TP2 – Parte B - Punto 5

*5.1 - Codifique un método que **seleccione** los documentos de la colección **países** donde la **región** sea **Americas**. Muestre el resultado por pantalla o consola.*

```
public static void PaisesAmerica(MongoCollection<Document> paises) {
        System.out.println("Documentos con la región Americas");
        BasicDBObject query = new BasicDBObject("region", "Americas");
        FindIterable<Document> docs = paises.find(query);
        for (Document doc: docs) {
            System.out.println(doc);
        }
    }
```

*5.2. Codifique un método que **seleccione** los documentos de la colección **países** donde la **región** sea **Americas** y la **población** sea mayor a **100000000**. Muestre el resultado por pantalla o consola.*

```
public static void PaisesAmericasConPoblacionMayorA100M(MongoCollection<Document>
paises) {
        System.out.println("Documentos con la región Americas y población mayor a
100000000");
        BasicDBObject query = new BasicDBObject("region", "Americas").append("poblacion",
new BasicDBObject("$gt", 100000000));
        FindIterable<Document> docs = paises.find(query);
        for (Document doc: docs) {
            System.out.println(doc);
        }
        System.out.println("-------------------------------------------------");
    }
```

*5.3. Codifique un método que **seleccione** los documentos de la colección **países** donde la **región** sea **distinto** de **Africa**. (investigue **$ne**). Muestre el resultado por pantalla o consola.*

```
public static void PaisesNoAfrica(MongoCollection<Document> paises) {
        System.out.println("Documentos con la región diferente a Africa");
        BasicDBObject query = new BasicDBObject("region", new BasicDBObject("$ne",
"Africa"));
        FindIterable<Document> docs = paises.find(query);
        for (Document doc : docs) {
            System.out.println(doc);
        }
        System.out.println("-------------------------------------------------");
    }
```

*5.4. Codifique un método que **actualice** el documento de la colección **países** donde el name sea Egypt, cambiando el name a "Egipto" y la población a 95000000*

```
public static void ActualizarEgipto(MongoCollection<Document> paises) {
        System.out.println("Actualizar Egipto");
        BasicDBObject updatedDocument = new BasicDBObject("nombre", "Egypt");
        FindIterable<Document> docs = paises.find(updatedDocument);
```

```
        paises.updateOne(updatedDocument, new BasicDBObject("$set", new
BasicDBObject("nombre", "Egipto")));
                        paises.updateOne(updatedDocument, new BasicDBObject("$set", new
                    BasicDBObject("superficie", 95000000)));

        System.out.println("-------------------------------------------------");
    }
```

*5.5. Codifique un método que **elimine** el documento de la colección **países** donde el **código** del país sea **258***

```
public static void BorrarPais(MongoCollection<Document> paises, int num){
        paises.deleteOne(new BasicDBObject("codigoPais", num));
        System.out.println("Se eliminó el país: "+ num);
    }
```

*5.6. Describa que sucede al ejecutar el método **drop()** sobre una colección y sobre una base de datos.*

Al ejecutar el comando drop() sobre una base de datos, se elimina dicha base de datos.

Ej: db.dropDatabase()

De igual manera, al ejecutar drop() sobre una colección, se elimina esa colección.

*5.7. Codifique un método que seleccione los documentos de la colección **países** cuya **población** sea mayor a 50000000 y menor a 150000000. Muestre el resultado por pantalla o consola.*

```
public static void PaisesConPoblacionEntre(MongoCollection<Document> paises,int
limiteMenor, int LimiteMayor ) {
        System.out.println("Paises con población entre" + limiteMenor + " y " + LimiteMayor);
        BasicDBObject query = new BasicDBObject("poblacion", new BasicDBObject("$gt",
limiteMenor)).append("poblacion", new BasicDBObject("$lt", LimiteMayor));
        FindIterable<Document> docs = paises.find(query);
        for (Document doc : docs) {
            System.out.println(doc);
        }
        System.out.println("-------------------------------------------------");
    }
```

Llamada = PaisesConPoblacionEntre(paises, 50000000, 150000000);

*5.8. Codifique un método que seleccione los documentos de la colección **países** ordenados por nombre (name) en forma Ascendente. **sort().** Muestre el resultado por pantalla o consola.*

```
public static void OrdenarPaisesAsc(MongoCollection<Document> paises){
   System.out.println("Paises ordenados por nombre de forma ascendente");
   FindIterable<Document> docs = paises.find().sort(new BasicDBObject("nombre", -1));
   for (Document doc : docs) {
         System.out.println(doc);
     }
   }
```

*5.9. Describa que sucede al ejecutar el método **skip()** sobre una colección. Ejemplifique con la colección **países**.*

Mediante skip() podemos omitir resultados y no recuperar ciertos documentos.

Ej:
```
FindIterable<Document> busqueda = paises.find().skip(2);
```

El parámetro dentro de skip es el número de elementos que queremos omitir antes de devolver el contenido.

*5.10. Describa y ejemplifique como el uso de expresiones regulares en Mongo puede reemplazar el uso de la cláusula **LIKE** de SQL.*

En SQL, nosotros podemos utilizar el operador **LIKE** para buscar coincidencias en donde la celda tenga una porción de una cadena.

Las consultas MongoDB like se resuelven mediante expresiones regulares. Lo que realizaremos mediante la siguinte sintaxis:

```
db.coleccion.find({campo:expresión_regular});
```

De esta forma tendremos las siguientes similitudes con los patrones LIKE.

cadena% /^cadena/

%cadena% /cadena/

%cadena /cadena$/

De esta forma la sentencia para realizar consultas MongoDB like será la siguiente:

```
cursor = db.ciudades.find({ciudad:/^M/});
```

En este caso hemos realizado un filtro LIKE de ciudades que empiecen por M.

A la hora de utilizar las expresiones regulares, los índices solo se utilizarán de forma eficiente cuando utilizemos la forma /^cadena/

Otros podrían ser ciudades que contengan una «r»:

```
cursor = db.ciudades.find({ciudad:/r/});
```

O ciudades que acaben en d:

```
cursor = db.ciudades.find({ciudad:/d$/});
```

Ya solo nos quedará recorrer con un cursor el resultado para volcar los documentos devueltos por la expresión regular:

```
while (cursor.hasNext()) {
  printjson(cursor.next());
}
```

Podemos buscar, de nuevo, aquellos que tenga “ra” con la siguiente consulta:

```
db.clientes.find({
   "Nombre": /.*ra.*/
 }
);
```

Si queremos que sea insensitivo a mayúsculas (que devuelva aquellos en donde diga ra, Ra, rA o RA) entonces modificaríamos la consulta así:

```
db.clientes.find({
   "Nombre": /.*ra.*/i
 }
);
```

*5.11. Cree un nuevo índice para la colección países asignando el campo código como índice. Investigue **createIndex**())*

```
public static void crearIndice(MongoCollection<Document> paises){
   paises.createIndex(new BasicDBObject("codigoPais",1));
   System.out.println("Se agregó codigoPais como índice");
   }
```

*5.12. Describa como se realiza un backup de la base de datos mongo **países_db**.*

Para **realizar** la copia de seguridad de la **base de datos** de **MongoDB**, **se** utiliza la utilidad mongodump, que **se** encuentra en el directorio bin. Esta utilidad **realiza** la copia de seguridad de todos los **datos** en la carpeta «dump» de la ubicación predeterminada /bin/dump.

Otra forma…

Para realizar backup de una base de datos MongoDB, debes usar la utilidad mongodump, que tiene disponible en todos los planes de hosting. Para ello se utiliza el sig comando:

```
mongodump -h host -u usuario -p passsword --authenticationDatabase base_datos -d base_datos -o dir
```

El comando almacenará los contenidos de las colecciones del backup en un directorio con el nombre **dir**

Posteriormente este contenido puede ser restaurado con mongorestore.

```
mongorestore -h host -u usuario -p passsword --authenticationDatabase base_datos -d base_datos -o dir
```

Este comando realiza la tarea inversa a mongodump. Lee el contenido en formato de mongodump que exista en el directorio **dir** y lo restaura en la base de datos **base_datos**