
ACCESO A DATOS MongoDB



mongoDB®

2º DAM

Curso 2021/2022 IES Teis (Vigo)



Índice

1. Qué es
2. Relacional vs MongoDB
3. Modelo de datos JSON
 1. Objetos
 2. Array
 3. Tipo de datos



Qué es

- ☐ Base de datos NoSQL - orientada a DOCUMENTOS
- ☐ Esquema libre → Cada entrada/registro puede tener un esquema de datos distinto
- ☐ Adecuado para implementar microservices
- ☐ Características:
 - ☐ Gran velocidad (C++)
 - ☐ Sencillo sistema de consultas



Relacional vs MongoDB

Modelo relacional	MongoDB
Base de datos	Base de datos
Tabla	Colección
Fila	Documento
Columna	Campo
Índice	Índice
Join	Documento embebido o referencia

Modelo de datos. JSON

Comparativa XML y JSON

```
<listaHoteles>
  <hotel>
    <nombre>Iberostar Parque Central</nombre>
    <categoria>5 estrellas</categoria>
    <pais>Cuba</pais>
    <poblacion>La Habana</poblacion>
    <precio>450</precio>
  </hotel>
  <hotel>
    <nombre>Pelicano</nombre>
    <categoria>4 estrellas</categoria>
    <pais>Cuba</pais>
    <poblacion>Cayo Largo del Sur</poblacion>
    <precio>220</precio>
  </hotel>
</listaHoteles>
```

```
{
  "listaHoteles": {
    "hotel": [
      {
        "nombre": "Iberostar Parque Central",
        "categoria": "5 estrellas",
        "pais": "Cuba",
        "poblacion": "La Habana",
        "precio": 450
      },
      {
        "nombre": "Pelicano",
        "categoria": "4 estrellas",
        "pais": "Cuba",
        "poblacion": "Cayo Largo del Sur",
        "precio": 220
      }
    ]
  }
}
```

JSON. Objeto

□ Ejemplo de objeto:

```
{
  "persona": {
    "nombre": "Sabela",
    "oficio": "analista",
    "ciudad": "Vigo"
  },
  "hotel": {
    "nombre": "Iberostar Parque Central",
    "categoria": "5 estrellas",
    "pais": "Cuba",
    "poblacion": "La Habana",
    "precio": 450
  }
}
```




JSON. Array

```
{
  "hotel": [
    {
      "nombre": "Iberostar Parque Central",
      "categoria": "5 estrellas",
      "pais": "Cuba",
      "poblacion": "La Habana",
      "precio": 450
    },
    {
      "nombre": "Pelícano",
      "categoria": "4 estrellas",
      "pais": "Cuba",
      "poblacion": "Cayo Largo del Sur",
      "precio": 220
    }
  ]
}
```



JSON. Tipos de datos

- Los **tipos de datos** válidos en JSON:
 - ☐ **String:** { "nombre":"Roger" }
 - ☐ **Number:** { "edad":19 }
 - ☐ **Objeto JSON:** { "empleado": { "nombre":"Roger","edad":19 } }
 - ☐ **Array:** { "empleados":["Roger","Nil","Biel"] }
 - ☐ **Boolean:** { "rebajado":true }
 - ☐ **Null:** { "sexo":null }

Práctica Inicial - MongoDB

- Convierte el fichero 'instituto.xml' a un objeto JSON
- Opciones:
 - Manual
 - Automático
 - Leer .xml (xStream) y paso Objeto Java
 - Escritura .json (GSON)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<instituto>
  <departamento telefono="112233" tipo="A">
    <codigo>IFC1</codigo>
    <nombre>Informática</nombre>
    <empleado salario="2000">
      <puesto>Asociado</puesto>
      <nombre>Juan Parra</nombre>
    </empleado>
    <empleado salario="2300">
      <puesto>Profesor</puesto>
      <nombre>Alicia Martín</nombre>
    </empleado>
  </departamento>
```



Instalación/Uso MongoDB





Índice

1. Descarga/Uso de Atlas/Docker
2. Iniciar instalación
3. Como un servicio
4. **Compass/Mongo** - Robo 3T <https://robomongo.org> /**Mongo** - Studio 3T
5. Conexión contra el servidor (Local/Atlas)
6. Cliente
7. CRUD



Atlas

- Cuenta en Atlas
- Conexión al servidor generado (ejemplo):
 - `mongodb+srv://root:root@freecluster.bgowr.mongodb.net/test`

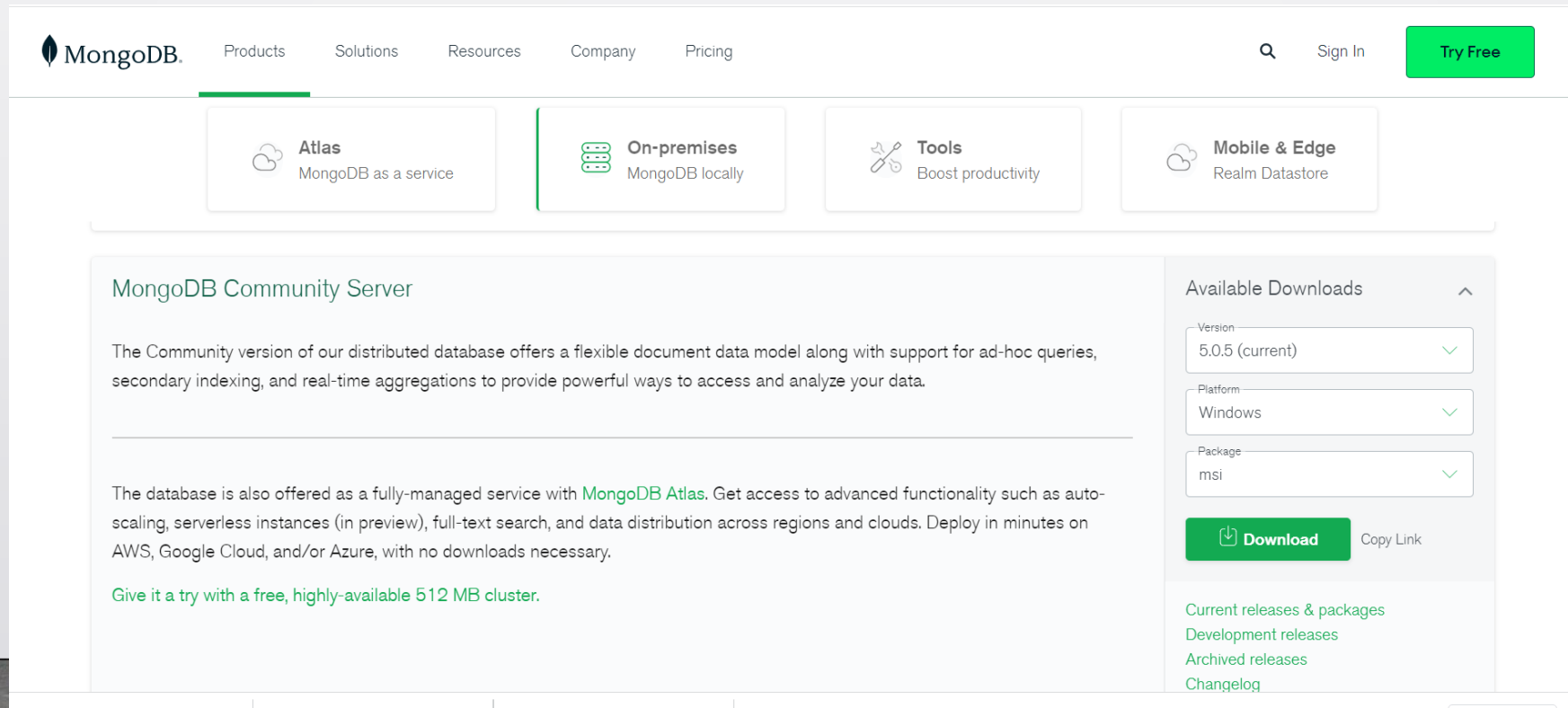


Docker

- https://hub.docker.com/_/mongo
- Instalación y servicio activo para conexión
 - (ej. En puerto **61076**)

Descarga

- Ir a la web oficial y descargar la última versión estable:
<https://www.mongodb.com/try/download/community>
- https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-5.0.5.zip



The screenshot shows the MongoDB website's download section. At the top, there's a navigation bar with links for Products, Solutions, Resources, Company, and Pricing, along with a search icon, a 'Sign In' link, and a 'Try Free' button. Below the navigation bar, there are four main product categories: Atlas (MongoDB as a service), On-premises (MongoDB locally), Tools (Boost productivity), and Mobile & Edge (Realm Datastore). The 'On-premises' category is highlighted with a green border. Below these categories, the 'MongoDB Community Server' section is visible. It describes the Community version as a flexible document data model with support for ad-hoc queries, secondary indexing, and real-time aggregations. It also mentions that the database is offered as a fully-managed service with MongoDB Atlas. To the right of the text, there's a 'Available Downloads' section with dropdown menus for Version (5.0.5 (current)), Platform (Windows), and Package (msi). Below these dropdowns are a 'Download' button and a 'Copy Link' button. At the bottom of the 'Available Downloads' section, there are links for 'Current releases & packages', 'Development releases', 'Archived releases', and 'Changelog'.

MongoDB

Products Solutions Resources Company Pricing

Atlas
MongoDB as a service

On-premises
MongoDB locally

Tools
Boost productivity

Mobile & Edge
Realm Datastore

MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with [MongoDB Atlas](#). Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

Give it a try with a free, highly-available 512 MB cluster.

Available Downloads

Version
5.0.5 (current) ✓

Platform
Windows ✓

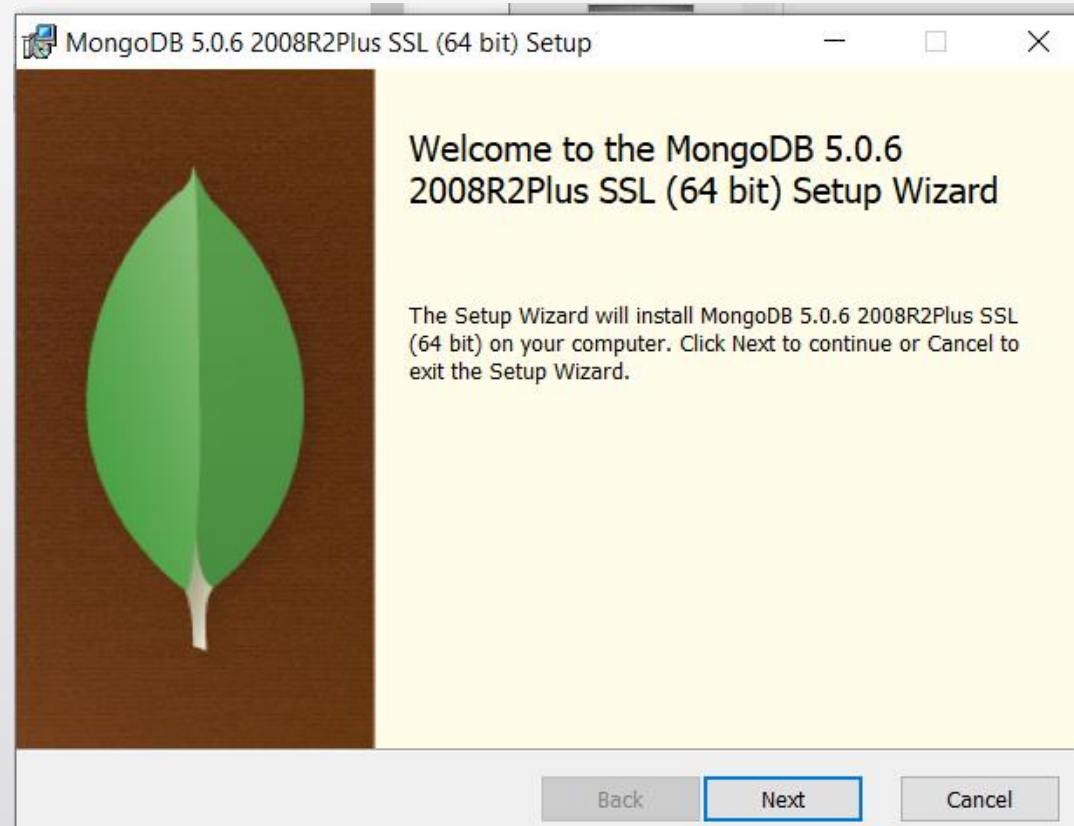
Package
msi ✓

[Download](#) [Copy Link](#)

[Current releases & packages](#)
[Development releases](#)
[Archived releases](#)
[Changelog](#)

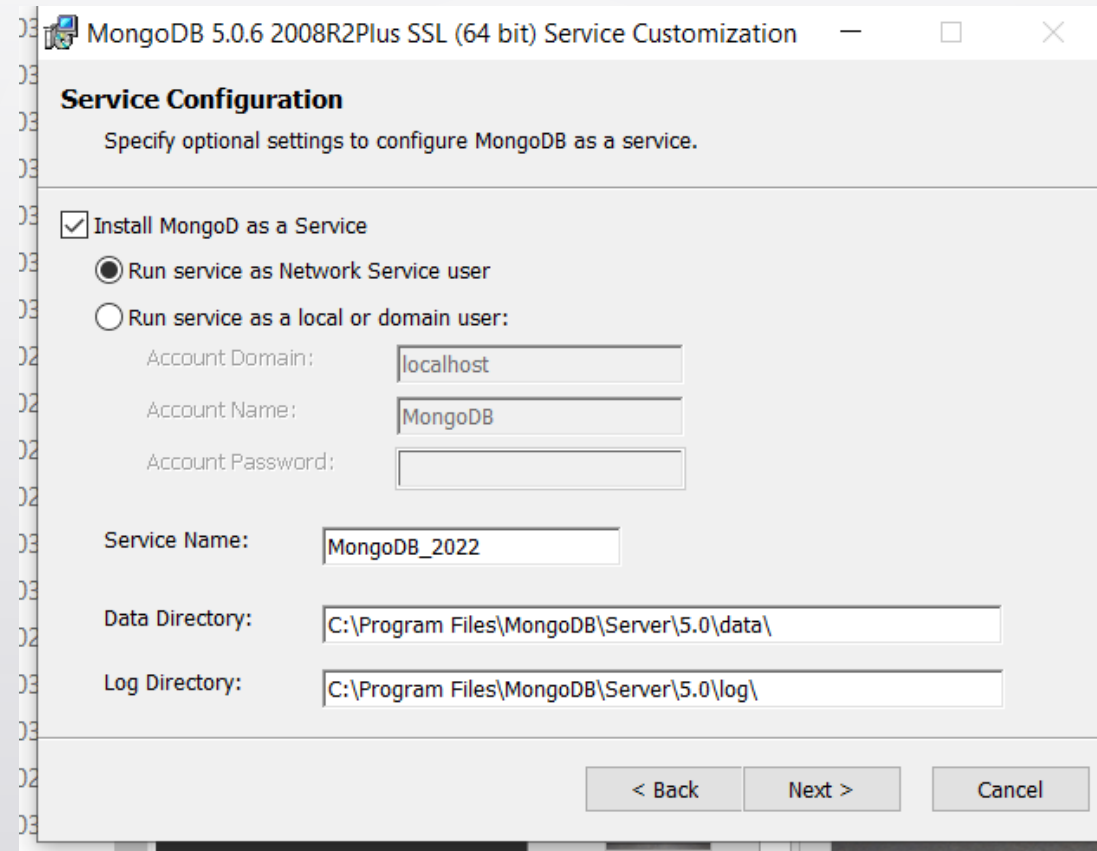
Iniciar instalación

- Empezar la instalación



Como un servicio

- No necesario. Servicio -> si uso frecuente



The screenshot shows the 'MongoDB 5.0.6 2008R2Plus SSL (64 bit) Service Customization' window. The 'Service Configuration' tab is active, with the instruction 'Specify optional settings to configure MongoDB as a service.' Below this, the 'Install MongoDB as a Service' checkbox is checked. Under the 'Run service as' section, 'Run service as Network Service user' is selected. The 'Run service as a local or domain user:' option is also visible, with fields for 'Account Domain:' (localhost), 'Account Name:' (MongoDB), and 'Account Password:'. The 'Service Name:' field contains 'MongoDB_2022'. The 'Data Directory:' is set to 'C:\Program Files\MongoDB\Server\5.0\data\' and the 'Log Directory:' is set to 'C:\Program Files\MongoDB\Server\5.0\log\'. At the bottom, there are '< Back', 'Next >', and 'Cancel' buttons.

MongoDB 5.0.6 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

☒ Install MongoDB as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain: localhost

Account Name: MongoDB

Account Password:

Service Name: MongoDB_2022

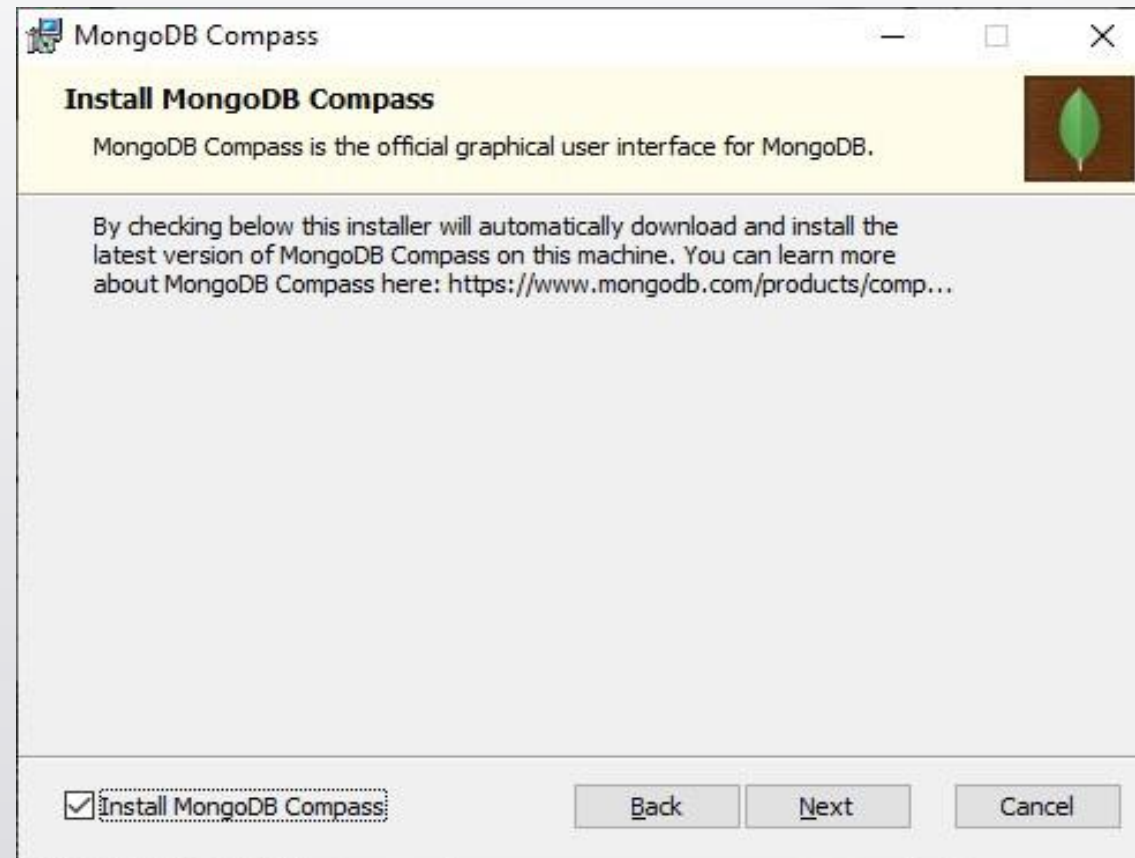
Data Directory: C:\Program Files\MongoDB\Server\5.0\data\

Log Directory: C:\Program Files\MongoDB\Server\5.0\log\

< Back Next > Cancel

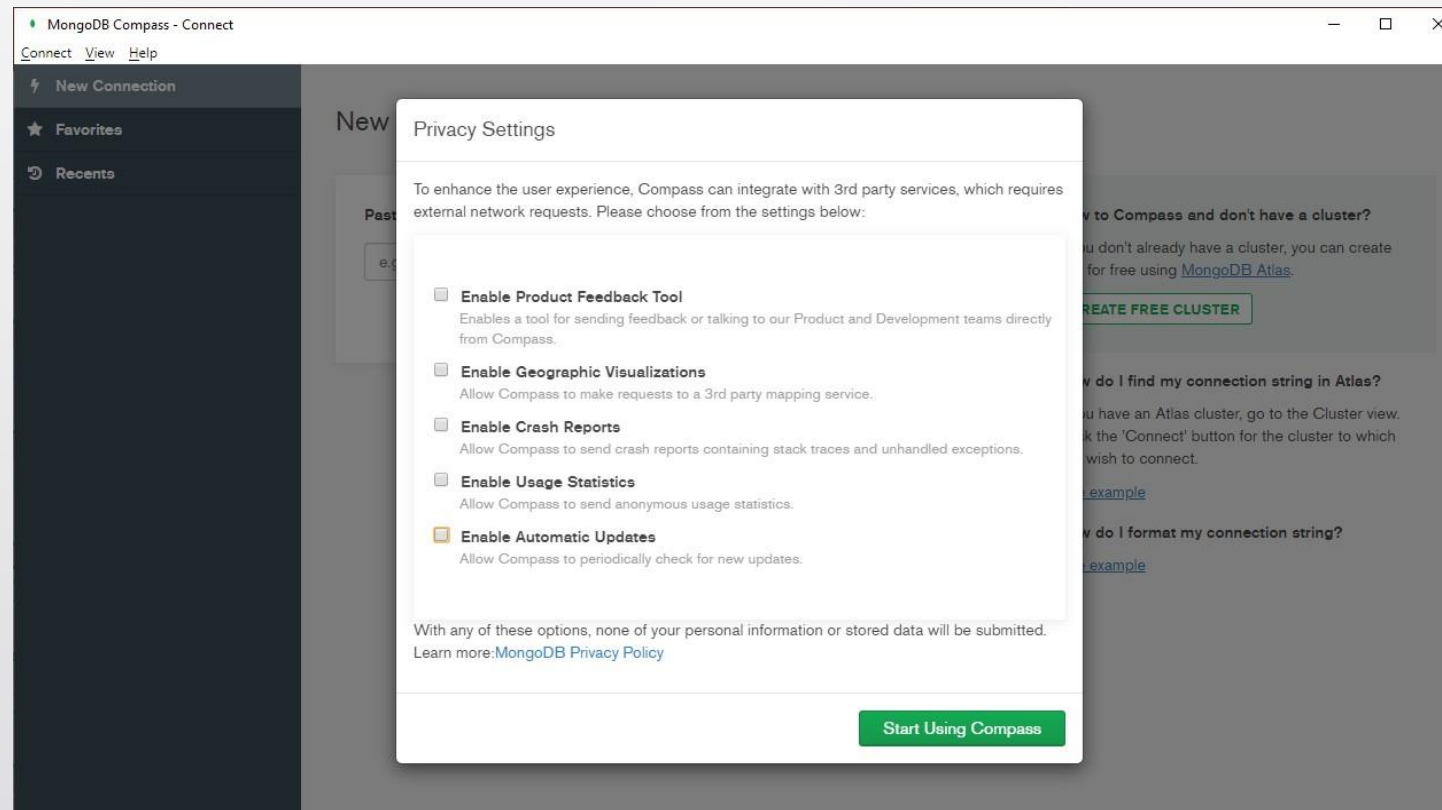
MongoDB Compass

- IG(Interfaz Gráfica) – alternativa MongoDB al uso de Interfaz Consola



Opciones de Compass

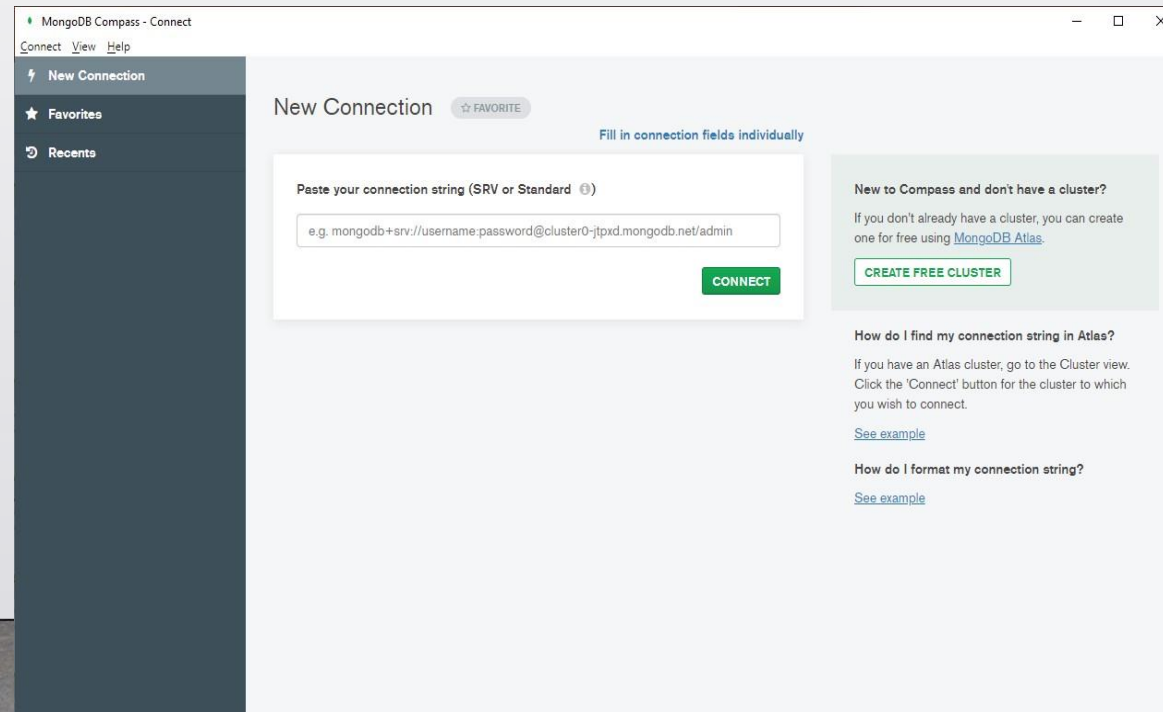
- Selección de las opciones de privacidad deseadas.



Servidor (I)

- No está el servidor funcionando. No se podrá conectar con él.
- Cerrar Compass.
- Instalación en:

C:\Users\usuario\AppData\Local\MongoDBCompass\MongoDBCompass.exe



//////////////////// Servidor (II) – Carpeta de datos

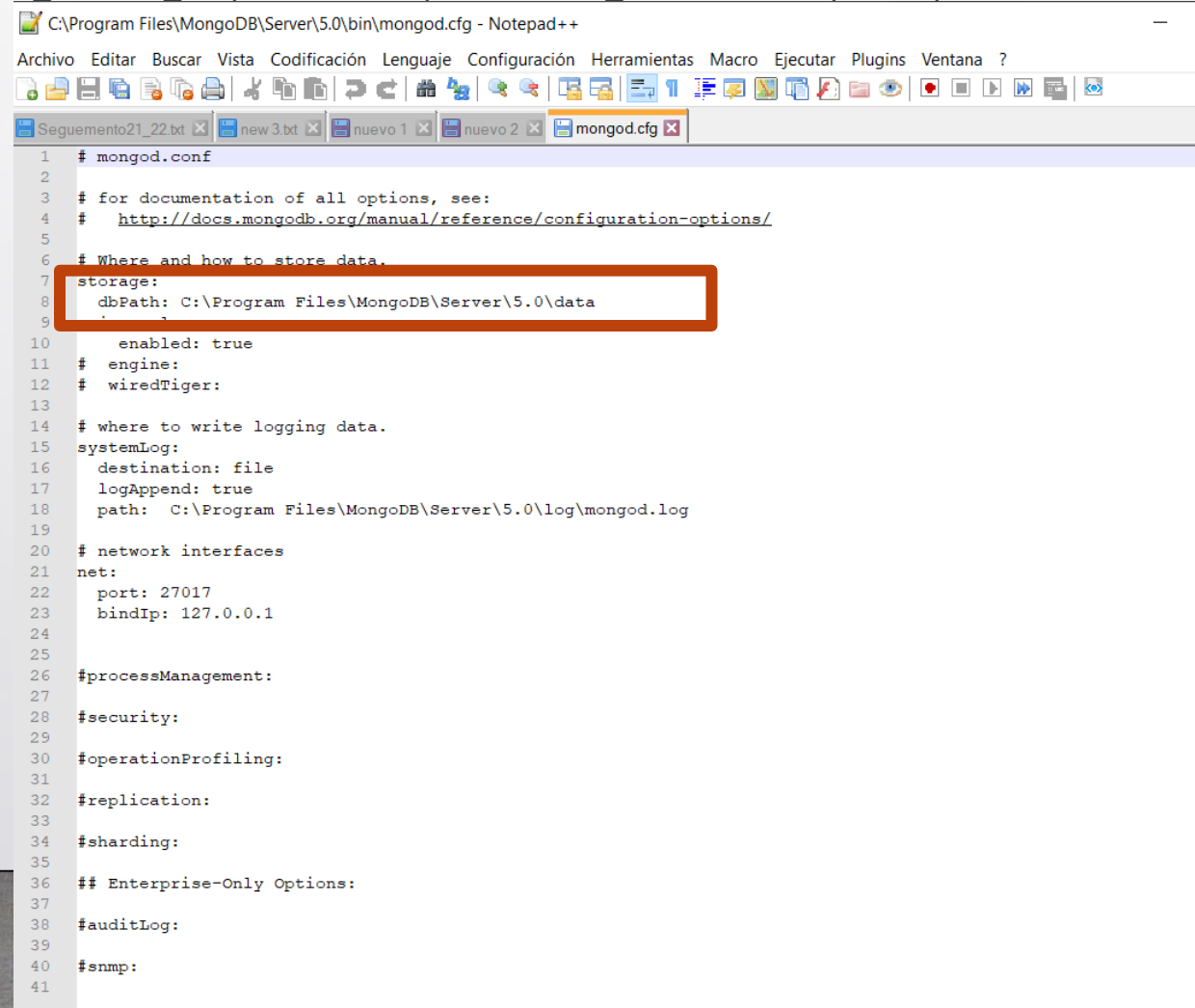
- ◆ MongoDB necesita una carpeta para guardar los datos.
- ◆ Creamos, por ejemplo, la carpeta z:\mongodb\data

```
Z:\>mkdir z:\mongodb\data
```

- ◆ También podría ser: C:\Program Files\MongoDB\Server\5.X\data

Servidor (III) – Archivo de configuración

- Editar el fichero “mongod.cfg” (Windows) o “mongod.conf” (Unix)



```
C:\Program Files\MongoDB\Server\5.0\bin\mongod.cfg - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
Seguemento21_22.txt  new3.txt  nuevo 1  nuevo 2  mongod.cfg
1  # mongod.conf
2
3  # for documentation of all options, see:
4  # http://docs.mongodb.org/manual/reference/configuration-options/
5
6  # Where and how to store data.
7  storage:
8    dbPath: C:\Program Files\MongoDB\Server\5.0\data
9
10    enabled: true
11  # engine:
12  # wiredTiger:
13
14  # where to write logging data.
15  systemLog:
16    destination: file
17    logAppend: true
18    path: C:\Program Files\MongoDB\Server\5.0\log\mongod.log
19
20  # network interfaces
21  net:
22    port: 27017
23    bindIp: 127.0.0.1
24
25
26  #processManagement:
27
28  #security:
29
30  #operationProfiling:
31
32  #replication:
33
34  #sharding:
35
36  ## Enterprise-Only Options:
37
38  #auditLog:
39
40  #snmp:
41
```

Servidor (IV) – Arranque con parámetros (.cmd/.bat)

- Creación fichero de procesoLotes (.bat/.cmd) al pasar dir como parámetro:

- >mongod --dbpath=ruta_base_de_datos

\$ mongod.exe

```
C:\Program Files\MongoDB\Server\4.4\bin>mongod.exe --dbpath z:\mongodb\data
{"t":{"$date":"2021-02-21T20:41:02.692+01:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically
disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2021-02-21T20:41:02.699+01:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportL
ayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-02-21T20:41:02.700+01:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP
FastOpen in use."}
{"t":{"$date":"2021-02-21T20:41:02.702+01:00"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"Mong
```

Cliente

\$ mongo.exe

- ▶ OJO: La cadenaConexión - ConnectionString
- ▶ mongo --localhost --port 27017 -u usuario -p password

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
MongoDB shell version v5.0.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5afd64b8-a28f-4956-a64d-ca22ef3306e7") }
MongoDB server version: 5.0.3
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
The server generated these startup warnings when booting:
  2022-01-01T16:43:13.323+01:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
MongoDB shell version v4.4.4
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7143178d-355a-4210-8544-70681f9bf77d") }
MongoDB server version: 4.4.4
---
The server generated these startup warnings when booting:
  2021-02-21T20:55:53.724+01:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
  2021-02-21T20:55:53.725+01:00: This server is bound to localhost. Remote systems will be unable to connect to th
is server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or wi
th --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to di
sable this warning
---
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```



Cliente

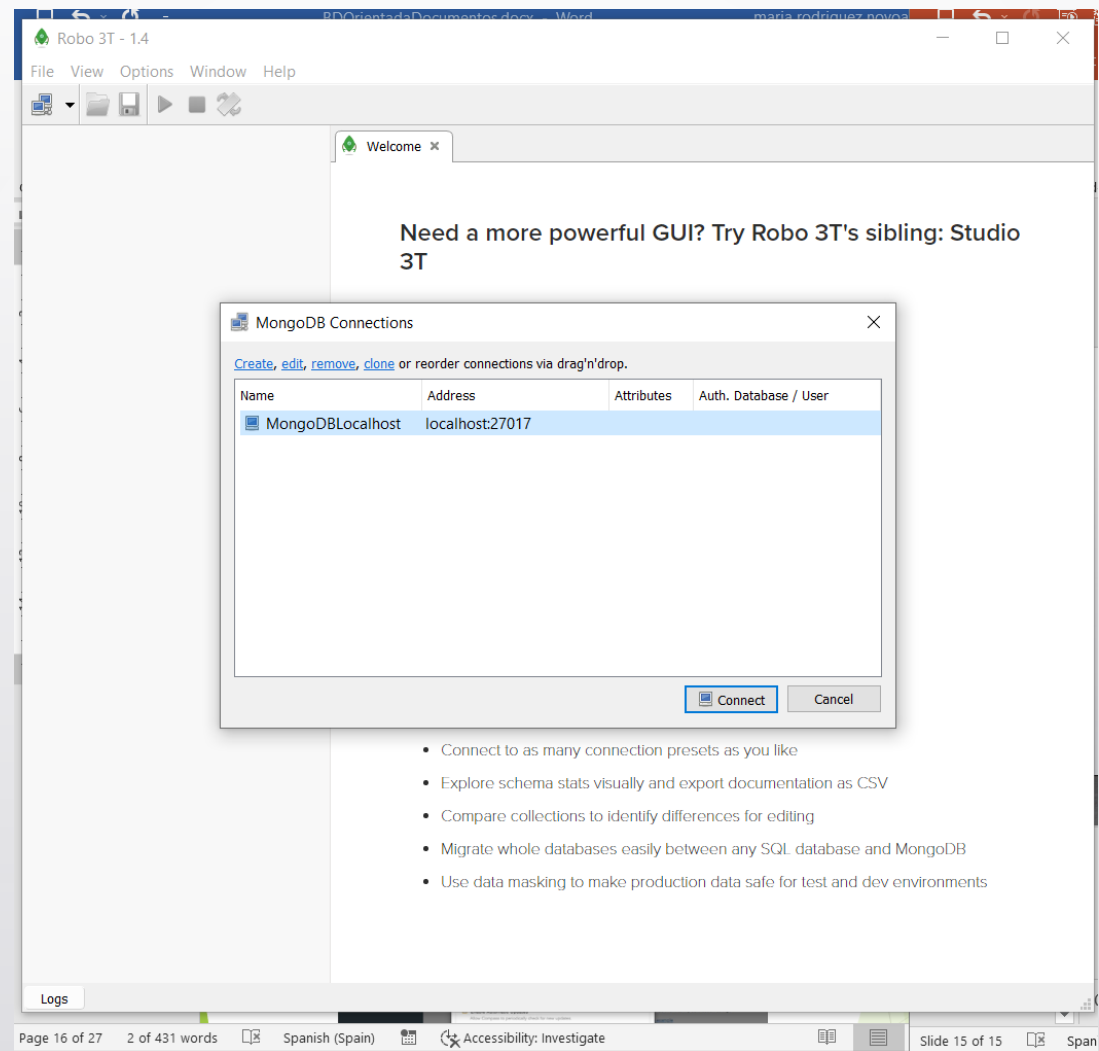
- show dbs
- use {bd}
- show collections
- db.dropDatabase()
- help



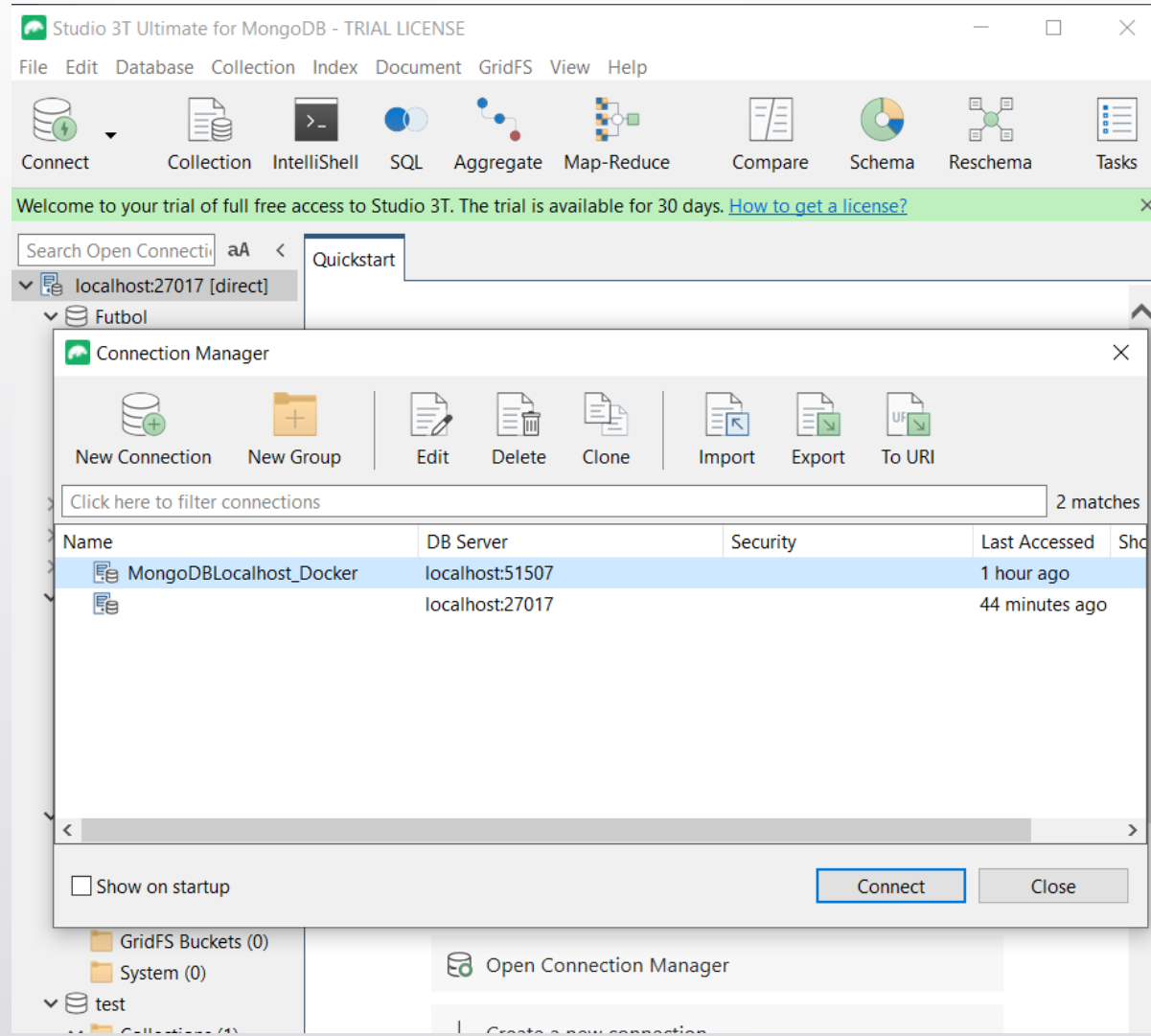
MongoShell

- Sustituirá el Cliente que se instala con el Servidor
- En futuras versiones será el usado por defecto.
- Mongosh
 - Instalador en aula virtual

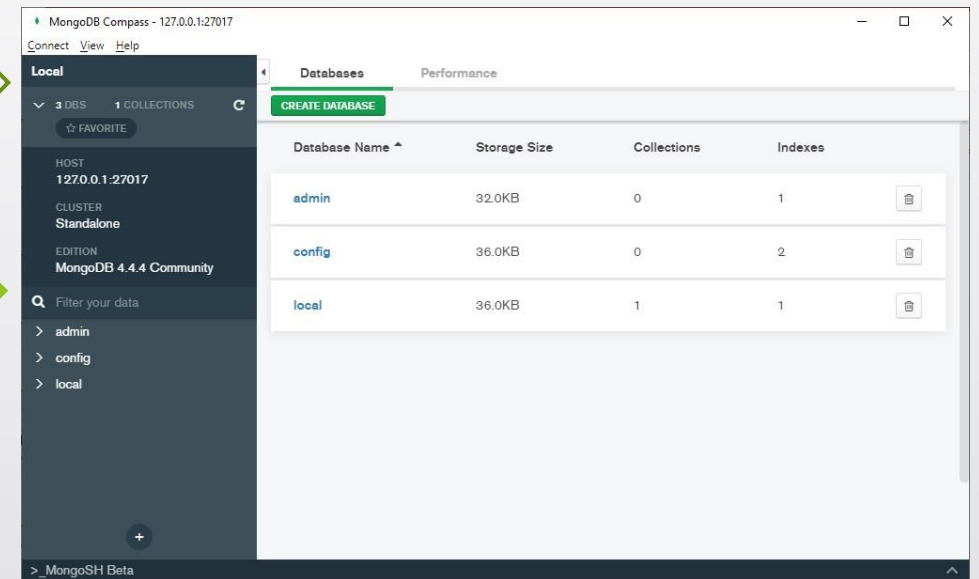
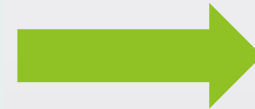
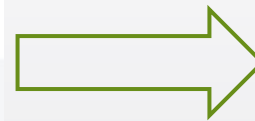
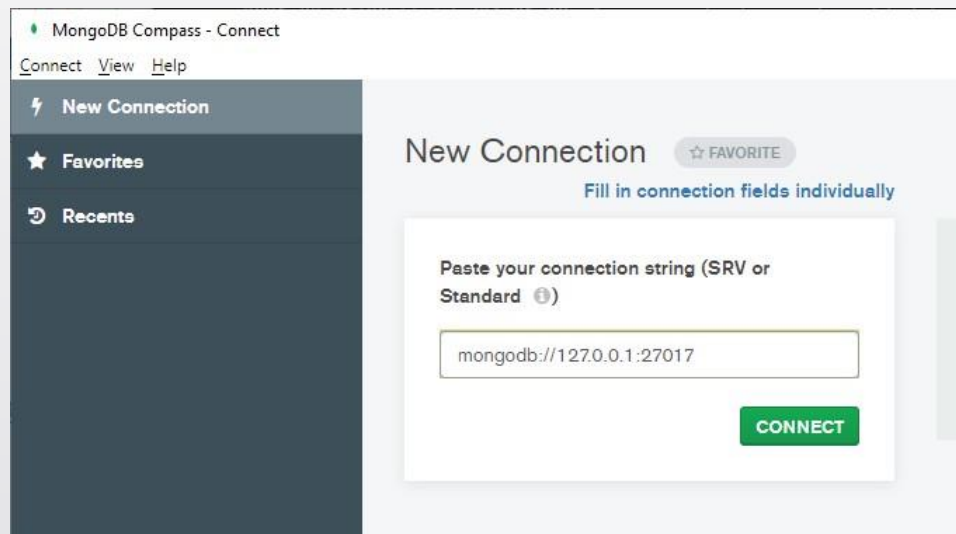
Robo 3T



Studio 3T



MongoDB Compass - Conexión





CRUD (Create, Read, Update, Delete)

- Crear registros (**Create**): `.save` / `.insert`
- Consulta/Lectura registros (**Read**): `.find()`
- Actualizar (**Update**): `.update`
- Borrar (**Delete**): `.remove` / `.drop`

////////////////////////////////////

CRUD (Create, Read, Update, Delete)

➤ Create

❑ insert

```
db.alumnos.insert({nombre: 'Isabel', telefono: '654858585', curso: '1º DAM', nota: 8});
```

```
db.alumnos.insert(
```

```
{ _id: 1, nombre: 'Pepe',
```

```
telefono: '654999999',
```

```
direccion: {calle: 'Avda. Galicia', numero: 33, ciudad: 'Vigo'},
```

```
curso: '2º DAM',
```

```
nota: 9});
```

❑ Save

```
db.alumnos.save({nombre: 'Eloy', telefono: 699333333, curso: '2º DAM', nota: 7});
```

////////////////////////////////////

CRUD (Create, Read, Update, Delete)

➤ Read

- `db.alumnos.find().sort({nombre:-1});`
- `db.alumnos.find().sort({nombre:1});`
- `db.alumnos.find({nombre:'Juan'});`
- `db.alumnos.find({nombre:'Juan'}, {telefono:1});`
- `db.alumnos.find({curso:'1º DAM'});`
- `db.alumnos.find({curso:'1º DAM'}, {nombre:1, nota:1});`
- `db.alumnos.find({}, {nombre:1, nota:1});`
- `db.alumnos.find({}, {_id:0, nombre:1, nota:1});`
- `db.alumnos.find({curso:'2º DAM'}).count();`
- `db.alumnos.find().pretty();`
- `db.alumnos.findOne();`

CRUD (Create, Read, Update, Delete)

➤ Read

□ Búsquedas de Comparación

- **\$eq, \$ne**
 - `db.alumnos.find({nota:{$ne:7}});`
- **\$gt, \$lt**
- **\$gte, \$lte**
 - `db.alumnos.find({nota:{$gte:7}});`
- **\$in, \$nin**
 - `db.alumnos.find({nota:{$in:[6,8,9]}});`
 - `db.alumnos.find({nota:{$nin:[6,8,9]}});`
- **\$type**
 - `db.alumnos.find({nota:{$type:2}}, {nombre:1, nota:1});`
 - `db.alumnos.find({nota:{$type:1}}, {nombre:1, nota:1});`

CRUD (Create, Read, Update, Delete)

➤ Read

❏ Búsqueda Lógicos

- \$or

```
db.alumnos.find({$or:[{nota:{$gt:8}},{curso:'1º DAM'}]});
```

- \$and

```
db.alumnos.find({$and:[{nota:7},{curso:'2º DAM'}]});
```

```
db.alumnos.find({nota:{$gte:7}},{curso:'2º DAM'});
```

- \$not - solo usarse con otros operadores \$gt o \$lt

```
db.alumnos.find({nota:{$not:{$gt:8}}});
```

- \$nor - acepta 2 o más valores

```
db.alumnos.find({$nor:[{nota:{$gt:8}},{curso:'1º DAM'}]});
```

- \$exist - busca la existencia de un campo

-- dos alumnos sin nota

```
db.alumnos.insert({nombre:'Alumno1',telefono:699111111,curso:'2º DAM'});
```

```
db.alumnos.insert({nombre:'Alumno2',telefono:699222222,curso:'2º DAM'});
```

```
db.alumnos.find({nota:{$exists:false}});
```

```
db.alumnos.find({nota:{$exists:true}});
```

CRUD (Create, Read, Update, Delete)

➤ Update

```
db.nombreColeccion.update(  
    filtroBusqueda, //condición localización registros/documentos  
    cambios_a_Realizar, //el resultado final del documento – los campos que no aparecen, se eliminan  
    {  
        upsert: booleano,  
        multi: booleano  
    }  
);
```

```
db.alumnos.update({nombre: "Ana"}, {nombre: "Ana María"})
```

- **\$set** - actualizar con nuevas propiedades

```
db.alumnos.update({nombre: "Beatriz"}, {$set: {nota: 8, direccion:"Vigo"}});
```

- **OPCIÓN multi** - cambio se realizará en todos los resultados

```
... "}}, {multi: true})
```

- **OPCIÓN upsert** - inserta el documento si este no existe. Se comprueba toda la consulta en lugar de solo el *_id*.

- **\$unset** - eliminar propiedades de un documento

```
db.alumnos.update({nombre: "Ana María"}, {$unset: {direccion:"Bilbao"}}, {multi: true})
```

- **\$inc**

```
db.alumnos.update({nombre: "Ana María"}, {$inc: {nota: 2}})
```

////////////////////////////////////

CRUD (Create, Read, Update, Delete)

➤ Delete

- **remove({nombre: valor})** - borra un documento que cumpla una condición
- **remove()** – borrar todos los elementos de la colección
- **drop()** – borra la colección

```
db.alumnos.remove({telefono:'654858585'}); //elimina doc con ese teléfono
```

Operaciones con Arrays

- **Crear**

En el ej. de la colección libros, habrá un array con los **temas**:

```
db.libros.insert({codigo: 1, titulo: 'Código Limpio', precio: 35, editorial: 'Anaya', autor: 'Robert C. Martín',  
temas: ['Nombres con sentido', 'Funciones', 'Comentarios', 'Formato']})
```

```
db.libros.insert({codigo: 2, titulo: 'Patrones de Diseño en Java', precio: 32, editorial: 'Eni', autor: 'Laurent Debrauwer', temas:  
['Patrones de construcción', 'Patrones de construcción', 'Patrones de comportamiento']})
```

```
db.libros.insert({codigo: 3, titulo: 'Java y Eclipse', precio: 40, editorial: 'Eni', autor: 'Frédéric Delechamp',  
temas: ['Diseño', 'Conexion', 'MVC']})
```

- **Modificar**

- **\$set** – modificar un campo
- **\$push** – añade elto. a un array
- **\$addToSet** – agrega eltos. a un array solo si estos no existen
- **\$each** – con los 2 anteriores para añadir varios
- **\$pop** – elimina primer (-1) u (otro) último
- **\$pull** – elimina el q cumpla el filtro

```
db.libros.update({}, {$pull:{temas: {$in: ["Diseño", "XML"]}}},{multi: true})
```

- **\$pullAll** – todos los eltos.

Funciones Agregado (AggregationFramework)

Tipos

FUNCIONES ARITMÉTICAS	
Función	Descripción
\$abs	Devuelve el valor absoluto de un número
\$add	Añade números a una cantidad o a una fecha, en este caso suma milisegundos
\$ceil	Devuelve el entero menor, mayor o igual que el número especificado
\$divide	Devuelve el resultado de dividir el primer número por el segundo. Tiene 2 argumentos.
\$floor	Devuelve el entero mayor, menor o igual que el número especificado
\$mod	Devuelve el resto de dividir el primer número por el segundo. Tiene 2 argumentos.
\$multiply	Multiplica varios números, acepta varios argumentos
\$pow	Eleva un número a la potencia indicada
\$sqtr	Calcula la raíz cuadrada
\$subtract	Devuelve el resultado de restar el primer número menos el segundo. Si los dos valores son números, devuelve la diferencia. Si los valores son fechas, devuelve la diferencia en milisegundos
\$trunc	Trunca un número

FUNCIONES DE CADENAS	
Función	Descripción
\$concat	Concatena varias cadenas, las que se pongan en la expresión
\$substr	Devuelve una subcadena de una cadena, a partir de una posición indicada hasta una longitud especificada. Tiene 3 argumentos, la cadena, la posición de inicio y la longitud
\$toLower	Convierte una cadena a minúsculas
\$toUpper	Convierte una cadena a mayúsculas
\$strcasecmp	Compara cadenas y devuelve 0 si las dos cadenas son equivalentes, 1 si la primera cadena es mayor que la segunda y -1 si la primera cadena es menor que la segunda.

FUNCIONES DE GRUPO	
Función	Descripción
\$sum	Devuelve la suma de los valores numéricos. Ignora los valores no numéricos.
\$avg	Devuelve la media de los valores numéricos. Ignora los valores no numéricos.
\$first	Devuelve el primer valor del grupo.
\$last	Devuelve el último valor del grupo
\$max	Devuelve el valor máximo de un grupo o de un array.
\$min	Devuelve el valor mínimo de un grupo o de un array.

FUNCIONES DE FECHA	
Función	Descripción
\$dayofYear	Devuelve el día del año. Un número entre 1 y 366
\$dayOfMonth	Devuelve el día del mes. Un número entre 1 y 31
\$dayOfWeek	Devuelve el día de la semana. Un número entre 1 (Domingo) y 7 (Sábado)
\$year	Devuelve el año en formato yyyy, por ejemplo 2019
\$month	Devuelve el número del mes entre 1 (Enero) y 12 (Diciembre)
\$hour	Devuelve la hora entre 0 y 23
\$minute	Devuelve los minutos entre 0 y 59
\$second	Devuelve los segundos entre 0 y 59
\$dateToString	Devuelve la fecha en formato String

Funciones Agregado

➤ Pipeline

Etapa	Descripción	Multiplicidad
\$project	Cambia la forma del documento. La proyección permite modificar la representación de los datos, por lo que en general se emplea para darles una nueva forma con la que resulte más cómodo trabajar	1:1
\$match	Filtra los resultados. La etapa match permite filtrar los documentos para que en el resultado de la etapa solo estén aquellos que cumplen ciertos criterios. Se puede filtrar antes o después de agregar los resultados, en función del orden en que definamos esta etapa.	n:1
\$group	Agrupación. Permite agrupar distintos documentos según compartan el valor de uno o varios de sus atributos, y realizar operaciones de agregación sobre los elementos de cada uno de los grupos. Se utilizan las funciones: su, max, min, avg, etc.	n:1
\$sort	Ordenación de documentos	1:1
\$skip	Salta N elementos	n:1
\$limit	Elige N elementos para el resultado	n:1
\$unwind	Normalizar arrays	1:n
\$out	Envía el resultado a una salida, se almacena en la BD como una nueva colección	1:1

Funciones Agregado

➤ Formato:

```
db.nombreColeccion.aggregate([
  {
    $etapa1: {
      .....
    }, {
    $etapa2: {
      .....
    }
  }, .....
])
```

//Obtener los títulos de los libros y la editorial en mayúsculas.

```
db.libros.aggregate( [ { $project: { titulo: {$toUpper: "$titulo"},editorial: {$toUpper: "$editorial"}}} ])
```

```
db.articulos.aggregate(
[ {$project:
{
artículo: {$toUpper: "$denominacion"},
importe: {$multiply: ["$pvp", "$uv"]},
stockactual: {$subtract: ["$stock", "$uv"]},
areponer: {
$cond: [{ $lte: [{ $subtract: ["$stock", "$uv"]}, 0]}, true,
false}}}}])
```

Nombre	Descripción
\$cond	Este operador evalúa una expresión y dependiendo del resultado, devuelve el valor de una de las otras dos expresiones. Recibe tres expresiones en una lista ordenada o tres parámetros con nombre. Formato: <i>{ \$cond: [<boolean-expresión>, <caso-true>, <caso-false>] }</i>
\$ifnull	Devuelve o bien el resultado no nulo de la primera expresión o el resultado de la segunda expresión si la primera expresión da como resultado nulo. Acepta dos expresiones como argumentos. El resultado de la segunda expresión puede ser nulo. Formato: <i>{ \$ifnull: [<expresión>, <expresión-es- null>] }</i>

1º) Obtener la denominación en mayúsculas, el importe de las ventas (precio * uv), y el stock actual que será el stock menos las unidades vendidas uv; además si el stock actual es negativo, asignaremos a un nuevo campo llamado areponer true si es menor que 0 y false si no lo es.

Utilizaremos la etapa **\$project**, la función **\$multiply** para (precio * uv), **\$subtract** para restar al stock uv, y **\$cond** para establecer la condición.

Relaciones entre documentos

➤ Uso Referencias

▣ Manuales

Se guarda el campo **_id** de un documento como referencia en otro documento.

El concepto es similar al de clave ajena en el modelo relacional.

En este método la aplicación debe ejecutar una segunda consulta para devolver los datos relacionados.

Este es el método más utilizado.

▣ BDEmpresa(departamentos,empleados)

- `departrabajo = db.departamentos.findOne({_id: "dep1"})`
- Recuperamos los empleados cuyo `_id` se encuentre enlazado a este departamento(departrabajo en el ejemplo)
- `empleadosDepartamento = db.empleados.find({_id: {$in: departrabajo.emple }})`

Prácticas Colecciones con JS - MongoDB

- Usando colecciones ejemplo, gestionar los ejercicios propuestos en:

- Consola/Mongosh

- App Gestión Mongo (Studio3T)

genera archivo .js

The screenshot displays the Studio 3T Ultimate for MongoDB interface. The left sidebar shows the database structure with 'centroeducativo' selected, containing a collection named 'alumnos'. The main editor area shows a JavaScript script for managing the 'alumnos' collection. A red arrow points from the 'genera archivo .js' text to the 'Script_EjColecciones_Consultas.js' tab. The bottom panel shows the 'alumnos' collection data in a table view.

```
167 db.alumnos.find({nota:{type:2}}, {nombre:1, nota:1});
168 db.alumnos.find({nota:{type:1}}, {nombre:1, nota:1});
169
170 --Selectores Búsquedas Lógicas
171 db.alumnos.find({$or:[{nota:{gt:8}}, {curso:'1º DAM'}]});
172 db.alumnos.find({$and:[{nota:7}, {curso:'2º DAM'}]});
173 db.alumnos.find({nota:{gte:7}}, {curso:'2º DAM'});
174 db.alumnos.find({nota:{not:{gt:8}}});
175
176 db.alumnos.find({$nor:[{nota:{gt:8}}, {curso:'1º DAM'}]});
177
178 -- dos alumnos sin nota
179 db.alumnos.insert({nombre:'Alumno1', telefono:699111111, curso:'2º DAM'});
180 db.alumnos.insert({nombre:'Alumno2', telefono:699222222, curso:'2º DAM'});
181 db.alumnos.find({nota:{exists:false}});
182 db.alumnos.find({nota:{exists:true}});
183
184 db.alumnos.save({nombre:'Eloy', telefono:699333333, curso:'2º DAM', nota:7});
185
186
```

_id	nombre	telefono	curso	nota	direccion
625aff577358c5...	Lucia	619858585.0	1º DAM	ocho	
625aff577358c5...	Patricia	623585858.0	1º DAM	nueve	
625bd8257358c...	Isabel	654858585	1º DAM	8.0	
6251.0	Pepe	654999999	2º DAM	9.0	{ 3 fields }
625bfb6b7358c5...	Eloy	699333333.0	2º DAM	7.0	

Práctica Colección Empleados - MongoDB

Crea la colección *empleados* dentro de la base de datos *mibasedatos*, y añade los siguientes registros:

```
Emp_no:1,nombre:"Juan",dep:10, salario:1000, fechaalta:"10/10/1999"  
Emp_no:2,nombre:"Alicia",dep:10, salario:1400, fechaalta:"07/08/2000",  
oficio: "Profesora"  
Emp_no:3,nombre:"María Jesús",dep:20, salario:1500, fechaalta:  
"05/01/2005", oficio: "Analista", comisión:100  
Emp_no:4,nombre:"Alberto",dep:20, salario:1100, fechaalta:"15/11/2001"  
Emp_no:5,nombre:"Fernando",dep:30, salario:1400, fechaalta:  
"20/11/1999", comisión:200, oficio: "Analista"
```

Realiza las siguientes consultas:

- Visualiza los empleados del departamento 10.
- Visualiza los empleados del departamento 10 y 20.
- Obtén los empleados con salario >1300 y oficio Profesora.
- Sube el salario a los analistas en 100€, a todos los analistas.

Práctica ColecciónLibros - MongoDB

Utilizando la colección libros realiza las siguientes consultas:

- Visualiza los libros de la editorial Garceta, con pvp entre 20 y 25 incluidos y que tengan el tema SOCKET.
- Agrega el tema SOCKET a los libros que no lo tengan.
- Baja a 5 el precio de los libros de la editorial Garceta.

Práctica Colección Trabajadores - MongoDB

Utilizando la colección trabajadores realiza las siguientes consultas:

- Visualiza la edad media, la media de salario y el número de trabajadores que hayan tenido una prima de 30 o de 80.
- Visualiza por población el número de trabajadores, el salario medio y el máximo salario.
- Visualiza el nombre, ape1 y ape2 del empleado que tiene máximo salario.
- A partir de la consulta anterior, obtén ahora el nombre, ape1, ape2 y salario del empleado que tiene máximo salario por cada población:

Práctica SucursalesCuentas - MongoDB

Utilizando la transformación del documento *sucursales.xml* a JSON, realizada en los apartados anteriores, se pide crear las colecciones *cuentas* y *sucursales*. Para crearlas primero crea la colección *cuentas* y luego crea la colección *sucursales* asignando a las sucursales las cuentas correspondientes.

Una vez creadas las colecciones realiza las siguientes consultas:

- Visualiza las cuentas de las sucursales de Madrid.
- Visualiza las cuentas con *saldohaber* > 10000 cuyo director sea Fernando Rato
- Sube 300 el *saldohaber* de las cuentas de la sucursal con código SUC1 .



Java - MongoDB

➤ Maven

```
<dependency>  
  <groupId>org.mongodb</groupId>  
  <artifactId>mongo-java-driver</artifactId>  
  <version>3.11.1</version>  
</dependency>
```

➤ Ant

mongo-java-driver-3.11.1.jar

Java - MongoDB

➤ BSON

- Formato serialización **binaria**
 - Almacenar documentos
 - Hacer llamada procedimientos

➤ Tipos Datos **\$type**

Tipo	Número	Alias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
Symbol	14	"symbol"
Timestamp	17	"timestamp"

Java - MongoDB

➤ Conexión BD

```
MongoClient cliente = new MongoClient(); // nos conectamos a MongoDB
MongoDatabase db = cliente.getDatabase("centroeducativo"); // nos conectamos a la base de datos
MongoCollection <Document> coleccion = db.getCollection("alumnos"); //obtenemos la coleccion
```

➤ Visualizar Datos Colección

```
// MongoClient es una interfaz genérica:
//y es el tipo predeterminado para devolver la búsqueda (.find) y agregados (.aggregate).
//El método de un solo argumento getCollection devuelve una instancia de MongoCollection <Document>
//el método .find().into()

List <Document> consulta = coleccion.find().into(new ArrayList <Document>());
//recorremos la lista
for(int i = 0; i < consulta.size(); i++) { //mostramos los documentos
    System.out.println(" ---- "+consulta.get(i).toString());
}

-----

Document alumno = consulta.get(i); //recuperamos un documento

System.out.println(alumno.getString("nombre")+"\t"+ //recuperar los valores de los campos del documento, utilizando los métodos get del objeto Document
    alumno.get("telefono")+"\t"+alumno.getString("curso"));
```

Java - MongoDB

➤ Insertar Documentos

- insertOne

```
Document alumno = new Document(); //Creamos el documento
alumno.put("nombre", "Jorge"); alumno.put("telefono", 121212); //le añadimos los campos
alumno.put("curso", "2º ASIR"); alumno.put("fecha", new Date());
coleccion.insertOne(alumno); //insertamos el documento en la colección

Document alumno2 = new Document("nombre", "Marisa", "teléfono", 1234, "cursos", "1º DAM", "2º DAM");
    .append("teléfono", 1234)
    .append("curso",
        new Document("curso1", "1DAM").append("curso2", "2DAM"));
coleccion.insertOne(alumno2);
```

- insertMany

```
for(int i=0; i< 5; i++) {listadocs.add(new Document("Valor de i", i));}
coleccion.insertMany(listadocs);
```

Java - MongoDB

➤ Consultar Documentos

`consulta.get(i)`

`Document doc = consulta.get(i); System.out.println(doc.getString("nombre")+"\t"+ doc.get(...)`

`MongoCursor <Document> cursor = coleccion.find().iterator();.. Document doc = cursor.next();
System.out.println(doc.toJson());`

❑ Filtros en Consultas

- **import** `com.mongodb.client.model.Filters;`

❑ Uso Proyecciones (para cambiar salidas-no usar todos datos documento-objeto **FindIterable**)

- **import static** `com.mongodb.client.model.Projections.*;`

Ade+ **first()**, la clase **FindIterable** - métodos ordenación (**sort()**), proyección (**projection()**), salto (**skip()**) y limitación (**limit()**).

❑ Uso Agregaciones (objeto **AggregationIterable**. Se puede recorrer con cursor)

Método **aggregate()** – con *pipeline* de operaciones (**\$match**, **\$project**, **\$group**, **\$sort**, etc) en forma de lista.

Java - MongoDB

➤ Actualizar Documentos

`updateOne()`, `updateMany()`, y `findOneAndUpdate()`. `findAndUpdate()` //parámetros: el documento con los criterios, la actualización y de forma optativa las opciones de actualización

➤ Borrar Documento de Colección

```
coleccion.deleteOne(findDocument);
```

```
DeleteResult deleteResult = coleccion.deleteMany(findDocument);
```

➤ Crear/Borrar Colección

```
db.createCollection("nuevaColeccion");
```

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

```
coleccion.drop(); //borrar la coleccion
```

➤ Listar Colecciones BD

```
(MongoIterable) db.listCollectionNames();
```

➤ Crear, Listar y Borrar BDs

```
MongoDatabase db = cliente.getDatabase("bd");//bd no se creará hasta que no se inserte un documento
```

```
cliente.getDatabase("bd").drop(); // nos conectamos a la base de datos y la borramos
```

```
MongoCollection <Document> colnueva =db.getCollection("colec nueva"); //creamos la coleccion
```

```
Document doc1 = new Document("nombre","Beatriz").append("telefono", 123).append("ciudad", "Vigo"); //crear un documento
```

```
colnueva.insertOne(doc1); //y añadirlo a la coleccion
```

➤ Datos MongoDB a/de FicheroDatos -MongoDB_vs_JSON.java

Práctica Java - MongoDB

// PASO 1: Conexión al Server de MongoDB Pasandole el host y el puerto

```
MongoClient mongoClient = new MongoClient("localhost", 27017); // MongoClient mongoClient = new MongoClient();
```

// PASO 2: Conexión a la base de datos

```
MongoDatabase db = mongoClient.getDatabase("NombreDB");
```

// PASO 3: Obtenemos una coleccion para trabajar con ella

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

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

// PASO 4: CRUD (Create-Read-Update-Delete)

// PASO 4.1: "CREATE" -> Metemos los objetos

// PASO 4.2.1: "READ" -> Leemos todos los documentos de la base de datos

// PASO 4.2.2: "READ" -> Hacemos una Query con condiciones

// PASO 4.3: "UPDATE" -> Actualizamos

// PASO 4.4: "DELETE" -> Borramos

// PASO FINAL: Cerrar la conexión

Práctica POJOs - CodecRegistry - MongoDB

//Gestión del CodecRegistry para hacer uso de los objetos definidos

```
CodecRegistry pojoCodecRegistry = CodecRegistries.fromRegistries(MongoClient.getDefaultCodecRegistry(),  
    CodecRegistries.fromProviders(PojoCodecProvider.builder().automatic(true).build())  
    );
```

// Conexión con el servidor Mongo

```
try (MongoClient cliente = new MongoClient("localhost",  
    MongoClientOptions.builder().codecRegistry(pojoCodecRegistry).build())  
    ){.....}
```

//Realizar mismas operaciones que en el caso anterior Sin CodecRegistry



ODM

- ORM use a SQL database Driver
 like ODBC, JDBC or OLEDB to translate the object notation - to **relational** notation
- and ODM use a JSON or JSONB api to translate the Object notation - to **Document** notation.
- There are different kind of implementations under the hood.
- Ejemplos
 - Mongoose
 - [mongodb](#) object modeling for [node.js](#)
 - Morphia
 - Object Document Mapper (ODM) for MongoDB in Java.



Práctica Morphia - MongoDB

- *Maven*

```
<dependency>  
  <groupId>dev.morphia.morphia</groupId>  
  <artifactId>core</artifactId>  
  <version>1.5.3</version>  
</dependency>
```

Práctica Morphia - MongoDB

- Creación

```
Morphia morphia = new Morphia();  
morphia.mapPackage("morphia");//paquete en que está el .java  
datastore = morphia.createDatastore(new MongoClient(), "libraryMorphia");  
datastore.ensureIndexes();
```

- Inserción

```
Publisher publisher = new Publisher(id, "Awsome Publisher");  
datastore.save(new Book("9781565927186", "Learning Java", "Tom Kirkman", 3.95, publisher));  
datastore.save(new Book("9781449313142", "Learning Perl", "Mark Pence", 2.95, publisher));  
datastore.save(new Book("9787564100476", "Learning Python", "Mark Pence", 5.95, publisher));  
datastore.save(new Book("9781449368814", "Learning Scala", "Mark Pence", 6.95, publisher));  
datastore.save(new Book("9781784392338", "Learning Go", "Jonathan Sawyer", 8.95, publisher));
```

```
Iterator<Author> authors = datastore.createAggregation(Book.class)  
    .group("author", grouping("books", push("title")))  
    .out(Author.class);
```



Más información

- <https://docs.mongodb.com/guides/>