Comandos Básicos:

- use nombreBD -> Cambia de base de datos (también sirve para crear una BD)

- show dbs. Muestra las bases de datos que hay en la instancia

- show collections. Muestra las colecciones de la BD activa

- show users. Muestra los usuarios de la BD activa

- db.help(). Muestra ayuda de los métodos globales de las bases de datos

- db.collection.help(). Muestra ayuda de los métodos de las colecciones

- db.shutdownServer(). Detiene el servidor de MongoDB al que está conectado el shell. Es necesario estar situado en la

base de datos admin (no lo usaremos en Atlas)

- Los métodos db.collection.insertOne() y db.collection.insertMany()

- db.createCollection()

Consultas MongoDB:

1) Estudiantes del Grado en "Ingeniería Informática"

```
db.estudiantes.find({"titulo": "Ingeniería Informática"})
```

2) Estudiantes que hayan comenzado sus estudios en 2021

```
db.estudiantes.find({"comienzo": 2021})
```

3) Estudiantes que hayan finalizado sus estudios en 2021

```
db.estudiantes.find({"fin": 2021})
```

4) Estudiantes que hayan comenzado sus estudios antes de 2016

```
db.estudiantes.find({"comienzo": {"$lt": 2016}})
```

5) Estudiantes que hayan comenzado sus estudios en 2016 o en años anteriores

```
db.estudiantes.find({"comienzo": {"$lte": 2016}})
```

6) Estudiantes del Grado en "Ingeniería Informática" que hayan comenzado sus estudios después de 2018

db.estudiantes.find({"titulo": "Ingeniería Informática", "comienzo": {"$gt": 2018}})

7) Estudiantes del Grado en "Ingeniería Mecánica" que hayan comenzado sus estudios entre 2014 y 2020

db.estudiantes.find({"titulo": "Ingeniería Mecánica", $and: [{"comienzo": {"$gt": 2014}}, {"comienzo": {"$lt": 2020}}]})

8) Estudiantes que han concluido sus estudios (parámetro $exists)

db.estudiantes.find({"fin": {"$exists": true}})

9) Todos los estudiantes mostrando, únicamente, su nombre

db.estudiantes.find({}, {"nombre":1, "_id":0})

10) Todos los estudiantes mostrando el nombre y el título

db.estudiantes.find({}, {"nombre":1, "titulo":1, "_id":0})

11) Estudiantes que hayan empezado sus estudios en 2016 y los hayan terminado en 2021

db.estudiantes.find({$and: [{"comienzo": 2016}, {"fin": 2021}]})

12) Estudiantes del Grado en "Ingeniería Informática" o del Grado en "Ingeniería Forestal"

db.estudiantes.find({$or: [{"titulo": "Ingeniería Informática"}, {"titulo": "Ingeniería Forestal"}]})

13) Estudiantes del Grado en "Ingeniería Mecánica" que no hayan finalizado sus estudios o que los hayan empezado en el año 2015

db.estudiantes.find({$and: [{"titulo": "Ingeniería Mecánica"}, {$or: [{"comienzo": 2015}, {"fin": {"$exists": false}}]}]})

Productos:

```
[{"_id": 1, "item": "cuaderno", "cantidad": 25,
  "colores": [ "negro", "rojo" ], "dim_cm": [ 14, 21 ]
```

```
,{ "_id": 2, "item": "tarjeta", "cantidad": 45,"colores": ["azul"
  ],"dim_cm": [ 10, 15.25]},
{ "_id": 3, "item": "bloc","cantidad": 50, "colores": [ "rojo", "blanco"],
"dim_cm": [14, 21]},{
  "_id": 4, "item": "cartulina","cantidad": 100, "colores": [
    "rojo", "blanco", "negro"], "dim_cm":[14, 21]
}, {"_id": 5, "item": "agenda", "cantidad":75, "colores":
["blanco","rojo"], "dim_cm": [22.85, 30]}]
```

1) Productos que sean de color rojo

db.productos.find({colores:"rojo"})

2) La consulta anterior devuelve todos los productos que son de color rojo pero que también pueden ser de otros colores. Si queremos encontrar los productos que solo son de color rojo

db.productos.find({colores:["rojo"]})

3) Productos que son de color rojo y blanco

db.productos.find({colores:["rojo", "blanco"]})

4) La consulta anterior solo devuelve los productos que son de color rojo y blanco pero cuyo array de colores esté escrito en ese orden. Para obtener los productos de

color rojo y blanco independientemente del orden, hay que usar el operador $all

db.productos.find({colores:{$all:["rojo", "blanco"]}})

5) Productos que tengan, al menos, una dimensión mayor que 26

db.productos.find({dim_cm:{$gt:26}})

6) Productos cuyas dimensiones sean o menor que 12 o mayor que 14

db.productos.find({dim_cm:{$lt:12, $gt:14}})

Devuelve el documento "_id: 2" ya que es el único donde todos los elementos del array "dim_cm": [10, 15.25]

cumplen algunas de las condiciones. En este caso, la dimensión "10" cumple una de las condiciones (10 < 12), y la

dimensión "15.25" cumple otra de las condiciones (15.25 > 14)

Los documentos con "dim_cm": [14, 21] no se devuelven porque la dimensión "14" no cumple ninguna de las

condiciones

7) Productos que tengan, al menos, una de sus dimensiones mayor que 20 y menor que 22

db.productos.find({dim_cm:{$elemMatch:{$gt:20, $lt:22}}})

db.productos.find({dim_cm:{$gt:20, $lt:22}})

El propósito del operador $elemMatch es buscar los documentos en los que, al menos, uno de los valores del array cumpla la condición

8. Productos cuya primera dimensión sea mayor que 13

db.productos.find({"dim_cm.0":{$gt:13}})

9. Productos que tengan 3 colores

db.productos.find({"colores":{$size:3}})

10. Productos que solo tengan 1 color

db.productos.find({colores:{$size:1}})

11. Productos cuya segunda dimensión sea 21

db.productos.find({"dim_cm.1":21})

12. Productos que tengan, al menos, colores rojo y negro

db.productos.find({colores:{$all:["rojo", "negro"]}})

13. Productos cuyas dimensiones sean exactamente 10 x 15.25

db.productos.find({dim_cm:[10, 15.25]})

14. Productos de color rojo de los que haya más de 60 unidades

db.productos.find({cantidad:{$gt:60}, colores:"rojo"})

15. Productos de color blanco o de los que haya más de 60 unidades

db.productos.find({$or:[{cantidad:{$gt:60}}, {colores:"blanco"}]})

16. Productos cuya segunda dimensión esté entre 10 y 18

db.productos.find({"dim_cm.1":{$gt:10, $lt:18}})

Productos2:

```
[{ "_id": 1, "item": "cuaderno", "cantidad": 25, "tam": { "h": 14, "w": 21, "unidad": "cm" }, "estado": "A" },
  { "_id": 2, "item": "tarjeta", "cantidad": 45, "tam": { "h": 10, "w": 15.25, "unidad": "cm" }, "estado": "A" },
  { "_id": 3, "item": "agenda", "cantidad": 75, "tam": { "h": 22.85, "w": 30, "unidad": "cm" }, "estado": "D" },
  { "_id": 4, "item": "bloc", "cantidad": 50, "tam": { "h": 8.5, "w": 11, "unidad": "in" }, "estado": "A" },
  { "_id": 5, "item": "cartulina", "cantidad": 100, "tam": { "h": 8.5, "w": 11, "unidad": "in" }, "estado": "D" }]
```

1) Productos con tamaño en pulgadas, altura menor a 15 y estado "A" o "D":

db.productos2.find({$and: [{"tam.unidad": "in", "tam.h": {$lt: 15},}, {$or: [{estado: "A"},{estado: "D"},],},]})

2) Productos con tamaño en centímetros y cantidad mayor a 20 o tamaño en pulgadas y cantidad menor o igual a 50:

db.productos2.find({$or: [{"tam.unidad": "cm", "cantidad": {$gt: 20 }}, {"tam.unidad": "in", "cantidad": {$lte: 50}}]})

3) Productos con cantidad mayor que 30 y tamaño en centímetros, proyectando sólo los campos "item" y "cantidad"

db.productos2.find({$and:[{"tam.unidad":"cm"},{"cantidad":{$gt:30}}]},{"item":1, "cantidad":1, "_id":0})

4) Productos con tamaño en centímetros y estado "A", ordenados por cantidad descendentemente

db.productos2.find({$and:[{"tam.unidad": "cm"},{"estado":"A"}]}).sort({"cantidad":-1})

5) Productos con estado "A" y cantidad mayor a 30, proyectando sólo los campos "item" y "tam", y ordenados por tamaño de ancho descendentemente

db.productos2.find({$and:[{"estado":"A"}, {"cantidad":{$gt:30}}]},{"item": 1,"tam": 1,"_id":0}).sort({"tam.w":-1})

Productos3:

```
[{ "_id": 1, "item": "cuaderno", "stock": [ { "almacen": "A", "cantidad": 5 }, { "almacen": "C", "cantidad": 15 } ] },
  { "_id": 2, "item": "tarjeta", "stock": [ { "almacen": "C", "cantidad": 5 } ] },
  { "_id": 3, "item": "agenda", "stock": [ { "almacen": "A", "cantidad": 60 }, { "almacen": "B", "cantidad": 15 } ] },
  { "_id": 4, "item": "bloc", "stock": [ { "almacen": "A", "cantidad": 40 }, { "almacen": "B", "cantidad": 5 } ] },
  { "_id": 5, "item": "cartulina", "stock": [ { "almacen": "B", "cantidad": 15 }, { "almacen": "C", "cantidad": 35 } ] } ]
```

1) Productos con stock en el almacén "A" y cantidad igual a 40 en el almacen

db.productos3.find({ "stock": { $elemMatch: { "almacen": "A", "cantidad": 40 }}})

2) Documentos que tienen al menos un almacén con una cantidad inferiror a 10 para cualquier producto

db.productos3.find({ "stock": { $elemMatch: { "cantidad": { $lt: 10 }}}})

3) Documentos que tienen stock en el almacén "C" con una cantidad mayor a 10 para cualquier producto

db.productos3.find({ "stock": { $elemMatch: { "almacen": "C", "cantidad": { $gt: 10 }}}})

4) Documentos que tienen stock en los almacenes "A" y "B" para cualquier producto

db.productos3.find({$and:[{"stock":{$elemMatch:{"almacen":"A"}}}, {"stock":{$elemMatch:{"almacen":"B"}}}]})

Ej1) Elimina todos los documentos de la colección "productos"

db.productos.deleteMany({})

Ej2) Elimina todos los documentos de la colección "productos" cuyo campo "cantidad" es mayor que 20

db.productos.deleteMany({cantidad:{$gt:20}})

1) Añadir el color "naranja" al producto "cuaderno"

db.productos.updateOne({item:"cuaderno"}, {$push:{colores:'naranja'}})

2) Añadir los colores "magenta" y "verde" al producto "bloc"

db.productos.updateOne({item:"bloc"}, {$push:{colores:{$each:['magenta','verde']}}})

3) Eliminar el color "blanco" al producto "cartulina"

db.productos.updateOne({item:"cartulina"}, {$pull:{colores:"blanco"}})

4) Aumentar en 10 las cantidades de los productos que tengan más de 50 unidades

db.productos.updateMany({cantidad:{$gt:50}}, {$inc:{cantidad:10}})

5) Añadir al producto "cuaderno" un nuevo campo de tipo array que se llame "precios" y asignarle los valores 30 y 60

db.productos.updateOne({item:"cuaderno"}, {$set:{precios:[30, 60]}})

6) Aumentar en 3 unidades la primera dimensión del producto "tarjeta"

db.productos.updateOne({item:"tarjeta"}, {$inc:{"dim_cm.0":3}})

1) Cambiar el estado del producto "tarjeta" al valor "C"

db.productos2.updateOne({item:'tarjeta'}, {$set:{estado:'C'}})

2) Cambiar las dimensiones de producto "bloc" a (10, 14)

db.productos2.updateOne({item:'bloc'}, {$set:{"tam.h":10, "tam.w":14}})

3) Pasar a "cm" todos los productos "in"

db.productos2.updateMany({"tam.unidad":'in'}, {$set:{"tam.unidad":'cm'}})

4) Añadir a todos los productos un nuevo campo "forma" dentro del campo "tam" con el valor "rectángulo"

db.productos2.updateMany({}, {$set:{"tam.forma":"rectángulo"}})

5) Eliminar los productos con estado "A"

db.productos2.deleteMany({estado:'A'})

1) Añadir un nuevo almacén "A" al stock del producto "tarjeta" con una cantidad de 40

db.productos3.updateOne({ item : 'tarjeta' }, { $push : { stock : { almacen : 'A', cantidad : 40 } } })

2) Aumentar en 100 unidades la cantidad del stock de todos los almacenes de tipo "C".

NOTA: para resolver esta consulta hay que usar el operador arrayFilters

db.productos3.updateMany({}, {$inc:{"stock.$[elemento].cantidad":100}}, {arrayFilters:[{"elemento.almacen":"C"}]})

Crear una base de datos "Agencia" y una colección "viajeros" con estos datos

db.viajeros.insertMany ([{ _id: 1, first : "Maria",

travel : [{ country: "Canada", visits: 3, rating: 7 },{ country: "Poland", visits: 1, rating: 8 },

{ country: "Thailand", visits: 2, rating: 9 }] },

{ _id: 2, first : "Chen",

travel : [{ country: "Thailand", visits: 3, rating: 7 },{ country: "Canada", visits: 2, rating: 9 },

{ country: "Costa Rica", visits: 4, rating: 8 }] },

{ _id: 3, first : "Gladys",

travel : [{ country: "Canada", visits: 1, rating: 8 }, { country: "Thailand", visits: 2, rating: 9 },{ country: "Australia", visits: 3, rating: 10 }] }]);

Realizar las siguientes operaciones CRUD:

1) Obtener el nombre de los viajeros que han visitado 2 veces Canadá y lo han valorado con un 9

```
db.viajeros.find({ "travel.country": "Canada", "travel.visits": 2, "travel.rating": 9 }, { "first": 1, "_id": 0 })
```

2) Obtener los viajeros que han realizado 4 visitas a un país

```
db.viajeros.find({ "travel.visits": 4 }, { "first": 1, "_id": 0 })
```

3) Obtener los viajeros que han realizado más de 3 visitas a un país

```
db.viajeros.find({ "travel.visits": { $gt: 3 } }, { "first": 1, "_id": 0 })
```

4) Obtener los viajeros que han realizado más de 2 visitas a un país y han puntuado con menos de un 8 (pueden ser distintos países)

```
db.viajeros.find({ "travel.visits": { $gt: 2 }, "travel.rating": { $lt: 8 } }, { "first": 1, "_id": 0 })
```

5) Obtener los viajeros que han realizado más de 2 visitas a un país y han puntuado con menos de un 8 pero del mismo país ($elemMatch)

```
db.viajeros.find({ "travel": { $elemMatch: { "visits": { $gt: 2 }, "rating": { $lt: 8 } } } }, { "first": 1, "_id": 0 })
```

6) Igual que la anterior pero ahora la puntuación debe estar entre 6 y 9

```
db.viajeros.find({ "travel": { $elemMatch: { "visits": { $gt: 2 }, "rating": { $gte: 6, $lte: 9 } } } }, { "first": 1, "_id": 0 })
```

7) Actualizar el número de visitas de María a Canadá con 5 visitas

```
db.viajeros.updateOne({ "_id": 1, "travel.country": "Canada" }, { $set: { "travel.$.visits": 5 } })
```

8) Actualizar los viajes de Chen añadiendo una visita a España y con una puntuación de 10

```
db.viajeros.updateOne({ "_id": 2 }, { $push: { "travel": { "country": "Spain", "visits": 1, "rating": 10 } } })
```

```
db.inventario.insertMany ( [
{ _id: 1, item: "cuaderno", tam: { h: 14, w: 21, unidad: "cm" }, estado: "A",
stock: [ { almacen: "A", cantidad: 5 }, { almacen: "C", cantidad: 15 } ] },
{ _id: 2, item: "tarjeta", tam: { h: 10, w: 15.25, unidad: "cm" }, estado: "A",
stock: [ { almacen: "C", cantidad: 5 } ] },
{ _id: 3, item: "agenda", tam: { h: 22.85, w: 30, unidad: "cm" }, estado: "D",
stock: [ { almacen: "A", cantidad: 60 }, { almacen: "B", cantidad: 15 } ] },
{ _id: 4, item: "bloc", tam: { h: 8.5, w: 11, unidad: "in" }, estado: "A",
stock: [ { almacen: "A", cantidad: 40 }, { almacen: "B", cantidad: 5 } ] },
{ _id: 5, item: "cartulina", tam: { h: 8.5, w: 11, unidad: "in" }, estado: "D",
stock: [ { almacen: "B", cantidad: 15 }, { almacen: "C", cantidad: 35 } ] } ] )
```

Resolver las siguientes operaciones CRUD:

1) Obtener el nombre de los productos que tienen estado "A"

```
db.inventario.find({ estado: "A" }, { item: 1, _id: 0 })
```

2) Obtener el nombre de los productos que tienen estado "D" o cuya primera dimensión ("h") es 14

```
db.inventario.find({ $or: [{ estado: "D" }, { "tam.h": 14 }] }, { item: 1, _id: 0 })
```

3) Obtener el nombre de los productos que solo están en un almacén

```
db.inventario.find({ "stock.1": { $exists: false } }, { item: 1, _id: 0 })
```

4) Mostrar, de todos los documentos, el nombre del producto y los almacenes en los que se encuentran (no interesa la cantidad)

```
db.inventario.find({}, { item: 1, "stock.almacen": 1, _id: 0 })
```

5) Obtener el nombre de los productos de los que queden menos de 8 unidades en cualquiera de sus almacenes

```
db.inventario.find({ "stock.cantidad": { $lt: 8 } }, { item: 1, _id: 0 })
```

6) Obtener el nombre de los productos de los que haya más de 10 unidades en el almacén "B"

db.inventario.find({ "stock.almacen": "B", "stock.cantidad": { $gt: 10 } }, { item: 1, _id: 0 })

7) El producto "tarjeta" se va a almacenar también en el almacén "B". Añadir ese almacén al stock de "tarjeta" (de momento con ninguna unidad)

db.inventario.updateOne({ item: "tarjeta" }, { $addToSet: { "stock": { almacen: "B", cantidad: 0 } } })

8) El producto "cartulina" ha dejado de estar en el almacén "B". Eliminar ese almacén de su stock

db.inventario.updateOne({ item: "cartulina" }, { $pull: { "stock": { almacen: "B" } } })

db.inventario.deleteOne({ item: "cartulina", "stock.almacen": "B" })