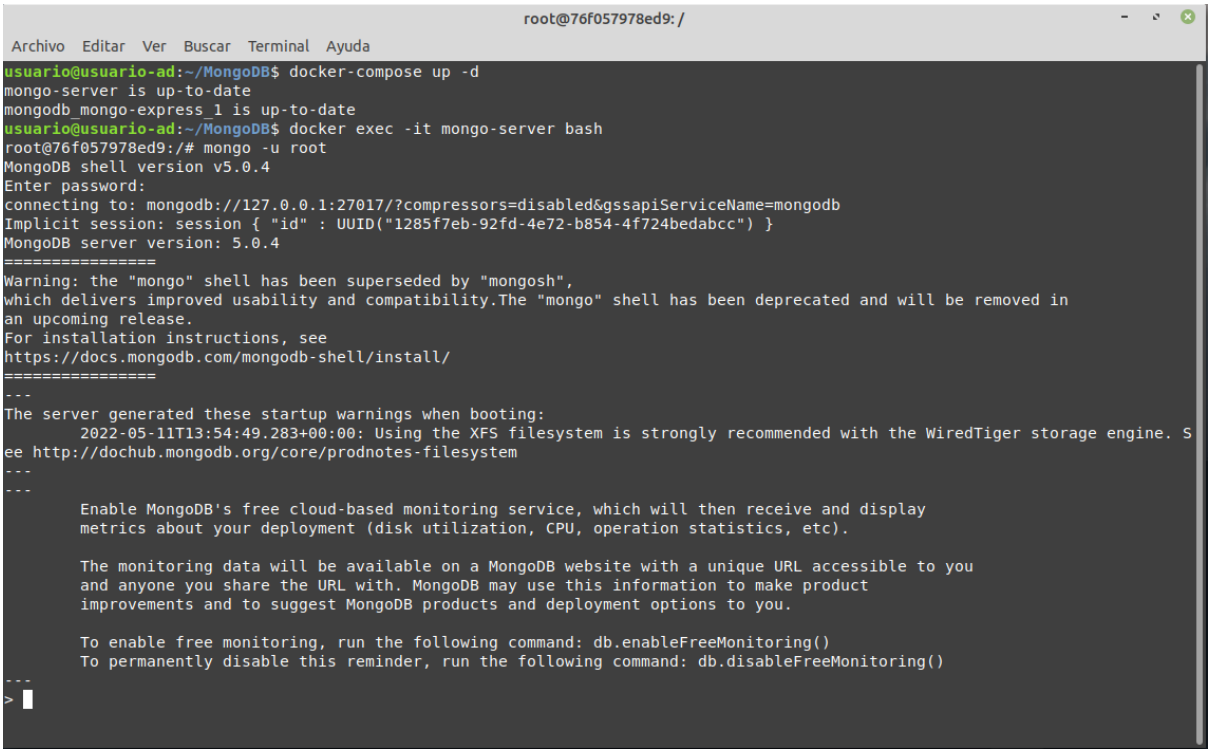

- Comando para obtener cual es la base de datos actual de trabajo

```
> db
test
```

- Comando para abrir otra base de datos de trabajo (si no existe la crea):

```
> use centroDB
switched to db centroDB
```

5.1 Insertar

- Ya hemos visto que para insertar un documento en una colección, se usa el método **insertOne** de la colección. Hay otros métodos para insertar:
  - **insertMany**: para insertar varios documentos en la misma instrucción.
  - **insert**: permite insertar uno o varios documentos.
  - **save**: permite insertar un documento o modificarlo si su _id ya existe.
- Actualmente **se recomienda no usar** insert y save.

```
> db.alumnos.insertOne(
... {
... _id:101,
... nombre: "Javier",
... apellidos: "Díaz Iturbe",
... curso: "DAW1",
... nota_media:6.7
... }
... )
```

```
> db.alumnos.insertMany([
... {_id: 102, nombre: "Raquel", apellidos: "San Román Avilés", curso: "DAW1", nota
... {_id: 201, nombre: "Ana", apellidos: "Pinta de la, Ruiz", curso: "DAW2", nota_m
... {_id: 202, nombre: "David", apellidos: "Valle Pérez", curso: "DAW2", nota_media
... ])
```

5.2 Actualizar documentos

- Vamos a ver dos ejemplos de modificación, el primero usa **updateOne** para modificar un solo documento y el otro usa **updateMany** para modificar varios documentos.

Ejemplo 5: modificar la nota del alumno con id 101 para que sea 6,80.

```
> db.alumnos.updateOne(
... { _id: 101 }, //Filtro
... { $set: { nota_media: 6.8} } //Modificación (SET)
... )
```

Ejemplo 6: incrementar en un punto la nota de los alumnos de DAW2.

```
> db.alumnos.updateMany(
... { curso: "DAW2" },
... { $inc: {nota_media: 1} }
... )
```

- Operadores de actualización

Fields

Name	Description
`$currentDate`	Sets the value of a field to current date, either as a Date or a Timestamp.
`$inc`	Increments the value of the field by the specified amount.
`$min`	Only updates the field if the specified value is less than the existing field value.
`$max`	Only updates the field if the specified value is greater than the existing field value.
`$mul`	Multiplies the value of the field by the specified amount.
`$rename`	Renames a field.
`$set`	Sets the value of a field in a document.
`$setOnInsert`	Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
`$unset`	Removes the specified field from a document.

Array

Operators

Name	Description
`$`	Acts as a placeholder to update the first element that matches the query condition.
`$[]`	Acts as a placeholder to update all elements in an array for the documents that match the query condition.
`$[<identifier>]`	Acts as a placeholder to update all elements that match the `arrayFilters` condition for the documents that match the query condition.
`$addToSet`	Adds elements to an array only if they do not already exist in the set.
`$pop`	Removes the first or last item of an array.
`$pull`	Removes all array elements that match a specified query.
`$push`	Adds an item to an array.
`$pullAll`	Removes all matching values from an array.

Modifiers

Name	Description
`$each`	Modifies the `$push` and `$addToSet` operators to append multiple items for array updates.
`$position`	Modifies the `$push` operator to specify the position in the array to add elements.
`$slice`	Modifies the `$push` operator to limit the size of updated arrays.
`$sort`	Modifies the `$push` operator to reorder documents stored in an array.

Bitwise

Name	Description
`$bit`	Performs bitwise `AND`, `OR`, and `XOR` updates of integer values.

5.3 Consultar documentos

- Vamos a ver cuatro ejemplos de consultas. Más adelante profundizaremos mucho más en la sintaxis de las consultas.

Ejemplo 7: obtener todos los datos de los alumnos de DAM1.

```
> db.alumnos.find( { curso: "DAM1" } )
```

Ejemplo 8: obtener nombre y apellidos de todos los alumnos

```
> db.alumnos.find(  {}, { _id:0, nombre: 1, apellidos: 1}  )
```

Ejemplo 9: obtener todos los datos de los alumnos de DAM1, excepto el id.

```
> db.alumnos.find(  {curso: "DAM1"}, { _id:0}  )
```

Ejemplo 10: obtener todos los datos de los alumnos de DAW1 con nota media superior o igual a 6.

```
> db.alumnos.find(  {curso: "DAW1", nota_media : {$gte : 6 }}  )
```

5.4 Eliminar documentos

- Vamos a ver dos ejemplos de eliminación, en los dos vamos a usar **deleteMany** para eliminar varios documentos.
- En el primero podríamos usar también **deleteOne**. El método deleteOne elimina sólo el primero de los eliminables.

Ejemplo 11: Eliminar el documento correspondiente a la alumna Marta Arribas Prieto.

```
> db.alumnos.deleteMany( {nombre : "Marta", apellidos: "Arribas Prieto"} )
{ "acknowledged" : true, "deletedCount" : 1 }
```

Ejemplo 12: eliminar los documentos de los alumnos del curso DAM1.

```
> db.alumnos.deleteMany( { curso: "DAM1" } )
{ "acknowledged" : true, "deletedCount" : 3 }
```

MongoDB Hoja 1

1.- Inicia una conexión a MongoDB con el cliente **mongo.** Abre una nueva base de datos **geografía**.

```
> use geografia
switched to db geografia
```

2.- Crea en la base de datos geografía una colección **ccaa** donde añades inicialmente los datos de una Comunidad Autónoma:

"codigo": "01",
"nombre": "Andalucia",
"abreviatura": "AND",
"capital": {
"nombre":"Sevilla",
"habitantes":690000
},
"provincias":["Almería","Cádiz", "Córdoba", "Granada", "Huelva", "Jaén", "Malaga"],
"extension":87168,
"habitantes":8284000

```
> db.ccaa.insertOne({
... codigo: "01",
... nombre: "Andalucia",
... abreviatura: "AND",
... capital: {nombre:"Sevilla", habitantes:690000},
... provincias:["Almería","Cádiz", "Córdoba", "Granada", "Huelva", "Jaén", "Malaga"],
... extension:87168,
... habitantes:8284000})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("61a204839a1489047d6751ea")
}
```

3.- Añade a **ccaa** los datos de las siguientes CCAA. Introduce los datos como se representan en las capturas. En la segunda captura el nombre del atributo **capitas** debes introducirlo con ese texto.

```
> db.ccaa.insertMany([
... {codigo: "02",
... nombre: "Aragón",
... abreviatura: "ARG",
... capitas: {nombre: "Zaragoza", habitantes: 690000},
... provincias: ["Huesca", "Teruel", "Zaragoza"],
... habitantes: 1398000},
...
... {codigo: "03",
... nombre: "Islas Baleares",
... abreviatura: "ISB",
... provincias: [ "Baleares"],
... extension: 4992,
... habitantes: 1130000},
...
... {codigo: "04",
... nombre: "Canarias",
... abreviatura: "CAN",
... capital: {nombre: "Santa Cruz de Tenerife", habitantes: 222000},
... provincias: ["Las Palmas", "Santa Cruz de Tenerife"],
... extension: 7447,
... habitantes: 2127000,
... usohorario: "UTC+00:00"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61a204b99a1489047d6751eb"),
                ObjectId("61a204b99a1489047d6751ec"),
                ObjectId("61a204b99a1489047d6751ed")
        ]
}
```

3.- En la ccaa de código 01 establece que los habitantes son 8384000

```
> db.ccaa.updateOne({codigo: "01"}, {$set:{habitantes: 8384000}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

4.- En la ccaa de código 01 establece que la extensión es la actual más 100.

```
> db.ccaa.updateOne({codigo: "01"}, {$inc:{extension: 100}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

5.- En la ccaa de nombre Andalucia añade un atributo **usohorario** con el valor UTC+01:00.

```
> db.ccaa.updateOne({nombre: "Andalucia"}, {$set:{usohorario: "UTC=00:00"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.ccaa.find({nombre: "Andalucia"}).pretty()
{
        "_id" : ObjectId("61a204839a1489047d6751ea"),
        "codigo" : "01",
        "nombre" : "Andalucia",
        "abreviatura" : "AND",
        "capital" : {
                "nombre" : "Sevilla",
                "habitantes" : 690000
        },
        "provincias" : [
                "Almería",
                "Cádiz",
                "Córdoba",
                "Granada",
                "Huelva",
                "Jaén",
                "Malaga"
        ],
        "extension" : 87268,
        "habitantes" : 8384000,
        "usohorario" : "UTC=00:00"
}
```

6.- En la ccaa de código 01 añade la provincia Sevilla al array de provincias.

```
> db.ccaa.updateOne({codigo: "01"}, {$push:{provincias: "Sevilla"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.ccaa.find({codigo: "01"}).pretty()
{
        "_id" : ObjectId("61a20a9a9a1489047d6751ee"),
        "codigo" : "01",
        "nombre" : "Andalucia",
        "abreviatura" : "AND",
        "capital" : {
                "nombre" : "Sevilla",
                "habitantes" : 690000
        },
        "provincias" : [
                "Almería",
                "Cádiz",
                "Córdoba",
                "Granada",
                "Huelva",
                "Jaén",
                "Malaga",
                "Sevilla"
        ],
        "extension" : 87268,
        "habitantes" : 8384000,
        "usohorario" : "UTC=00:00"
}
```

7.- En la ccaa de código 01 elimina la provincia Malaga (sin acento).

8.- En la ccaa de código 01 añade la provincia Málaga (con acento).

```
> db.ccaa.updateOne({codigo: "01"}, {$pull:{provincias: "Malaga"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.ccaa.updateOne({codigo: "01"}, {$push:{provincias: "Málaga"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

9.- En la ccaa de código 01 elimina el atributo **usohorario.**

```
> db.ccaa.updateOne({codigo: "01"}, {$unset:{usohorario: "*"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.ccaa.find({codigo: "01"}).pretty()
{
        "_id" : ObjectId("61a20a9a9a1489047d6751ee"),
        "codigo" : "01",
        "nombre" : "Andalucia",
        "abreviatura" : "AND",
        "capital" : {
                "nombre" : "Sevilla",
                "habitantes" : 690000
        },
        "provincias" : [
                "Almería",
                "Cádiz",
                "Córdoba",
                "Granada",
                "Huelva",
                "Jaén",
                "Sevilla",
                "Málaga"
        ],
        "extension" : 87268,
        "habitantes" : 8384000
}
```

10.- Elimina las comunidades autónomas con menos de 50000 habitantes.

```
> db.ccaa.deleteMany({habitantes: {$lt:50000}})
{ "acknowledged" : true, "deletedCount" : 0 }
```

Ninguna tiene menos de 50000 habitantes.

11.- En la ccaa 02 añade el atributo extension con valor 47720 y modifica el nombre del atributo **capitas** por **capital**.

```
> db.ccaa.updateOne({codigo: "02"},{$rename:{'capitas': 'capital'}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.ccaa.updateOne({codigo: "02"},{$set:{extension: 47720}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

12.- En la ccaa 02 establece que los habitantes de la capital son 666000.

```
> db.ccaa.updateOne({codigo: "02"},{$set:{capital: {habitantes: 666000}}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.ccaa.find({codigo: "02"}).pretty()
{
        "_id" : ObjectId("61a63b4c184dabeabf67416f"),
        "codigo" : "02",
        "nombre" : "Aragón",
        "abreviatura" : "ARG",
        "provincias" : [
                "Huesca",
                "Teruel",
                "Zaragoza"
        ],
        "habitantes" : 1398000,
        "capital" : {
                "habitantes" : 666000
        },
        "extension" : 47720
}
```

13.- En la ccaa de nombre Islas Baleares, añade los datos de su capital Palma de Mallorca con 440000 habitantes.

```
> db.ccaa.updateOne({nombre: "Islas Baleares"},{$set:{capital: {nombre:"Palma de Mayorca", habitantes: 440000}}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.ccaa.find({nombre: "Islas Baleares"}).pretty()
{
        "_id" : ObjectId("61a63b4c184dabeabf674170"),
        "codigo" : "03",
        "nombre" : "Islas Baleares",
        "abreviatura" : "ISB",
        "provincias" : [
                "Baleares"
        ],
        "extension" : 4992,
        "habitantes" : 1130000,
        "capital" : {
                "nombre" : "Palma de Mayorca",
                "habitantes" : 440000
        }
}
```

6. Consultas en MongoDB

- Para consultar documentos de una colección se usa el método **find**.
  Fundamentalmente, este método admite tres sintaxis:
    - **find()**: devuelve el contenido de todos los documentos de la colección. Devuelve todos sus campos o atributos.
    - **find({filtro})**: devuelve el contenido de todos los atributos de los documentos de la colección que cumplen la condición del filtro.
    - **find({filtro},{proyección})**: devuelve el contenido de los atributos que se indiquen en la proyección para los documentos que cumplan la condición de filtro.

```
db.inventario.insertMany([
   { elemento: "journal", cantidad: 25, dim: { h: 14, w: 21, um: "cm" }, estado: "A" },
   { elemento: "notebook", cantidad: 50, dim: { h: 8.5, w: 11, um: "in" }, estado: "A" },
   { elemento: "paper", cantidad: 100, dim: { h: 8.5, w: 11, um: "in" }, estado: "D" },
   { elemento: "planner", cantidad: 75, dim: { h: 22.85, w: 30, um: "cm" }, estado: "C" },
   { elemento: "postcard", cantidad: 45, dim: { h: 10, w: 15.25, um: "cm" }, estado: "A" },
   { elemento: "postcard", cantidad: 45, dim: { h: 10, w: 15.25, um: "cm" }, estado: "A" },
   { elemento: "album", dim: { h: 12, w: 12, um: "in" }, estado: "B" },
   { elemento: "pencil", cantidad: 100, dim: null, estado: "D" }
]);
```

Ejemplo 1: obtener todos los datos de todos los documentos de inventario

```
> db.inventario.find()
```

Ejemplo 2: obtener todos los datos de los documentos de inventario cuyo elemento sea paper. Se usa un filtro de igualdad.

```
> db.inventario.find( { elemento: "paper"} )
```

6.1 Uso de proyecciones

- Siempre que no se pase un segundo parámetro al método **find**, se devuelven todos los atributos de una colección. Veamos dos ejemplos que no requieren indicar una proyección.

Ejemplo 3: obtener los valores de _id, elemento, cantidad de todos los documentos de inventario

```
> db.inventario.find( {}, { elemento: 1, cantidad: 1} )
```

Ejemplo 4: obtener los valores de elemento y cantidad de todos los documentos de inventario.

```
> db.inventario.find( {}, {_id: 0,  elemento: 1, cantidad: 1} )
```

Ejemplo 5: obtener los valores de los atributos de todos los datos de los elementos con estado A exceptuando _id y estado.

```
> db.inventario.find( {estado: "A"}, {_id: 0,  estado: 0} )
```

6.2 Uso de filtros

- Un filtro establece las condiciones que deben cumplir los documentos que se van a obtener como resultado.
- En un filtro se pueden tener varias condiciones de selección (selectores) que se podrán relacionar con otros selectores lógicos para realizar operaciones AND, OR y NOT.

Condiciones de igualdad:

- Se establecen con la sintaxis
  - **atributo:valor**
- Que equivale a poner en una instrucción SQL
  - **WHERE atributo=valor**

Ejemplo 6: obtener todos los elementos con estado A.

```
> db.inventario.find( {estado: "A"} )
```

Ejemplo 7: obtener todos los elementos cuya unidad de medida de dimensión sea cm

```
> db.inventario.find( { "dim.um": "cm"})
```

*Dado que um es un atributo se un subdocumento dim, su notación debe ser anidación de objetos **dim.um** y se debe escribir entre comillas.*

Ejemplo 8: obtener todos los elementos con estado A y cuya unidad de medida de dimensión sea cm.

```
> db.inventario.find( { estado: "A", "dim.um": "cm"})
```

*Las condiciones de un filtro separadas por coma implican **casi siempre** una operación AND.*

Selectores de comparación

Selector	Comparación realizada
$gt	Si valor de atributo es mayor que un valor
$gte	Si valor de atributo es mayor o igual que un valor
$lt	Si valor de atributo es menor que un valor
$lte	Si valor de atributo es menor o igual que un valor
$in	Si valor de atributo está incluido en lista de valores
$nin	Si valor de atributo no está incluido en lista de valores
$ne	Si valor de atributo distinto de valor

- Por ejemplo, para obtener los elementos con cantidad superior a 60:

```
db.inventario.find( { cantidad : {$gt : 60 } } );
```

- Y para obtener los elementos con estado A, C o D:

```
db.inventario.find( {estado : {$in :["A","C","D"] } } );
```

Ejemplo 9: obtener todos los elementos con estado A y cuya cantidad sea inferior a 50.

```
> db.inventario.find( { estado: "A", cantidad: {$lt: 50} })
```

Ejemplo 10: obtener todos los elementos con estado A, B o C y con ancho en cm superior o igual a 21.

```
> db.inventario.find( { estado: {$in: ["A","B","C"]}, "dim.um": "cm", "dim.w": {$gte: 21} })
```

Ejemplo 11: obtener todos los elementos cuyo nombre del elemento es mayor que "p".

```
> db.inventario.find( { elemento: {$gt: "p"} })
```

Ejemplo 12: obtener todos los elementos con cantidad comprendida entre 40 y 60 unidades.

```
> db.inventario.find(  { cantidad: {$gt: 40}, cantidad: {$lt: 60}} )
```

*Verás que esta solución **no es válida** (en casos en que se usa el mismo atributo en dos comparaciones, solo se hace la última)*

Veremos después la solución correcta usando $and.

Selectores lógicos

Selector	Operación realizada
$and	Realiza la operación AND entre una lista de expresiones
$or	Realiza la operación OR entre una lista de expresiones
$not	Realiza la opresión NOT sobre la expresión que se indique

Ejemplo 13: obtener todos los elementos con cantidad comprendida entre 40 y 60 unidades.

```
> db.inventario.find({ $and : [ {cantidad : {$gte: 40}}, {cantidad : {$lte: 60}} ]
```

Ejemplo 14: obtener todos los elementos con cantidad no comprendida entre 40 y 60 unidades.

```
> db.inventario.find({ $or : [ {cantidad : {$lt: 40}}, {cantidad : {$gt: 60}} ]  })
```

Ejemplo 15: obtener todos los elementos con estado A o D y cantidad superior a 70 unidades.

```
> db.inventario.find({ $and : [ { $or : [{ estado: 'A'}, {estado: 'D'}]  }, {cantidad : {$gt: 70}} ]  })
```

```
> db.inventario.find({ $and : [ {estado: {$in: ['A','D']}}, {cantidad : {$gt: 70}} ]  })
```

Ejemplo 16: obtener todos los elementos con cantidad no superior a 60 unidades.

```
> db.inventario.find({cantidad: {$not : {$gt: 60}}})
```

También se puede plantear, mucho más fácil y mejor, obteniendo los elementos con cantidad menor o igual a 60 unidades.

```
> db.inventario.find({cantidad: {$lte: 60}})
```

Pero hay una diferencia: En la primera solución se muestran elementos que tienen cantidad a valor null o que no tienen el atributo cantidad. En la segunda solución se muestran los que tienen atributo cantidad y, de ellos, los que en cantidad tienen un valor y es inferior o igual que 60.

Selectores de elemento

Selector	Operación realizada
$exists	Comprueba si cada documento contiene o no el atributo indicado
$type	Comprueba si cada documento contiene un atributo de un determinado tipo

Ejemplo 17: obtener todos los elementos que no tienen atributo cantidad.

```
> db.inventario.find( {cantidad: {$exists: false}}   )
```

Ejemplo 18: obtener todos los elementos que tienen atributo cantidad.

```
> db.inventario.find( {cantidad: {$exists: true}}   )
```

Ejemplo 19: obtener todos los elementos que tienen atributo dim con valores de tipo object (subdocumento).

```
> db.inventario.find( {dim: {$type : "object"}}   )
```

Selectores de evaluación

Selector	Operación realizada
$expr	Permite usar expresiones de agregación para seleccionar documentos
$regex	Obtiene documentos cuyos valores cumplen con una expression regular o patrón
$text	Obtiene documentos filtrados por una búsqueda de texto

Ejemplo 20: obtener todos los elementos cuyo nombre comienza por p.

```
> db.inventario.find( {elemento: {$regex : "^p.*"}}   )
```

Ejemplo 21: obtener todos los elementos cuyo nombre comienza por pa o por Pa

```
> db.inventario.find( {$or: [{elemento: {$regex : "^pa.*"}}, {elemento: {$regex: "^Pa*"}} ] } )
```

Ejemplo 22: obtener todos los elementos cuyo nombre termina en r.

```
> db.inventario.find( {elemento: {$regex : ".*r$"}}   )
```

Ejemplo 23: obtener todos los elementos cuyo nombre contiene el texto ANN sin diferenciar mayúsculas y minúsculas.

```
> db.inventario.find( {elemento: {$regex : "ANN", $options: "i"}}   )
```

6.3 Consultas sobre arrays

Selectores de arrays

Selector	Operación realizada
$all	Selecciona documentos que tienen un array que contiene todos los elementos que se indiquen en la consulta
$elemMatch	Selecciona documentos que tienen un array en el que alguno de sus elementos cumple con los criterios que se establezcan.
$size	Selecciona documentos que tienen un array con el tamaño indicado en la consulta

Para los ejemplos que vamos a usar, crea la siguiente colección **inventario2**.

```
db.inventario2.insertMany([
   { elemento: "journal", cantidad: 25, colores: ["black", "red"], dim_cm: [ 14, 21 ] },
   { elemento: "notebook", cantidad: 50, colores: ["red", "black"], dim_cm: [ 14, 21 ] },
   { elemento: "paper", cantidad: 100, colores: ["red", "black", "green"], dim_cm: [ 14, 21 ] },
   { elemento: "planner", cantidad: 75, colores: ["black", "red", "blue"], dim_cm: [ 22.85, 30 ] },
   { elemento: "postcard", cantidad: 45, colores: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

Ejemplo 24: obtener los elementos que incluyen el color red.

```
> db.inventario2.find( { colores: "red"} )
```

Ejemplo 25: obtener los elementos que tengan sólo los colores red y black y en este orden.

```
> db.inventario2.find( { colores: ["red", "black"]} )
```

Ejemplo 26: obtener los elementos que tengan los colores black y red.

```
> db.inventario2.find( { colores: {$all: ["red", "black"] } } )
```

Ejemplo 27: obtener los elementos que tengan 3 colores.

```
> db.inventario2.find( { colores: {$size: 3 } } )
```

Ejemplo 28: obtener los elementos que tienen alguna de sus dimensiones mayores que 22 y menores que 30.

```
> db.inventario2.find( { dim_cm : { $elemMatch: { $gt: 22, $lt: 30 }  } }  )
```

Ejemplo 29: obtener los elementos que tienen alguna de sus dimensiones mayores que 20 .

```
> db.inventario2.find( { dim_cm : { $elemMatch: { $gt: 20 }  } }  )
```

MongoDB Hoja 2

1.- Muestra los datos de todas las ccaa.

```
> db.ccaa.find().pretty()
{
        "_id" : ObjectId("61a63b3a184dabeabf67416e"),
        "codigo" : "01",
        "nombre" : "Andalucia",
        "abreviatura" : "AND",
        "capital" : {
                "nombre" : "Sevilla",
                "habitantes" : 690000
        },
        "provincias" : [
                "Almería",
                "Cádiz",
                "Córdoba",
                "Granada",
                "Huelva",
                "Jaén",
                "Sevilla",
                "Málaga"
        ],
        "extension" : 87268,
        "habitantes" : 8384000
}
{
        "_id" : ObjectId("61a63b4c184dabeabf67416f"),
        "codigo" : "02",
        "nombre" : "Aragón",
        "abreviatura" : "ARG",
        "provincias" : [
                "Huesca",
                "Teruel",
                "Zaragoza"
        ],
        "habitantes" : 1398000,
        "capital" : {
                "nombre" : "Zaragoza",
                "habitantes" : 666000
        },
        "extension" : 47720
}
{
        "_id" : ObjectId("61a63b4c184dabeabf674170"),
        "codigo" : "03",
        "nombre" : "Islas Baleares",
        "abreviatura" : "ISB",
        "provincias" : [
                "Baleares"
        ],
```

...

2.- Muestra sólo los nombres y habitantes de todas las ccaa

```
> db.ccaa.find({},{_id: 0, nombre: 1, habitantes: 1}).pretty()
{ "nombre" : "Andalucia", "habitantes" : 8384000 }
{ "nombre" : "Aragón", "habitantes" : 1398000 }
{ "nombre" : "Islas Baleares", "habitantes" : 1130000 }
{ "nombre" : "Canarias", "habitantes" : 2127000 }
```

3.- Muestra los nombres de las ccaa que tienen entre uno y dos millones de habitantes.

```
> db.ccaa.find({$and:[{habitantes: {$gte: 1000000}},
...     {habitantes: {$lte: 2000000}}]},
... {_id: 0, nombre: 1})
{ "nombre" : "Aragón" }
{ "nombre" : "Islas Baleares" }
```

4.- Muestra los nombres de las ccaa cuya capital tiene más de medio millón de habitantes.

```
> db.ccaa.find({"capital.habitantes": {$gte: 500000}}, {_id: 0, nombre: 1})
{ "nombre" : "Andalucia" }
{ "nombre" : "Aragón" }
```

5.- Muestra los nombres de las ccaa que tengan 3 o más provincias. Mostrar también el nombre de las provincias

```
> db.ccaa.find({provincias: {$exists:true}, $where:'this.provincias.length>=3'}, {_id: 0, nombre: 1})
{ "nombre" : "Andalucia" }
{ "nombre" : "Aragón" }
```

2. Solution

Actually, the `$where` operator is working fine, just not all documents contains the "tag" field, and caused the "no properties" error.

2.1 To fix it, try combine `$exists` and `$where` together :

```
db.domain.find( {tag : {$exists:true}, $where:'this.tag.length>3'} )

{
        "_id" : 1001,
        "domainName" : "google.com"
        "tag" : [
                "search engine",
                "search",
                "find anything",
                "giant"
                ]
}
```

7. Consultas avanzadas en MongoDB

7.1 Métodos de find

- A los resultados obtenidos con **find** se le pueden aplicar varios métodos modificadores. Puedes verlos ejecutando **db.inventario.find().help()**

Método	Funcionamiento
sort({campo: 1 | -1})	Ordenar los resultados por un campo
limit(n)	Mostrar sólo n documentos (los primeros)
skip(n)	Mostrar desde el documento n
count()	Total de documentos que devuelve la consulta
size()	Total de documentos que muestra la consulta

Ejemplo 1: obtener los nombres y cantidades de los elementos de inventario ordenando por cantidad de mayor a menor.

```
> db.inventario.find({}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad: -1})
```

Ejemplo 2: obtener el nombre del elemento que mayor cantidad tiene en inventario

```
> db.inventario.find({}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad: -1}).limit(1)
```

Ejemplo 3: obtener el nombre y cantidad disponible de los 3 elementos con menores cantidades en inventario.

```
> db.inventario.find({cantidad: {$exists: true}}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad:1}).limit(3)
```

Ejemplo 4: obtener el nombre y cantidad del elemento que ocupa la cuarta posición por cantidad disponible en inventario.

```
> db.inventario.find({cantidad: {$exists: true}}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad:-1}).skip(3).limit(1)
```

Ejemplo 5: obtener cuantos elementos con cantidad disponible superior a 40 hay cargados en inventario.

```
> db.inventario.find({cantidad: {$gt: 40}}).count()
```

Ejemplo 6: obtener los elementos y cantidades del inventario ordenados por cantidad ascendentemente y de forma que no salgan los dos primeros.

```
> db.inventario.find({}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad:1}).skip(2)
```

Ejemplo 7: obtener cuantos documentos genera la consulta anterior.

```
> db.inventario.find({}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad:1}).skip(2).count()
```

Ejemplo 8: obtener cuantos elementos se han mostrado en la consulta del ejemplo 6.

```
> db.inventario.find({}, {_id: 0, elemento: 1, cantidad: 1}).sort({cantidad:1}).skip(2).size()
```

MongoDB Hoja 3

Inicia una conexión cliente a la base de datos **geografía** de **MongoDB** y ejecuta las instrucciones para realizar las siguientes operaciones sobre la colección **ccaa.**

1.- Añade los siguientes documentos de ccaa:

```
{
        "codigo" : "05",
        "nombre" : "Cantabria",
        "abreviatura" : "CTB",
        "capital" : { "nombre" : "Santander", "habitantes" : NumberInt(180000) },
        "provincias" : [
                "Cantabria"
        ],
        "extension" : NumberInt(5321),
        "habitantes" : NumberInt(581000),
        "usohorario" : "UTC+01:00"
}

  {
        "codigo" : "06",
        "nombre" : "Asturias",
        "abreviatura" : "AST",
        "capital" : { "nombre" : "Oviedo", "habitantes" : NumberInt(220000) },
        "provincias" : [
                "Asturias"
        ],
        "extension" : NumberInt(10604),
        "habitantes" : NumberInt(1025000),
        "usohorario" : "UTC+01:00"
  }
```

...

```
> db.ccaa.insertMany([
... {
... codigo : "05",
... nombre : "Cantabria",
... abreviatura : "CTB",
... capital : { nombre : "Santander", habitantes : 180000 },
... provincias : ["Cantabria"],
... extension : 5321,
... habitantes : 581000,
... usohorario : "UTC+01:00"
... },
... {
... codigo : "06",
... nombre : "Asturias",
... abreviatura : "AST",
... capital : { nombre : "Oviedo", habitantes : 220000 },
... provincias : ["Asturias"],
... extension : 10604,
... habitantes : 1025000,
... usohorario : "UTC+01:00"
... },
... {
... codigo : "07",
... nombre : "Castilla-La Mancha",
... abreviatura : "CLM",
... capital : { nombre : "Toledo", habitantes : null },
... provincias : ["Albacete","Ciudad Real","Cuenca","Guadalajara","Toledo"],
... extension : 79461,
... habitantes : 2030000
... },
... {
... codigo : "08",
... nombre : "Castilla y León",
... abreviatura : "CyL",
... capital : null,
... provincias : [],
... extension : 96224,
... habitantes : 2410000
... }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61a66ac7a45bc65caa4829ad"),
                ObjectId("61a66ac7a45bc65caa4829ae"),
                ObjectId("61a66ac7a45bc65caa4829af"),
                ObjectId("61a66ac7a45bc65caa4829b0")
        ]
}
```

2.- Obtén los nombres y códigos de las comunidades autónomas con códigos 01,03,04,06 y 09.

```
> db.ccaa.find(
... {codigo: {$in: ["01","03","04","06","09"]}},
... {_id: 0, nombre: 1, codigo: 1}
... )
{ "codigo" : "03", "nombre" : "Islas Baleares" }
{ "codigo" : "04", "nombre" : "Canarias" }
{ "codigo" : "01", "nombre" : "Andalucia" }
{ "codigo" : "06", "nombre" : "Asturias" }
```

3.- Obtén los nombres y códigos de todas las comunidades autónomas con códigos comprendidos entre 03 y 06 ambos incluidos.

```
> db.ccaa.find(
... {codigo: {$in: ["03","04","05","06"]}},
... {_id: 0, nombre: 1, codigo: 1}
... )
{ "codigo" : "03", "nombre" : "Islas Baleares" }
{ "codigo" : "04", "nombre" : "Canarias" }
{ "codigo" : "05", "nombre" : "Cantabria" }
{ "codigo" : "06", "nombre" : "Asturias" }
```

4.- Obtén los nombres y códigos de todas las comunidades autónomas que tienen más de un millón de habitantes excepto las de códigos 03 y 05.

```
> db.ccaa.find(
... {$and:[
... {habitantes: {$gt: 1000000}},
... {codigo: {$nin: ["03","05"]}}
... ]},
... {_id: 0, nombre: 1, codigo: 1}
... )
{ "codigo" : "02", "nombre" : "Aragón" }
{ "codigo" : "04", "nombre" : "Canarias" }
{ "codigo" : "01", "nombre" : "Andalucia" }
{ "codigo" : "06", "nombre" : "Asturias" }
{ "codigo" : "07", "nombre" : "Castilla-La Mancha" }
{ "codigo" : "08", "nombre" : "Castilla y León" }
```

5.- Obtén los nombres y habitantes de las comunidades autónomas con capitales de más de 400000 habitantes y, de todas ellas, aquellas que tienen menos de un millón de habitantes en la CA o más de dos millones de habitantes.

```
> db.ccaa.find(
... {$and:[
... {"capital.habitantes": {$gt: 400000}},
... {$or: [
... {habitantes: {$lt: 1000000}},
... {habitantes: {$gt: 2000000}}
... ]}
... ]},
... {_id: 0, nombre: 1, habitantes: 1}
... )
{ "nombre" : "Andalucia", "habitantes" : 8384000 }
```

6.- Obtén los nombres y habitantes de las comunidades autónomas que tienen menos de un millón de habitantes y de las que tienen más de dos millones y menos de 400000 en su capital.

```
> db.ccaa.find(
... {$or:[
... {"capital.habitantes": {$gt: 400000}},
... {$or: [
... {habitantes: {$lt: 1000000}},
... {habitantes: {$gt: 2000000}}
... ]}
... ]},
... {_id: 0, nombre: 1, habitantes: 1}
... )
{ "nombre" : "Canarias", "habitantes" : 2127000 }
{ "nombre" : "Andalucia", "habitantes" : 8384000 }
{ "nombre" : "Cantabria", "habitantes" : 581000 }
{ "nombre" : "Castilla-La Mancha", "habitantes" : 2030000 }
{ "nombre" : "Castilla y León", "habitantes" : 2410000 }
```

7.- Obtén el nombre de cada comunidad autónoma y el de su capital.

```
> db.ccaa.find({},{_id:0, nombre:1, "capital.nombre":1})
{ "nombre" : "Aragón" }
{ "nombre" : "Islas Baleares" }
{ "nombre" : "Canarias", "capital" : { "nombre" : "Santa Cruz de Tenerife" } }
{ "nombre" : "Andalucia", "capital" : { "nombre" : "Sevilla" } }
{ "nombre" : "Cantabria", "capital" : { "nombre" : "Santander" } }
{ "nombre" : "Asturias", "capital" : { "nombre" : "Oviedo" } }
{ "nombre" : "Castilla-La Mancha", "capital" : { "nombre" : "Toledo" } }
{ "nombre" : "Castilla y León" }
```

8.- Obtén los nombres de comunidades autónomas que tienen el atributo capital a valor **null**.

```
> db.ccaa.find(
... {capital: null},
... {_id:0, nombre:1})
{ "nombre" : "Aragón" }
{ "nombre" : "Islas Baleares" }
{ "nombre" : "Castilla y León" }
```

9.- Obtén los nombres de comunidades autónomas que **no** tienen atributo usohorario.

```
> db.ccaa.find(
... {uso_horario: {$exists: false}},
... {_id:0, nombre:1})
{ "nombre" : "Aragón" }
{ "nombre" : "Islas Baleares" }
{ "nombre" : "Canarias" }
{ "nombre" : "Andalucia" }
{ "nombre" : "Cantabria" }
{ "nombre" : "Asturias" }
{ "nombre" : "Castilla-La Mancha" }
{ "nombre" : "Castilla y León" }
```

10.- Obtén los nombres de las comunidades autónomas que tienen atributo provincias y que no tienen cargadas provincias en ese atributo.

```
> db.ccaa.find(
... {$and:[
... {provincias: {$exists: true}},
... {provincias: {$size: 0}}
... ]},
... {_id:0, nombre:1})
{ "nombre" : "Castilla y León" }
```

11.- Añade las provincias de la CCAA de abreviatura CyL.

```
> db.ccaa.updateMany(
... {nombre: "Castilla y León"},
... {$push:{ provincias:{$each: ["Leon","Palencia","Burgos","Zamora","Valladolid","Soria","Salamanca","Avila","Segovia"]}}}
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

12.- Obtén los nombres de comunidades autónomas cuyo nombre comienza por C.

```
> db.ccaa.find(
... {nombre: {$regex : "^C.*"}},
... {_id:0, nombre:1}
... )
{ "nombre" : "Canarias" }
{ "nombre" : "Cantabria" }
{ "nombre" : "Castilla-La Mancha" }
{ "nombre" : "Castilla y León" }
```

13.- Obtén los datos de la comunidad autónoma cuya capital es Toledo.

```
> db.ccaa.find(
... {"capital.nombre": "Toledo"},
... {}
... ).pretty()
{
        "_id" : ObjectId("61ab2f3d8e540af9ecf97a44"),
        "codigo" : "07",
        "nombre" : "Castilla-La Mancha",
        "abreviatura" : "CLM",
        "capital" : {
                "nombre" : "Toledo",
                "habitantes" : null
        },
        "provincias" : [
                "Albacete",
                "Ciudad Real",
                "Cuenca",
                "Guadalajara",
                "Toledo"
        ],
        "extension" : 79461,
        "habitantes" : 2030000
}
```

14.- Obtén cuantas comunidades autónomas hay cargadas (cuántos documentos en ccaa).

```
> db.ccaa.find().count()
8
```

8. Importar y exportar

8.1 Exportar colecciones

mongoexport --autenticationDatabase admin -u root --db geografia --collection ccaa --out ccaa_completo.jason

mongoimport --autenticationDatabase admin -u root --db geografia --collection ccaa --file ccaa_completo.jason --jasonArray

Conexión Java a base de datos MongoDB

EJERCICIOS

1.- Inicia un proyecto Java para realizar una conexión básica con una base de datos de MongoDB. Añade las dependencias Maven necesarias para usar el driver Java para MongoDB.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>Hoja_4_Ejercicio_1</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mongodb</groupId>
            <artifactId>mongodb-driver-sync</artifactId>
            <version>4.3.1</version>
        </dependency>
    </dependencies>
</project>
```

==========

```
<dependencies>
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver-sync</artifactId>
      <version>4.3.1</version>
    </dependency>
  </dependencies>
```

==========

2.- En el método **main**:

Crea e instancia un objeto conector al servidor MongoDB que se está ejecutando en la misma máquina en la que se va a ejecutar este programa. La instanciación del objeto se realiza mediante una llamada al método *static* **create** de la clase *MongoClients***.** A este método se le pasa una URL con los datos de conexión. En este caso, vamos a pasar una URL especificando que el servidor está en **localhost** y escucha a los clientes en el puerto **27017** (son los valores por defecto si no se pasara esta URL). Además, vamos a añadirle el usuario y la contraseña (root, root en nuestro caso).

```
try (MongoClient cliente = MongoClients.create("mongodb://root:root@localhost:27017"))
{
}
```

Ojo! La interface MongoClient que se usa en la anterior instrucción es la que está incluida en el paquete ***com.mongodb.client****.*

3.- Programa que se responda a la instanciación del objeto **cliente** para indicar si se ha establecido o no la conexión (evaluando si **cliente** es o no **null).**

4.- Con el objeto **cliente,** crea e instancia un objeto para conectar con una de las bases de datos disponibles en el servidor, en nuestro caso **centroDB.** Este objeto de conexión se construye con una llamada al método **getDatabase** del objeto conector al servidor.

```
MongoDatabase db = cliente.getDatabase("centroDB");
```

5.- Mediante este último objeto crea un objeto para manejar los documentos de la colección alumnos.

```
MongoCollection<Document> alumnos = db.getCollection("alumnos");
```

6.- Una vez que tenemos el objeto **coleccion,** podemos realizar cualquier operación CRUD. En el siguiente código tienes una de las alternativas que hay para consultar todos los documentos de una colección.

```
MongoCursor<Document> cursor = alumnos.find().iterator();

while (cursor.hasNext())
{
Document doc = cursor.next();
System.out.println(doc.getString("nombre") + doc.getString("apellidos"));
}
```

```
public class H4E1_Main {
    public static void main(String[] args) {
        try(MongoClient cliente = MongoClients.create("mongodb://root:root@localhost:27017")){
            MongoDatabase db = cliente.getDatabase("geografia");
            MongoCollection<Document> ccaa = db.getCollection("ccaa");

            MongoCursor<Document> cursor = ccaa.find().iterator();
            while(cursor.hasNext()){
                Document doc = cursor.next();
                System.out.println(doc.getString("nombre") + " - " + doc.getString("abreviatura"));
            }
        }
    }
}
```

Programación Java con base de datos MongoDB

EJERCICIO

Realiza un programa Java que conecta con la base de datos **geografía de MongoDB** usada en la Hoja03_MongoDB_03 y que presenta el siguiente menú y, en función de la opción elegida, realiza sobre la colección **ccaa** la operación solicitada en las condiciones que se indican a continuación:

1.- Añadir Comunidad Autónoma sin provincias
2.- Consultar Comunidad Autónoma
3.- Asignar provincias a Comunidad Autónoma
4.- Añadir provincia a Comunidad Autónoma
5.- Modificar nombre de Comunidad Autónoma
6.- Eliminar provincia en Comunidad Autónoma
7.- Asignar capital a Comunidad Autónoma
8.- Asignar fecha Estatuto Autonomía
9.- Eliminar Comunidad Autónoma

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>Hoja_5_Ejercicio_1</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.mongodb</groupId>
            <artifactId>mongodb-driver-sync</artifactId>
            <version>4.3.1</version>
        </dependency>
    </dependencies>
</project>
```

Conexion.java

```java
public class Conexion {
    private static Conexion instance;
    private MongoDatabase db;
    private MongoCollection<Document> coleccion;

    private Conexion(){
        MongoClient cliente = MongoClients.create("mongodb://root:root@localhost:27017");
        if(cliente != null){
            System.out.println("Conexion OK");
            db = cliente.getDatabase("geografia");
            coleccion = db.getCollection("ccaa");
        }
    }

    public MongoDatabase getBaseDatos() {
        return db;
    }

    public MongoCollection<Document> getColeccion() {
        return coleccion;
    }

    public static Conexion getInstance() {
        if(instance == null){
            instance = new Conexion();
        }
        return instance;
    }
}
```

ComunidadAutonoma.java

```java
public class ComunidadAutonoma {
    private String codigo;
    private String nombre;
    private String abreviatura;
    private String nombreCapi;
    private double habitantes;
    private double habitantesCapi;
    private double extension;

    public ComunidadAutonoma() {...2 lines }
    public String getCodigo() {...3 lines }
    public void setCodigo(String codigo) {...3 lines }
    public String getNombre() {...3 lines }
    public void setNombre(String nombre) {...3 lines }
    public String getAbreviatura() {...3 lines }
    public void setAbreviatura(String abreviatura) {...3 lines }
    public String getNombreCapi() {...3 lines }
    public void setNombreCapi(String nombreCapi) {...3 lines }
    public double getHabitantes() {...3 lines }
    public void setHabitantes(double habitantes) {...3 lines }
    public double getHabitantesCapi() {...3 lines }
    public void setHabitantesCapi(double habitantesCapi) {...3 lines }
    public double getExtension() {...3 lines }
    public void setExtension(double extension) {...3 lines }
}
```

Main.java

```java
public class H5E1_Main {

    private static GestorBaseDatos gestor;

    public static void main(String[] args) {

        gestor = new GestorBaseDatos();

        int select;
        do {
            System.out.println("|--------------------MENU------------------|");
            System.out.println("| 1.Añadir Comunidad Autónoma sin provincias |");
            System.out.println("| 2.Consultar Comunidad Autónoma             |");
            System.out.println("| 3.Asignar provincias a Comunidad Autónoma  |");
            System.out.println("| 4.Añadir provincia a Comunidad Autónoma    |");
            System.out.println("| 5.Modificar nombre de Comunidad Autónoma   |");
            System.out.println("| 6.Eliminar provincia en Comunidad Autónoma |");
            System.out.println("| 7.Asignar capital a Comunidad Autónoma     |");
            System.out.println("| 8.Asignar fecha Estatuto Autonomía         |");
            System.out.println("| 9.Eliminar Comunidad Autónoma              |");
            System.out.println("|10.Muestra ccaa y capitales                 |");
            System.out.println("|11.Obtener ccaa con habitantes entre valores|");
            System.out.println("|12.Obtener ccaa uniprovinciales             |");
            System.out.println("|13.Obtener capitales con más habitantes que |");
            System.out.println("|14.Obtener ccaa sin fecha de estatuto       |");
            System.out.println("|15.Obtener provincias de ccaa               |");
            System.out.println("|16.Crear fichero JSON                       |");
            System.out.println("| 0.Salir                                    |");
            System.out.println("|--------------------------------------------|");
            System.out.println("| Selecciona opcion:                         |");
            select = Teclado.introInt("");
            System.out.println("|--------------------------------------------|");
            switch (select) {
```

GestorBaseDatos.java

```java
public class GestorBaseDatos {
    private final MongoCollection<Document> coleccion;

    public GestorBaseDatos() {
        this.coleccion = Conexion.getInstance().getColeccion();
    }
```

OPCIÓN 1:

Se piden por teclado los siguientes datos de una Comunidad Autónoma:

- Código
- Nombre
- Abreviatura
- Nombre y habitantes de la ciudad capital de la Comunidad Autónoma.
- Habitantes
- Extensión

Con los datos recogidos, se debe añadir un nuevo documento a la colección ccaa con los nombres de atributos usados en la hoja anterior. Los datos de la capital se guardan dentro de un subdocumento capital.

Se debe crear un objeto **Document** en el que se cargan todos los atributos y sus valores. Una vez que tenemos todos los atributos en el **Document,** lo insertamos mediante el método **insertOne.**

A continuación tienes un código de ejemplo que añade un documento con los datos de Extremadura (incluyendo las provincias).

```
Document doc1=new Document("codigo","09")
     .append("nombre","Extremadura")
     .append("abreviatura","EXT")
     .append("capital",
          new Document ("nombre","Mérida")
              .append("habitantes",59200))
     .append("provincias", Arrays.asList("Badajoz","Cáceres"))
      .append("habitantes",1067000)
     .append("extension", 41635);
coleccion.insertOne(doc1);
```

Main.java

```
case 1:
    ComunidadAutonoma ccaa = new ComunidadAutonoma();
    ccaa.setCodigo(Teclado.introString("Codigo de la ccaa:"));
    ccaa.setNombre(Teclado.introString("Nombre de la ccaa:"));
    ccaa.setAbreviatura(Teclado.introString("Abreviatura de la ccaa:"));
    ccaa.setNombreCapi(Teclado.introString("Nombre de la capital:"));
    ccaa.setHabitantesCapi(Teclado.introInt("Habitantes de la capital"));
    ccaa.setHabitantes(Teclado.introInt("Habitantes de la ccaa:"));
    ccaa.setExtension(Teclado.introInt("Extension de la ccaa:"));

    gestor.insertarCCAA(ccaa);
    break;
```

GestorBaseDatos.java

```
public void insertarCCAA(ComunidadAutonoma ccaa) {
    Document doc = new Document("codigo", ccaa.getCodigo())
            .append("nombre", ccaa.getNombre())
            .append("abreviatura", ccaa.getAbreviatura())
            .append("capital", new Document("nombre", ccaa.getNombreCapi())
                .append("habitantes", ccaa.getHabitantesCapi()))
            .append("habitantes", ccaa.getHabitantes())
            .append("extension", ccaa.getExtension());
    this.coleccion.insertOne(doc);
}
```

OPCIÓN 2:

Se pide por teclado el código de una Comunidad Autónoma y, si existe, se escribe su nombre, número de habitantes, extensión y el nombre de su capital. A continuación, se escriben los datos de la Comunidad Autónoma en formato JSON. Si no existe, se escribe un mensaje indicándolo.

Main.java

```
case 2:
    ccaa= gestor.consultaCCAA(Teclado.introString("Codigo de la ccaa que buscas:"));

    System.out.println("Comunidad Autonoma: " + ccaa.getNombre());
    System.out.println("Abreviatura: " + ccaa.getAbreviatura());
    System.out.println("Numero de Habitantes: " + ccaa.getHabitantes());
    System.out.println("Extension: " + ccaa.getExtension());
    System.out.println("Capital: " + ccaa.getNombreCapi());
    System.out.println("Habitantes de la Capital: " + ccaa.getHabitantesCapi());
    break;
```

GestorBaseDatos.java

```
public ComunidadAutonoma consultaCCAA(String codigo){
    ComunidadAutonoma ccaa = new ComunidadAutonoma();
    Document doc = this.coleccion.find(eq("codigo", codigo)).first();
    if(doc != null){
        ccaa.setCodigo(codigo);
        ccaa.setNombre(doc.getString("nombre"));
        ccaa.setAbreviatura(doc.getString("abreviatura"));
        ccaa.setHabitantes(doc.getDouble("habitantes"));
        ccaa.setExtension(doc.getDouble("extension"));
        ccaa.setNombreCapi(((Document) doc.get("capital")).getString("nombre"));
        ccaa.setHabitantesCapi(((Document) doc.get("capital")).getDouble("habitantes"));

          System.out.println("//TEXTO EJEMPLO ==========================================");
          System.out.println("//Comunidad Autonoma: " + doc.getString("nombre"));
          System.out.println("//Numero de Habitantes: " + doc.getDouble("habitantes"));
          System.out.println("//Extension: " + doc.getDouble("extension"));
          System.out.println("//Capital: "+((Document) doc.get("capital")).getString("nombre"));
          System.out.println("//");
          System.out.println("//Documento en modo JASON");
          System.out.println("//");
          System.out.println("//" + doc.toJson());
          System.out.println("//==========================================================");

    }
    return ccaa;
}
```

OPCIÓN 3:

Se pide por teclado el código de una comunidad autónoma y, si existe, se piden por teclado los nombres de las provincias de esa comunidad autónoma y se cargan esos nombres en un atributo **provincias** del documento correspondiente a la comunidad autónoma.

Tenemos que cargar un atributo que tiene valores de tipo array. En el valor asignado se debe pasar un ArrayList con los nombres de las provincias. Se cargarán todas las provincias (utilizar **set**)

Main.java

```
case 3:
    String codigo = Teclado.introString("Codigo de la ccaa: ");
    gestor.asignarProvincia(codigo);
    break;
```

GestorBaseDatos.java

```
public void asignarProvincia(String codigo) {
    Document doc = this.coleccion.find(eq("codigo", codigo)).first();
    if(doc != null){
        System.out.println("Insertando las provincias de " + doc.getString("nombre"));
        List<String> provincias = this.rellenarProvincias();
        this.coleccion.updateOne(eq("codigo", codigo), set("provincias", provincias));
    }else System.out.println("Error al abrir el documento");
}

private List<String> rellenarProvincias() {
    List<String> provincias = new ArrayList();
    boolean continuar = true;

    do{
        provincias.add(Teclado.introString("Provincia: "));
        continuar = Teclado.introBoolean("Insertar otra provincia? ");
    }while(continuar == true);
    return provincias;
}
```

OPCIÓN 4:

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado el nombre de una provincia de esa comunidad autónoma y se añade a las provincias ya existentes de esa comunidad autónoma.

Recuerda que el operador de modificación que usamos en las instrucciones mongodb para este tipo de modificación es **addToSet.**

Se debe comprobar si se ha realizado la modificación o no. Para ello, debes usar el objeto **UpdateResult** que pueden devolver los métodos **updeteOne y updateMany.**

Main.java

```
case 4:
    gestor.addProvincia(Teclado.introString("Codigo de la ccaa: "),
                        Teclado.introString("Provincia de la ccaa: "));
    break;
```

GestorBaseDatos.java

```
public void addProvincia(String codigo, String provincia) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null) {
        UpdateResult resul =
                coleccion.updateOne(eq("codigo", codigo), addToSet("provincias", provincia));

        System.out.println(resul.getMatchedCount() + " Documentos filtrados");
        System.out.println(resul.getModifiedCount() + " Documentos modificados");
    } else
        System.out.println("Error al abrir el documento");
}
```

OPCIÓN 5:

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado un nuevo nombre de esa comunidad autónoma y se **modifica** en el documento de la colección ese nombre.

Main.java

```
case 5:
    gestor.modificarNombre(Teclado.introString("Codigo de la ccaa: "),
                           Teclado.introString("Nuevo nombre: "));
    break;
```

GestorBaseDatos.java

```
public void modificarNombre(String codigo, String nombre) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null)
    {
        UpdateResult resul = coleccion.updateOne(eq("codigo", codigo), set("nombre", nombre));

        System.out.println(resul.getMatchedCount() + " Documentos filtrados");
        System.out.println(resul.getModifiedCount() + " Documentos modificados");
    } else
        System.out.println("Error! Documento no encontrado");
}
```

OPCIÓN 6:

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado el nombre de una provincia y se **elimina** ese nombre en el array de provincias.

Se debe escribir un mensaje que confirme si se ha realizado o no la eliminación de la provincia.

Main.java

```
case 6:
    gestor.eliminarProvincia(Teclado.introString("Codigo de la ccaa: "),
                             Teclado.introString("Provincia a eliminar: "));
    break;
```

GestorBaseDatos.java

```
public void eliminarProvincia(String codigo, String provincia) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null) {
        UpdateResult resul =
                coleccion.updateOne(eq("codigo", codigo), pull("provincias", provincia));

        System.out.println(resul.getMatchedCount() + " Documentos filtrados");
        System.out.println(resul.getModifiedCount() + " Documentos modificados");
    }else System.out.println("Error al abrir el documento");
}
```

OPCIÓN 7:

Se pide por teclado el código de una comunidad autónoma y, si existe y no tiene asignada capital, se piden los datos de su capital (nombre y habitantes) y se asignan al documento de la comunidad autónoma en el atributo capital.

Main.java

```
case 7:
    gestor.asignarCapital(Teclado.introString("Codigo de la ccaa: "),
            Teclado.introString("Capital de la ccaa: "),
            Teclado.introDouble("Habitantes de la capi: "));
    break;
```

GestorBaseDatos.java

```
public void asignarCapital(String codigo, String nombreCapi, double habitantesCapi) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null) {
        if ((Document) doc.get("capital") == null) {
            Document capital = new Document("nombre", nombreCapi)
                    .append("habitantes", habitantesCapi);
            UpdateResult resul
                    = coleccion.updateOne(eq("codigo", codigo), set("capital", capital));

            System.out.println(resul.getMatchedCount() + " Documentos filtrados");
            System.out.println(resul.getModifiedCount() + " Documentos modificados");
        } else
            System.out.println("Error! Ya existe capital");
    } else
        System.out.println("Error! Documento no encontrado");
}
```

OPCIÓN 8:

Se pide por teclado el código de una comunidad autónoma y, si existe, se pide por teclado la fecha de la entrada en vigor de su estatuto de autonomía en formato dd/MM/yyyy y se almacena en un atributo fecha_estatuto.

Main.java

```
case 8:
    String codigo = Teclado.introString("Codigo de la ccaa:");
    System.out.println("Fecha de entrada en vigor del estatuto autonomico:");
    int dia = Teclado.introInt("Dia: ");
    int mes = Teclado.introInt("Mes: ");
    int anio = Teclado.introInt("Anio: ");
    String fecha = dia + "/" + mes + "/" + anio;

    gestor.asignarFechaEstatuto(codigo, fecha);
    break;
```

GestorBaseDatos.java

```
public void asignarFechaEstatuto(String codigo, String fecha) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null) {
        UpdateResult resul
                = coleccion.updateOne(eq("codigo", codigo), set("fecha_estatuto", fecha));

        System.out.println(resul.getMatchedCount() + " Documentos filtrados");
        System.out.println(resul.getModifiedCount() + " Documentos modificados");
    } else
        System.out.println("Error! Documento no encontrado");
}
```

OPCIÓN 9:

Se pide por teclado el código de una comunidad autónoma y, si existe, se escribe su contenido en JSON y se pregunta si se quiere eliminar. Si la respuesta es S, se elimina el documento de la colección.

Main.java

```
case 9:
    gestor.eliminarComunidad(Teclado.introString("Codigo de la ccaa:"));
    break;
```

GestorBaseDatos.java

```
public void eliminarComunidad(String codigo) {
    Document doc = coleccion.find(eq("codigo", codigo)).first();
    if (doc != null) {
        System.out.println(doc.toJson());
        boolean eliminar = Teclado.introBoolean("Seguro que quieres eliminarlo?");
        if (eliminar == true) {
            DeleteResult resul = coleccion.deleteOne(eq("codigo", codigo));
            System.out.println(resul.getDeletedCount() + " Documentos eliminados");
        } else
            System.out.println("Bien pensado, no eliminamos");
    } else
        System.out.println("Error! Documento no encontrado");
}
```

Consultas avanzadas en Java con base de datos MongoDB

EJERCICIO

Realiza un programa Java que conecta con la base de datos **geografia de MongoDB,** presenta el siguiente menú y, en función de la opción elegida, realiza sobre la colección **ccaa** la operación solicitada en las condiciones que se indican a continuación:

1.- Obtener comunidades autónomas y capitales
2.- Obtener comunidades autónomas con habitantes comprendidos entre valores
3.- Obtener comunidades autónomas uniprovinciales.
4.- Obtener capitales de comunidad autónoma con más habitantes que
5.- Obtener comunidades autónomas sin fecha de estatuto
6.- Obtener provincias de comunidad autónoma
7.- Crear fichero JSON

OPCIÓN 1:

Se muestran los nombres de todas las comunidades autónomas y los de sus capitales con formato JSON.

Debes realizarlo con **projection.**

Main.java

```
case 10:
    gestor.comuMasCapi();
    break;
```

GestorBaseDatos.java

```
public void comuMasCapi(){
    MongoCursor<Document> cursor =
            this.coleccion.find().projection(fields(include("nombre", "capital.nombre"),
                    exclude("_id"))).iterator();
    while(cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println(doc.toJson());
    }
}
```