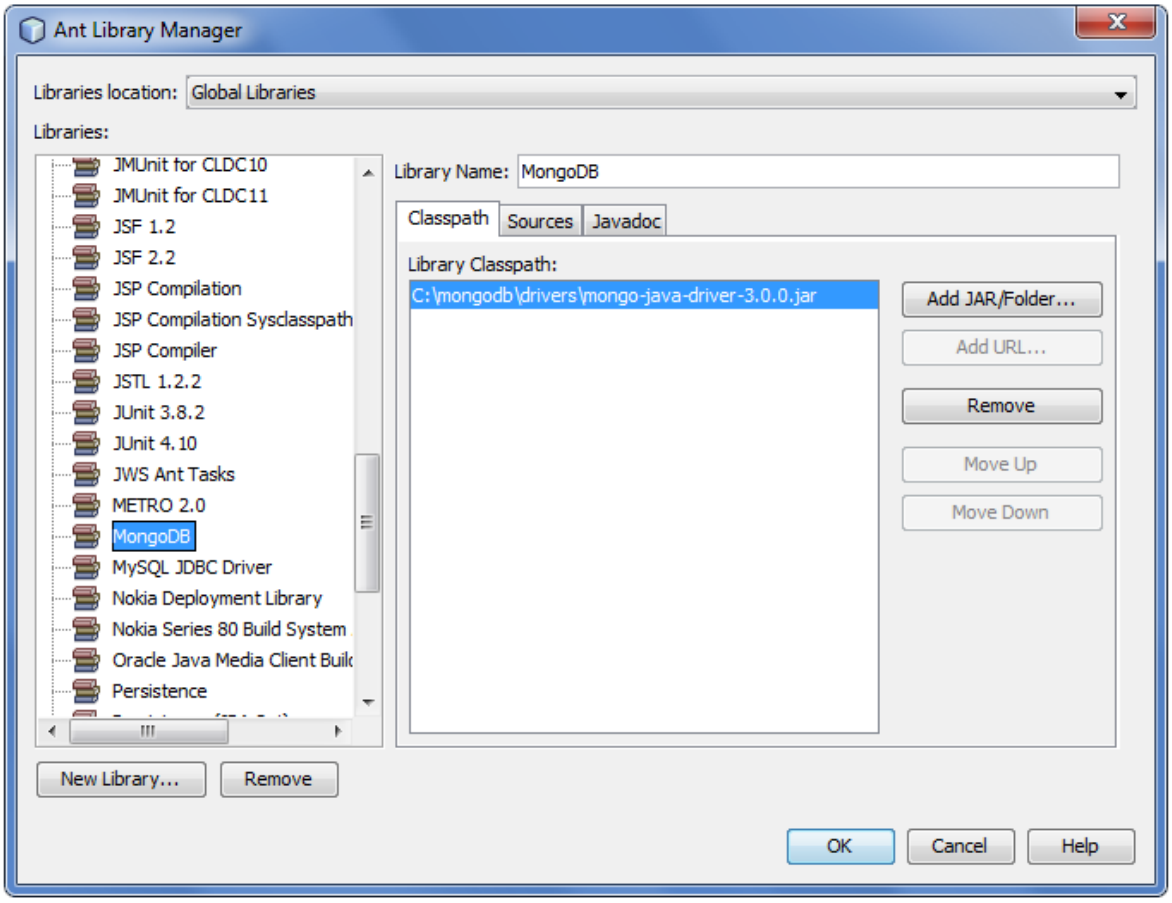


Figura 29 Ventana 'Nueva Librería'

c. Luego buscamos la nueva librería en el listado de librerías y la seleccionamos haciendo click sobre ella, luego hacemos click en el botón ‘Add JAR/Folder…’.



d. En el explorador de archivos que se abre buscamos el driver que descargamos y lo adjuntamos. El driver ya queda relacionado a la librería. Hacemos click en el botón ‘OK’.



3. Conectarse a MongoDB desde JAVA:

Para conectarse a una base de datos y ejecutar sentencias, hay que tener activo el servicio de MongoDB, que se explica con detalle al inicio del procedimiento de esta guía (Iniciar MongoDB.). Se empieza por crear un proyecto JAVA en Netbeans. El código que se explica va dentro de un método main en la clase que se crea por defecto en el nuevo proyecto.

Para conectarse al motor se declara un objeto de tipo 'MongoClient' que se va a llamar 'mongo'. Luego de esto se instancia el objeto creado, que solicita dos parámetros. El primer parámetro es la dirección del servidor (en este caso es "localhost") y el segundo es el número del puerto.

Instrucción

```
MongoClient mongo = null;
mongo = new MongoClient("localhost", 27017);
```

4. Conectarse a una base de datos o crearla:

Se declara un objeto de tipo 'DB' que se va a llamar 'db'. Para instanciar el objeto se utiliza la función '`getDB()`' que se invoca del objeto 'mongo'. El método '`getDB()`' exige un parámetro que corresponde al nombre de la base de datos. Si la base de datos no existe, la crea. La base de datos se va a llamar 'ejemplo'.

Instrucción

```
DB db = mongo.getDB("ejemplo");
```

5. Crear una colección

Se declara un objeto de tipo 'DBCollection' y se instancia con el método '`getCollection()`' que se invoca del objeto 'db'. Éste método pide como parámetro el nombre de la colección. Si la colección no existe, se crea. En este caso se va a crear una colección que se va a llamar 'usuario'.

Instrucción

```
DBCollection collection = db.getCollection("usuario");
```

6. Crear un documento e insertarle campos:

Las estructuras básicas de los documentos se representan con objetos de tipo 'BasicDBObject'. Si se desean insertar campos en uno o varios documentos, se recurre a los métodos que brinda la clase 'BasicDBObject'.
Creamos 5 documentos como ejemplo. A estos documentos le agregaremos cinco campos con sus valores respectivos.

- Nombre (Cadena de texto)
- Apellidos (Cadena de texto)
- Cedula (Numérico)
- Edad (Numérico)

Para realizar esta inserción se hace uso de la función '`put()`' que requiere dos parámetros, el primero es el nombre del campo, y el segundo el valor que contendrá el campo.
Instrucción

```
BasicDBObject document1 = new BasicDBObject();
document1.put("nombre", "Jacinta");
document1.put("apellidos", "Roma Estupiñan");
document1.put("documento", 10419854662);
document1.put("edad", 25);

BasicDBObject document2 = new BasicDBObject();
document2.put("nombre", "Pablo");
document2.put("apellidos", "Mora");
document2.put("documento", 52039232);
document2.put("edad", 45);

BasicDBObject document3 = new BasicDBObject();
```

```
document3.put("nombre", "Gabriela");
document3.put("apellidos", "Corrales");
document3.put("documento", 1031189562);
document3.put("edad", 18);

BasicDBObject document4 = new BasicDBObject();
document4.put("nombre", "Pablo");
document4.put("apellidos", "Rodríguez Roncancio");
document4.put("documento ", 104198546);
document4.put("edad", 20);

BasicDBObject document5 = new BasicDBObject();
document5.put("nombre", "Martina");
document5.put("apellidos", "Rodríguez");
document5.put("documento", 980205569);
document5.put("edad", 17);
```

Ahora se utiliza la función '`insert()`' del objeto que representa la colección para poder almacenar los documentos.

Instrucción

```
collection.insert(document1);
collection.insert(document2);
collection.insert(document3);
collection.insert(document4);
collection.insert(document5);
```

7. Consultar documentos

Para visualizar todos los documentos, se almacenan en un cursor que se llena utilizando la función '`find()`' del objeto que representa a la colección. Este cursor es un objeto de tipo 'DBcursor'. Para recorrer el cursor se hace uso de un ciclo 'while' que realizará las iteraciones sobre el cursor, utilizando el método '`hasNext()`'. La instrucción quedaría así.

Instrucción

```
System.out.println("Listar los registros de la tabla: ");
DBCursor cur = table.find();
while (cur.hasNext()) {
   System.out.println(" - " +
     cur.next().get("nombre") + " " +
     cur.curr().get("apellidos") + " -- " +
     cur.curr().get("documento") + " años (" +
     cur.curr().get("edad") + ")"
   );
}
System.out.println();
```

Resultado

```
Listar los registros de la tabla:
 - Jacinta Roma Estupiñan -- 104198546 años (25)
 - Pablo Mora -- 52039232 años (45)
 - Gabriela Corrales -- 1031189562 años (18)
 - Pablo Rodríguez Roncancio -- 104198546 años (20)
 - Martina Rodríguez -- 980205569 años (17)
```

Figura 30 Resultado consulta documentos insertados

8. Actualizar documentos

Para actualizar los campos de algún documento en particular aplicando alguna condición, se requiere crear dos objetos de tipo 'BasicDBObject'. Como ejemplo se van a modificar las edades a 50 para los usuarios que se llamen 'Pablo'.

En el primer objeto se va a pasar como parámetros el operador de actualización y el campo con el valor de reemplazo.

Instrucción

```
BasicDBObject updateEdad = new BasicDBObject();
updateEdad.append("$set",new BasicDBObject().append("edad",50));
```

En el segundo objeto va la condición.

Instrucción

```
BasicDBObject searchById = new BasicDBObject();
searchById.append("nombre", "Pablo");
```

Seguido a esto se invoca el método '`updateMulti()`' que pide como parámetros los dos objetos que se crearon, así.

Instrucción

```
collection.updateMulti(searchById, updateEdad);
```

Al imprimir el resultado quedaría así.

Resultado

```
Listar los registros de la tabla:
 - Jacinta Roma Estupiñan -- 104198546 años (25)
 - Pablo Mora -- 52039232 años (50)
 - Gabriela Corrales -- 1031189562 años (18)
 - Pablo Rodríguez Roncancio -- 104198546 años (50)
 - Martina Rodríguez -- 980205569 años (17)
```

Figura 31 Resultado consulta documentos modificados

9. Eliminar Documentos:

Para eliminar documentos se utiliza la función '`remove()`' del objeto que equivale a una colección. Esta función pide como parámetro un objeto de tipo '`BasicDBObject`' que contiene el nombre del campo y la condición. Como ejemplo se van eliminar los usuarios cuya edad sea menor a 30.

Instrucción

```
BasicDBObject query = new BasicDBObject();
query.put("edad", new BasicDBObject("$lt", 30));
collection.remove(query);
```

Al imprimir los documentos de la colección 'usuario', este sería el resultado:

Resultado

```
Listar los registros de la tabla:
 - Pablo Mora -- 52039232 años (50)
 - Pablo Rodríguez Roncancio -- 104198546 años (50)
```

Figura 32 Resultado consulta al eliminar documentos

REFERENCIAS

[1] Yohan Graterol. MongoDB en español: T1, El principio. Autoedición. 29p. 2014

1. [2] Db engines – Base de conocimientos de sistemas de gestión de bases de datos relacionales y NoSQL. http://db-engines.com/en/ranking_definition

[3] MongoDB x007A0078xdesde Cero: Actualizaciones / Updates [En línea]: http://codehero.co/mongodb-desde-cero-actualizaciones-updates [Citado el 15 de Abril de 2015]