MongoDB (CHULETA)

Base de Datos orientada a objetos

Guia de Comandos de MobgoDB

Rafael Gamez Diaz

Comandos en la consola MongoDB

1. Ver lista de las bases de datos existentes:

```
> show dbs
```

2. Usar una base de datos:

```
> use nombre_db
```

3. Base de datos actual:

```
> db
```

4. Insertar un elemento en una colección, si no existe la crea: base de datos:

```
> db.products.insert({"name": "Laptop DELL"})
```

5. Ver las colecciones que contiene una base de datos:

```
> show collections
```

6. Crear colección:

```
> show createCollection("users")
```

7. Eliminar colección:

```
> db.sellers.drop()
```

8. Ver el contenido de una colección:

```
> db.products.find()
```

9. Ver el contenido de una colección en forma las visual:

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

10. Insersion de multiple líneas:

```
> db.products.insert([{"name": "Laptop DELL"}, {"name": "Laptop Lenovo",
    "price": 999.99}])
```

11. Buscar por el valor de un campo

```
> db.products.find({"name": "Latitude"})
```

12. Buscar por un contenido dentro de un campo mayor que cierto valor numérico $gt:

```
> db.products.find({name: { $gt: 20}})
```

13. Devuelve solo el primer resultado del query

```
> db.products.findOne({"name": "Latitude"})
```

14. Obtener campos especificos

```
> db.products.find({"name": "Latitude"},{"name":1}).
```

15. Obtener campos especificos de todos los registros

```
> db.products.find({},{"name":1})
```

16. Si no quiero un campo específico

```
> db.products.find({},{"_id":0})
```

17. Ordenando por un campo (1 Ascendente, -1 Descendente)

```
> db.products.find({},{"_id": 0}).sort({"name":1}).pretty()
```

```
> db.products.find({},{"_id": 0}).sort({"name":-1}).pretty()
```

18. Obtener un número limitado de resultados

```
> db.products.find({},{"_id": 0}).limit(3).pretty()
```

19. Contar registros

```
> db.products.find().count()
```

20. Utilizar funciones como el forEach

```
> db.products.find().forEach(product => print("Product Name:" + product.name))
```

21. Actualizar un documento, a la izq el objeto que se busca reemplazar, buscado por el nombre. Si existe "price" se actualiza, si no existe se agrega al documento.

```
> db.products.update({"name": "Laptop Acer"}, {$set: {"price": 200}})
```

22. El parametro upsert: true, permite que si el elemento no existe, el update lo creará

```
> db.products.update({"name": "Tablet Samsung"},{$set: {"description": "Blab la bla"}},{upsert: true})
```

23. Actualizar datos numéricos, se incrementa el precio en 0.01

```
> db.products.update({"name":"Desktop Lenovo"},{$inc: {"price": 0.01}})
```

24. Renombrar un nombre de campo

```
> db.products.update({"name":"Desktop Lenovo"},{$rename: {"name": "nombre"}})
```

25. Actualizar el nombre de un campo en todos los documentos

```
> db.products.update({},{$rename: {"name": "user"}}, {multi:true})
```

26. Eliminar todos los productos que cumplan la condición

```
> db.products.remove({"user": "Laptop Acer"})
```

27. Eliminar todos los documentos dentro de la colección

```
> db.products.remove({})
```

28. Eliminar todos los documentos diferentes del calor del campo de la condición

```
> db.products.find({user: { $ne: "Laptop Latitude DELL"}})
```

29. Se combinan dos condiciones, user diferente de "Rafael" y age > 30. Otros operadores son $gte (mayor e igual), $lt (menor que), $lte (menor o igual)

```
> db.products.find({user: { $ne: "Rafael"}, age: { $gt : 100}})
```

30. Combinando dos condicionales

```
> db.products.find({price: { $gt : 500, $lt : 1001}},{price:1})
```

31. Cursores. Veamos varias cosas que podemos hacer con ellos

```
> var cursor = db.products.find()
```

```
> cursor.forEach( function (item) { print(item.nombre_campo) } )
```

```
> cursor.forEach( function (item) { item.nombre_campo = 100;
db.products.save(item) } )
```

32. Adicionar valores a un array que esté dentro de un documento. En este caso se agrega el valor 4 al array que está dentro del documento. Este metodo solo agregará el 4 una vez, si no existe. Si se desea continuar agregando el mismo valor debemos cambiar el método $addToSet a $push.

```
> var arreglo = [1,2,3]
> var usuario = {nombre: "Rafael", valores: arreglo}
> db.products.insert(usuario)
> db.products.update({}, {$addToSet : {valores: 4}})
> db.products.find().pretty()
        {
                "_id" : ObjectId("5e9f95f6d5245b5a7803cea4"),
                "nombre" : "Rafael",
                "valores" : [
                        1,
                        2,
                        3,
                        4
                ]
        }
```

33. Para agregar una lista de valores dentro del array del documento

```
> db.products.update({}, {$push : {valores: {$each : [5,6,7,8]} }})
```

34. Para insertar en una posición determinada del array

```
> db.products.update({}, {$push : {valores: {$each : [100, 101], $position : 3} }})
```

35. Para insertar y ordenar el array de forma ascendente (ascendente 1, descendente -1). La segunda linea de comandos ordena descendentemente y al pasarle un array vacio no se insertó ningun elemento

```
> db.products.update({}, {$push : {valores: {$each : [34, 90], $sort : 1} }})
```

```
> db.products.update({}, {$push : {valores: {$each : [], $sort : -1} }})
```

36. Para eliminar elementos de un array (elimina todas las coincidencias)

```
> db.products.update({}, {$pull : {valores: 101 }})
```

37. Para eliminar elementos de un array según una condición

```
> db.products.update({}, {$pull : {valores: {$gt: 33} }})
```

38. Para eliminar elementos específicos de un array

```
> db.products.update({}, {$pullAll : {valores: [1,3,5] }})
```

39. Para eliminar elementos específicos de un array

```
> db.products.update({}, {$pullAll : {valores: [1,3,5] }})
```

40. Para obtener un subconjunto de elementos de un array

```
> db.products.find({}, {_id:0 , valores: { $slice: 3 } })
```

41. Para obtener todos los documentos que contengan un array que inlcuya alguno de los valores de la condicion $in. Aquí 3345 y 3430, pueden estar en dos arrays de dos documentos diferntes

```
> db.products.find({valores: { $in: [3345,3430] } })
```

42. Agrupar por un campo

```
> db.products.aggregate([ {$group: { "item" } } ]
```

43. Contar los agrupados por un campo

```
> db.products.aggregate([ {$group: { _id: "$item", "repetidos": {$sum:1} } } ])
```

44. Sumar un campor de los agrupados por un campo

```
> db.products.aggregate([ {$group: { _id: "$item", "repetidos": {$sum:1}, "suma_valor": {$sum: "$valor"} } } ])
```

45. Promedia los agrupados por un campo

```
> db.products.aggregate([ {$group: { _id: "$item", "promedio": {$avg: "$valor"} } } ])
```

46. Buscar por un contenido dentro de un campo (Expresiones Regulares):

```
> db.products.find({"name": /Lati/})
```

```
> db.products.find({valor: /^roci/})
```

```
> db.products.find({valor: /.com$/})
```

Exportar e Importar una base de datos completa

1 . Exportando con ***mongodump***

Debemos estar duera del Shell de mongo, es decir, en una terminal normal. Nos ubicamos dentro de la carpeta donde queremos respaldar la base de datos.

```
$> mongodump --db nombre_bd
```

Donde hemos especificado el nombre de la base de datos que vamos a exportar. Una vez ejecutado el proyecto, Mongo va a crear una carpeta llamada dump, dentro de la ubicación donde nos encontrabamos en la consola cuando ejecutamos el comando. Dentro de dicha carpeta Mongo creará dos archivos por cada colección que tengamos.

2 . Importanto con ***mongorestore***

Importamos la base de datos, podemos asignarle un nuevo nombre si deseamos.

```
$> mongorestore --db nombre_que_se_dara_a_la_bd carpeta_donde_esta/
```