

Bases de datos no estructuradas

Unidad 2: Operaciones esenciales con MongoDB

Descargable

Índice

- I. Etapas del proceso de instalación de MongoDB y principales comandos de gestión
- II. Operaciones CRUD en documentos y subdocumentos

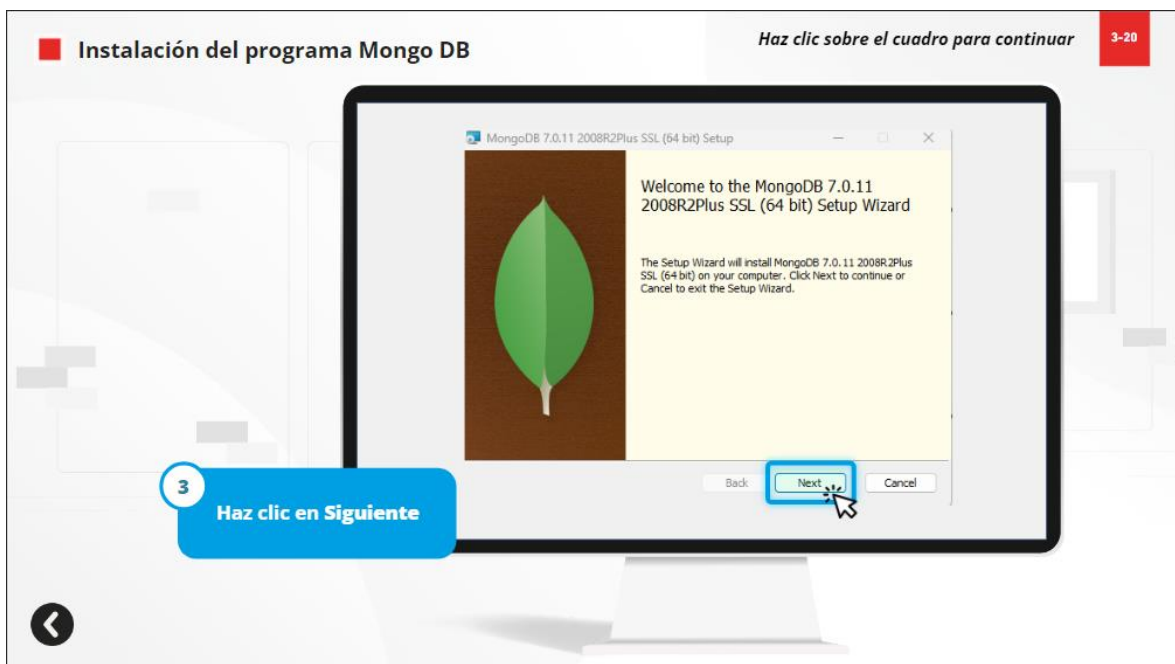
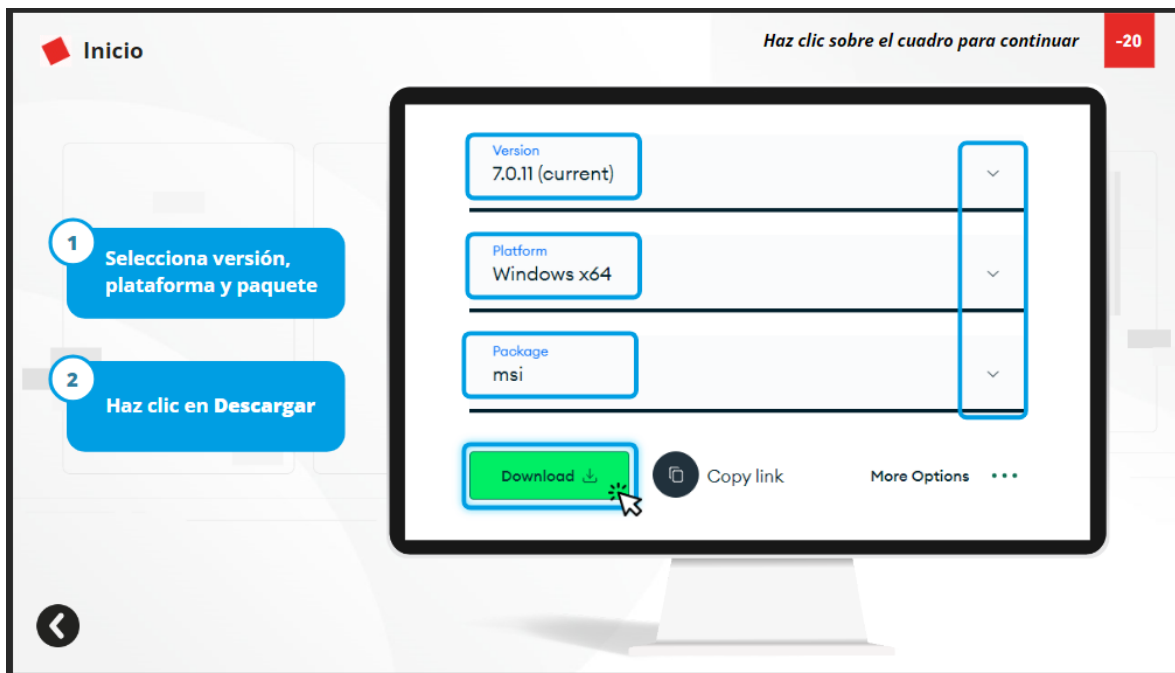
Introducción

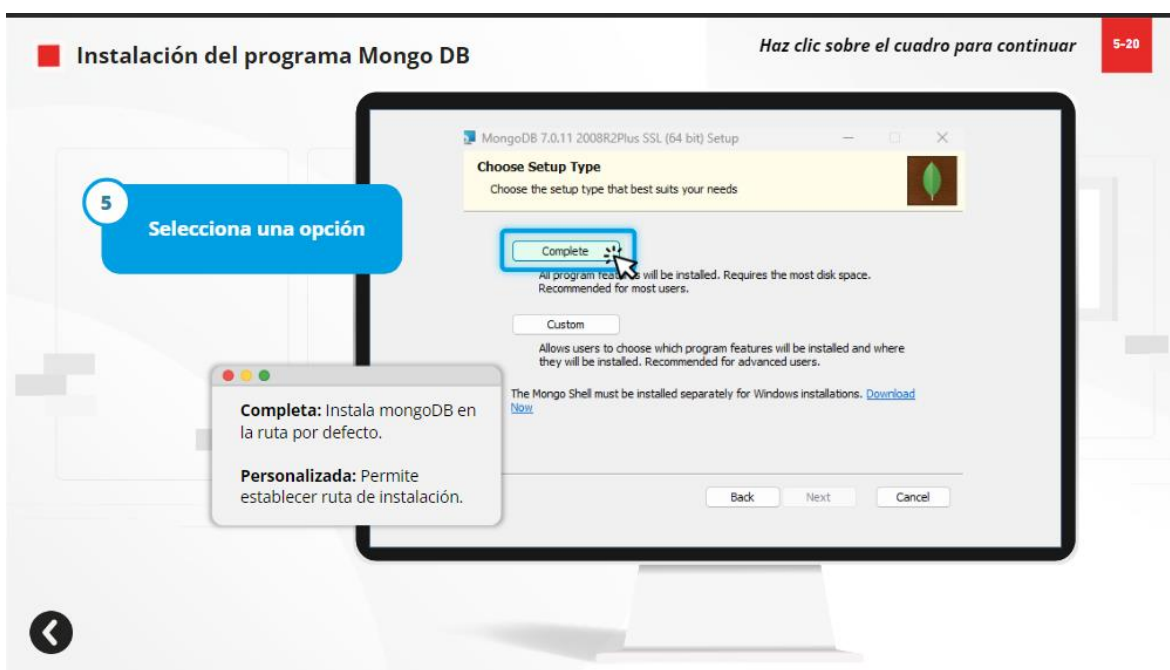
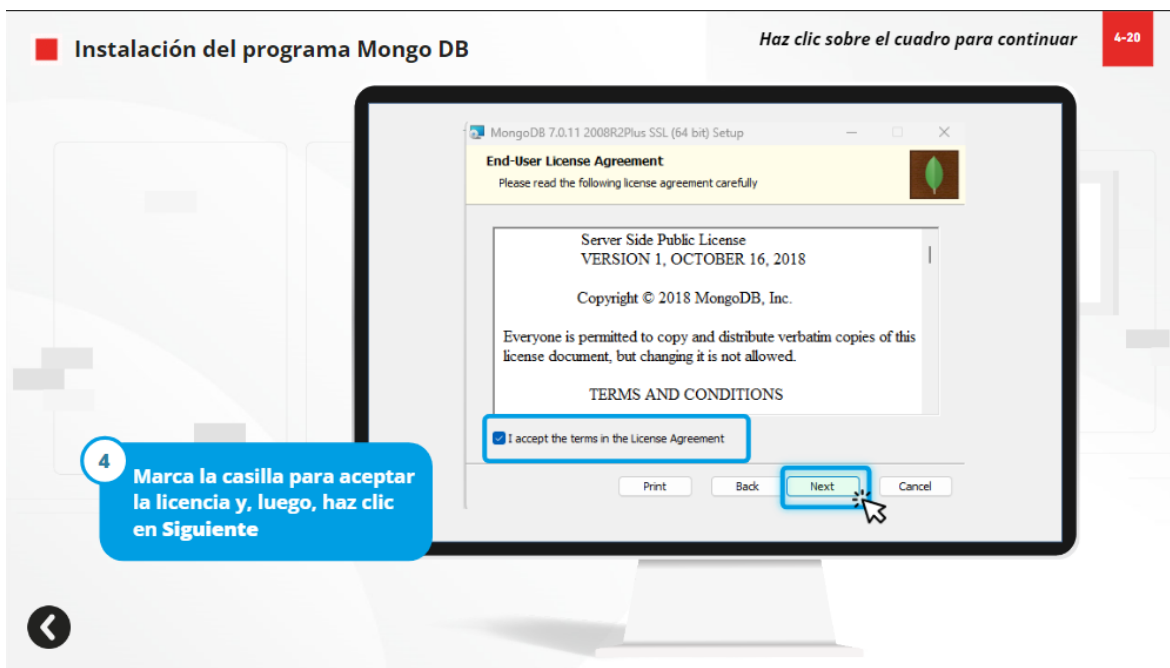
En esta unidad, explorarás cómo descargar e instalar la base de datos MongoDB y, además, conocerás sus principales comandos. También aprenderás cómo ejecutar operaciones CRUD (Creación, Búsqueda, Modificación y Eliminación) en documentos y subdocumentos, además de ver el uso de arreglos con sus diferentes métodos.

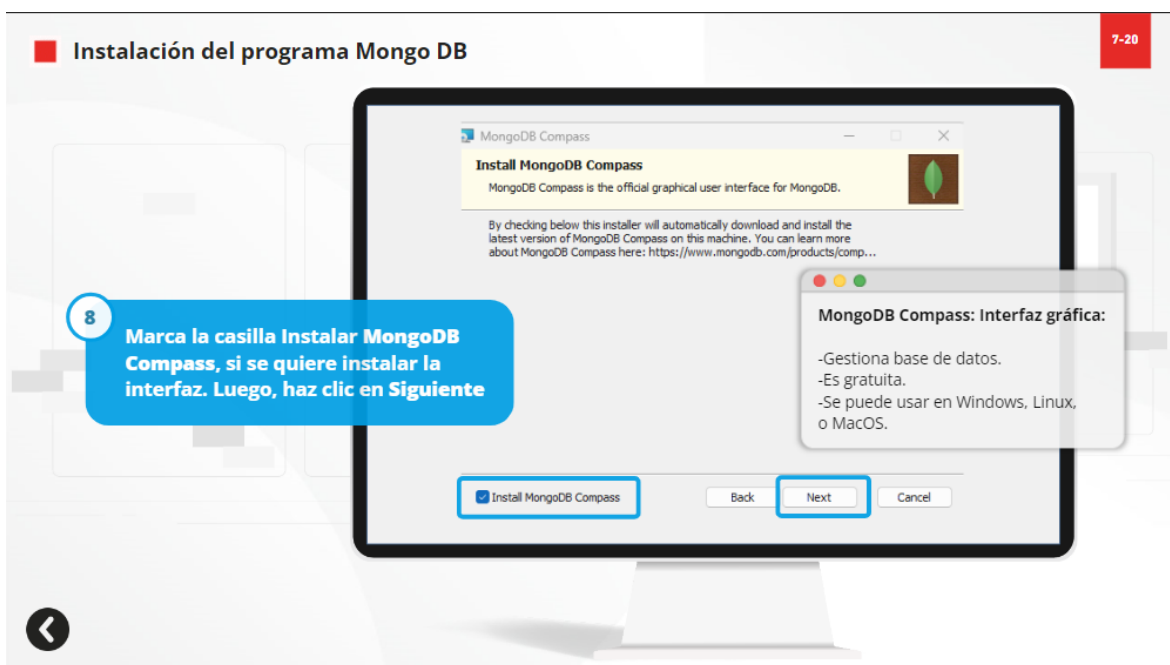
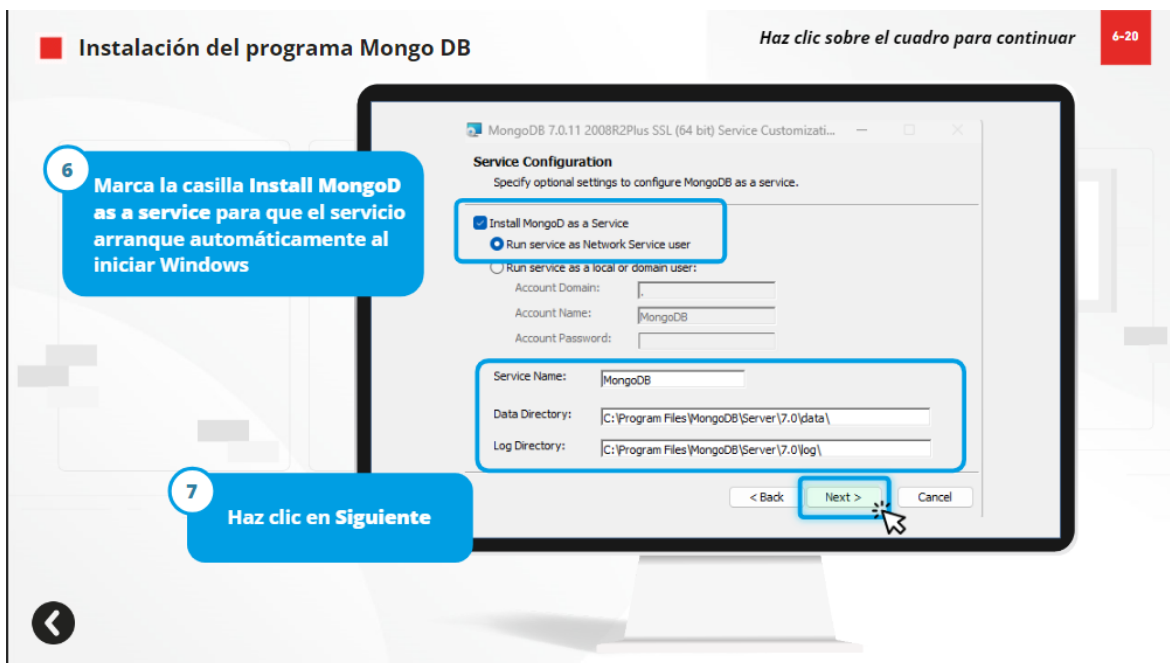


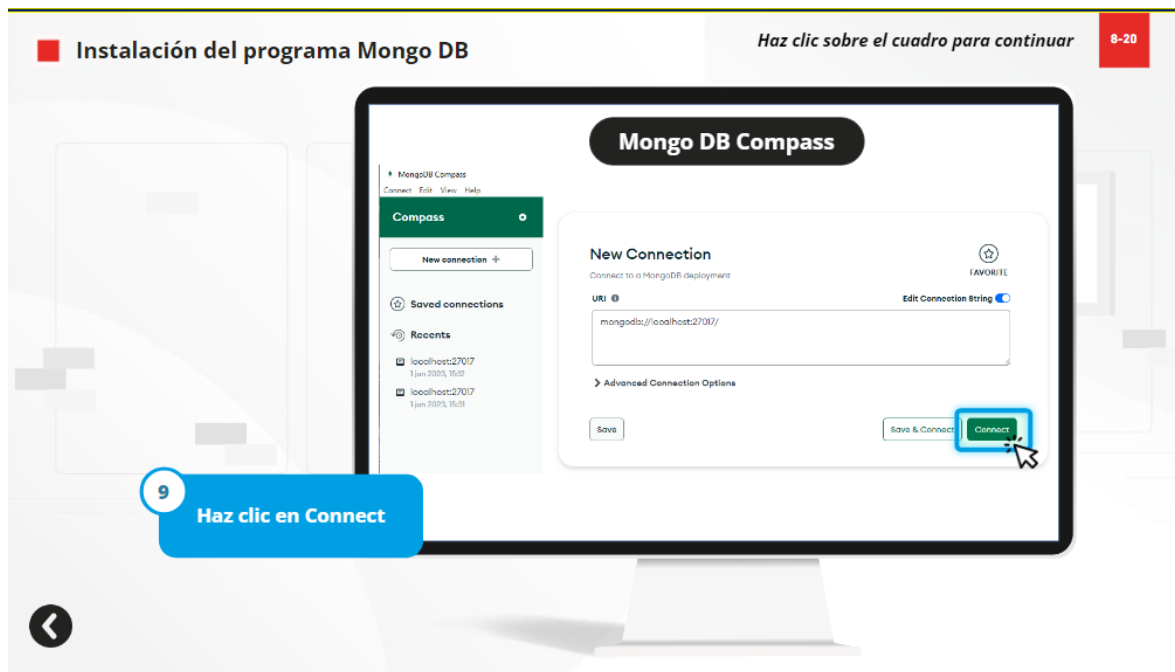
Etapas del proceso de instalación de MongoDB y principales comandos de gestión

Dirígete al sitio web de MongoDB y, luego, sigue los pasos que te muestra el siguiente video instructivo: <https://www.mongodb.com/try/download/community>



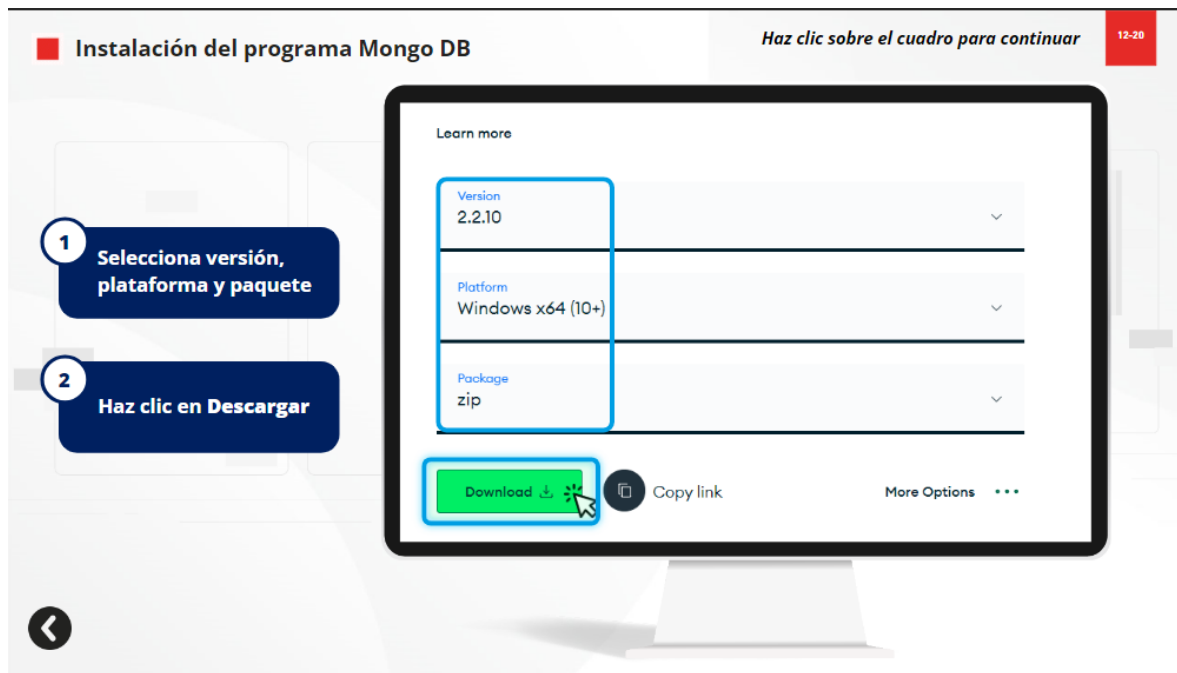
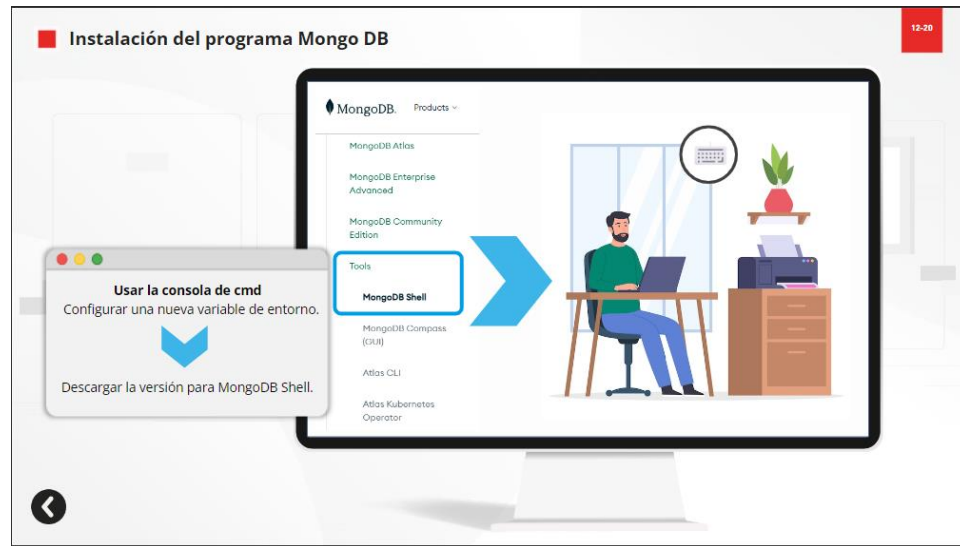






Recuerda que el modo consola de MongoDB Compass se encuentra en la parte inferior de la ventana y que esta se puede ampliar desde el borde superior de la misma. Por defecto, deja ubicado "test" en la base de datos.

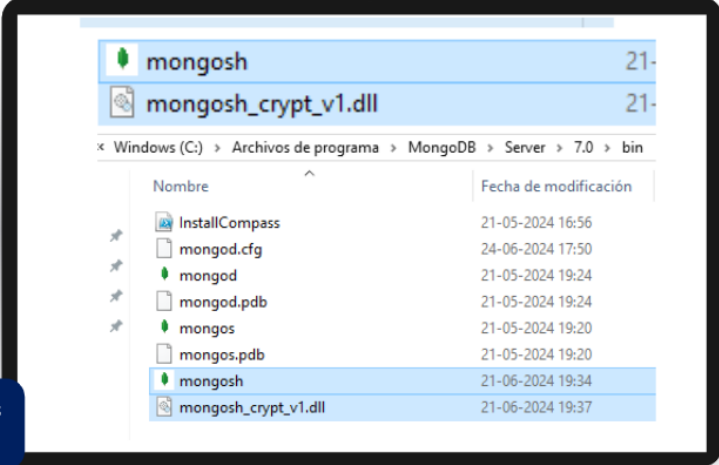
También se puede usar la consola de cmd. Para ejecutar el programa de cualquier ruta, se debe (2) configurar una nueva variable de entorno. Para ello, primero debes (3) descargar la versión para MongoDB Shell. Hazlo desde (3) el apartado Tools, ubicado en la columna izquierda de la página de descarga,



Instalación del programa Mongo DB 13-20

3 Descomprime la descarga

4 Copia los dos archivos en la carpeta bin



Nombre	Fecha de modificación
InstallCompass	21-05-2024 16:56
mongod.cfg	24-06-2024 17:50
mongod	21-05-2024 19:24
mongod.pdb	21-05-2024 19:24
mongos	21-05-2024 19:20
mongos.pdb	21-05-2024 19:20
mongosh	21-06-2024 19:34
mongosh_crypt_v1.dll	21-06-2024 19:37

Instalación del programa Mongo DB 14-20

5 Copia la ruta de MongoDB



C:\Program Files\MongoDB\Server\7.0\bin

15-20

Editar las variables de entorno del sistema

Panel de control

Configuración

Editar las variables de entorno de esta cuenta

Buscar en Internet

variables - Ver más resultados de la búsqueda

variables de entorno

variables del sistema

Carpetas

- Variables - en VisualScripting.Core
- Variables - en Framework
- Variables - en VisualScripting.Core

Documentos

- Variables.meta - en Framework

variables

16-20

Propiedades del sistema

Nombre de equipo Protección del sistema Hardware

Opciones avanzadas Acceso remoto

Para realizar la mayoría de estos cambios, inicie sesión como administrador.

Rendimiento
Efectos visuales, programación del procesador, uso de memoria y memoria virtual

[Configuración...](#)

Perfiles de usuario
Configuración del escritorio correspondiente al inicio de sesión

[Configuración...](#)

Inicio y recuperación
Inicio del sistema, errores del sistema e información de depuración

[Configuración...](#)

[Variables de entorno...](#)

[Aceptar](#) [Cancelar](#) [Aplicar](#)

Instalación del programa Mongo DB Haz clic sobre el cuadro para continuar 16-20

Variables de entorno

Variables de usuario para Usach

Variable	Valor
OneDrive	C:\Users\Usach\OneDrive
Path	C:\Users\Usach\AppData\Local\Programs\Python\Python39\Script...
QT_DEVICE_PIXEL_RATIO	auto
TEMP	C:\Users\Usach\AppData\Local\Temp
TMP	C:\Users\Usach\AppData\Local\Temp

Nuevo... **Editar...** Eliminar

8 Selecciona Path y, luego, haz clic en Editar

Instalación del programa Mongo DB 17-20

Editar variable de entorno

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps

Nuevo Modificar Examinar... Eliminar

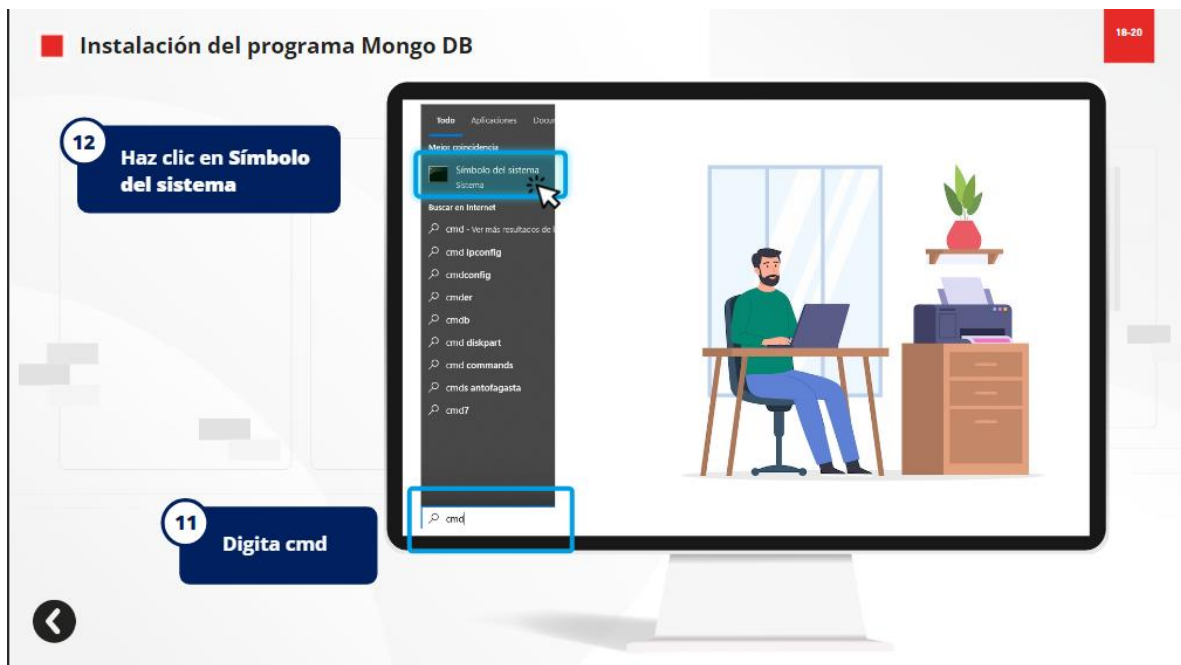
Editar variable de entorno

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files\MongoDB\Server\7.0\bin

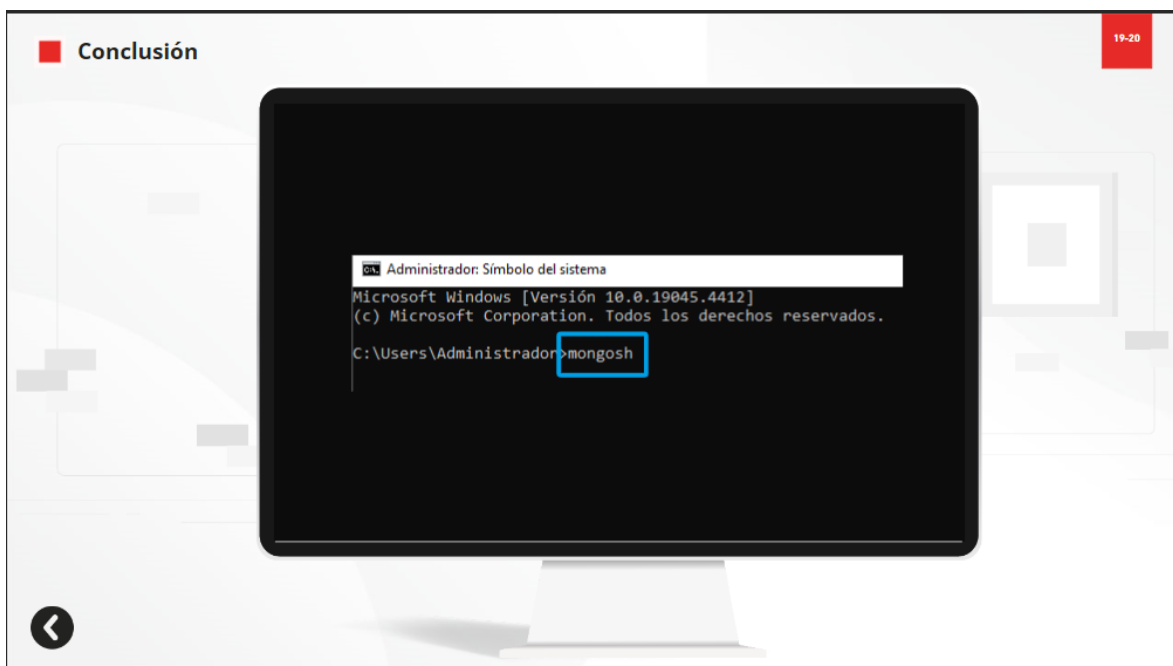
9 Haz clic en Nuevo

11. Pega la ruta de MongoDB

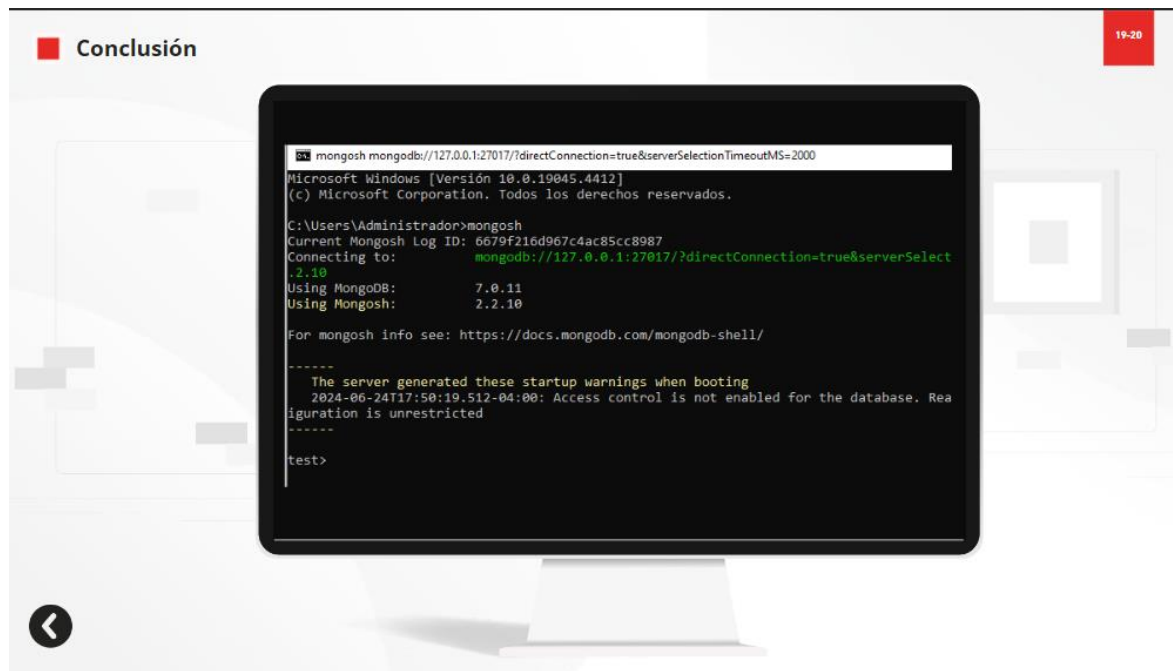
Luego de pegar la ruta de MongoDB, confirma los cambios haciendo clic en Aceptar en todas las ventanas activas.



Finalmente, digita *mongo* para entrar al entorno.



¡Listo! Ya puedes comenzar a usar comandos de MongoDB



Comandos básicos

Para aprovechar al máximo las capacidades de **MongoDB**, es esencial familiarizarse con sus comandos básicos. Estos comandos te permitirán realizar operaciones fundamentales como crear y eliminar bases de datos, insertar, consultar, actualizar y eliminar documentos, así como gestionar índices y colecciones.

Comandos con cmd

Visualización de bases de datos existentes

```
shell
```

[Copiar código](#)

```
show dbs
```

```
show dbs
```

Consultar la base de datos actual

```
shell Copiar código  
  
db
```

Db

Crear una base de datos

```
shell Copiar código  
  
use bd_datos
```

No se crean como tal hasta que no se produce una inserción en alguna de sus colecciones.

use bddato

Recuerda que este comando solo indica que se va a utilizar la base de datos.


Crear colecciones

```
shell Copiar código  
  
db.createCollection("vehiculos")
```

db.createCollection("vehiculos")

Visualizar colecciones de la base de datos actual

shell

 Copiar código


```
show collections
```

show collections

Comandos con MongoDB Compass

Eliminar colecciones

shell


 Copiar código

```
db.vehiculos.drop()
```

db.vehiculos.drop()

Ayuda en MongoDB

shell


 Copiar código

```
db.help()
```

db.help()

Eliminar base de datos activa

shell


 Copiar código

```
db.dropDatabase()
```

db.dropDatabase()

Limpiar pantalla

shell


 Copiar código

```
cls
```

cls

Otro comando de MongoDB Compass es Obtener estadísticas: **db.stats()**

shell

 Copiar código

```
>db.stats()

1
"db" : "test",
"collections" : 5,
"views" : 0,
"objects" : 53829,

"avgObjSize" : 43.555,
"dataSize" : 2344556121,
"storageSize" : 3124416336,

"numExtents" : 0,
"indexes" : 7,

"indexSize" : 8096876,

"ok" : 1
```

A medida que te familiarices con estos comandos básicos, te será más fácil avanzar hacia comandos y características más avanzadas, optimizando así tus capacidades y eficiencia en el manejo de datos.

Operaciones CRUD en documentos y subdocumentos

Operaciones CRUD en documentos

Existe un formato general para insertar documentos, pero ha sido reemplazados por los métodos que se definirán a continuación.

Insert(): para insertar los documentos de uno en uno

Ejemplos:

```
//Creación de base de datos
> use empresa
//Crear nueva colección
> db.createCollection("trabajador")
//Inserción de datos a documento de la colección con insert()
> db.trabajador.insert({rut:"13.444.555-6", nombre:"Juan Perez", ciudad:"Santiago",
genero:"M", edad:27, sueldo:530000})
```

Los dos métodos actuales para insertar documentos:

- `insertOne()`: para insertar los documentos de uno en uno.
- `insertMany()`: para insertar varios documentos en un mismo comando.

Inserción simple

Para insertar un documento de forma individual el método que se debe usar es `insertOne()`. El siguiente ejemplo crea en la colección `álbumes` perteneciente a la base de datos `música`,

un documento con el álbum "Revolver" de los Beatles:

```
> use musica
> db.createCollection("trabajador")
> db.albumes.insertOne({"nombre" : "Revolver", "autor" : "The Beatles", "año" : 1966 })
```

Observación: No se ha incluido el campo `_id`, pero MongoDB lo añadirá automáticamente. Se puede visualizar el documento con la instrucción `findOne()`:

```
> db.albumes.findOne()
{
  "_id" : ObjectId("5f856f714f02f31ac68acf3e"),
  "nombre" : "Revolver",
  "autor" : "The Beatles",
  "año" : 1966
}
```

Inserción múltiple

Para añadir varios documentos de una sola vez a una colección se usa `insertMany()`.

Para inserciones múltiples esta es la forma más eficiente, ya que se envía la información de una única vez. Este método recibe como argumento un array de documentos.

El siguiente ejemplo introduce otros tres álbumes de los Beatles con este método.

```
> db.albumes.insertMany([
  {
    "nombre" : "Abbey Road",
    "autor" : "The Beatles",
    "año" : 1969
  },
  {
    "nombre" : "Let It Be",
    "autor" : "The Beatles",
    "año" : 1970
  },
  {
    "nombre" : "The White Album",
    "autor" : "The Beatles",
    "año" : 1968
  }
])
```

```
{
  "nombre" : "Help!",
  "autor" : "The Beatles",
  "año" : 1965
}
])
```

Verificando la inserción

```
> db.albumes.find()
{ "_id" : ObjectId("5f85824b4f02f31ac68acf43"), "nombre" : "Revolver", "autor" : "The Beatles", "año" : 1966 }
{ "_id" : ObjectId("5f85824e4f02f31ac68acf44"), "nombre" : "Abbey Road", "autor" : "The Beatles", "año" : 1969 }
{ "_id" : ObjectId("5f85824e4f02f31ac68acf45"), "nombre" : "Let It Be", "autor" : "The Beatles", "año" : 1970 }
{ "_id" : ObjectId("5f85824e4f02f31ac68acf46"), "nombre" : "Help!", "autor" : "The Beatles", "año" : 1965 }
```

Observación: El inicio y fin de la inserción es con paréntesis cuadrados [] y la separación entre los documentos es una coma (,).

- **Método insertMany()**

El método insertMany() tiene un segundo parámetro opcional que consiste en un documento con opciones de cómo realizar la actualización. Este documento-parámetro puede tener dos claves, las dos opcionales:

- **writeConcern:** Se utiliza para indicar a MongoDB un nivel de confirmación respecto a si el dato ha sido guardado correctamente. Normalmente, solo lo usaremos en instalaciones con sharding o conjuntos de réplicas.
- **Ordered:** Este parámetro puede tomar los valores true (valor por defecto) o false. Le indica a MongoDB si es necesario insertar los documentos en el mismo orden en el que se envían o puede modificarse dicho orden. Si lo se

configura true, entonces en caso de que ocurra algún fallo en la actualización de un documento se abortará la carga. Si se configura false se omitirá el error y la carga continuará sin problemas.

Inserción por medio de import

Mongoimport es una herramienta nativa de MongoDB, pero no se incluye por defecto en la instalación estándar. Para utilizarla, es necesario instalar el paquete mongodb-org-tools, que contiene diversas herramientas, incluida mongoimport.

Pasos para instalar mongodb-org-tools:


Paso 1. Descargar la herramienta del sitio


<https://www.mongodb.com/try/download/database-tools>

MongoDB Command Line Database Tools Download

The MongoDB Database Tools are a collection of command-line utilities for working with a MongoDB deployment. These tools release independently from the MongoDB Server schedule enabling you to receive more frequent updates and leverage new features as soon as they are available. See the [MongoDB Database Tools](#) documentation for more information.

Version	100.9.4	▼
Platform	Windows x86_64	▼
Package	msi	▼

Download 

 Copy link

More Options ...

Paso 2: Instalar por defecto

Paso 3: Copiar la ruta donde se instaló la herramienta. Por ejemplo:

C:\Program Files\MongoDB\Tools\100\bin

Paso 4:

Ir a las variables de entorno Y Configurar una nueva variable de entorno. Esto produce que se puedan ejecutar comandos de cualquier directorio de CMD

Luego pegar la ruta de la herramienta y confirmar los cambios.

Paso 5: Entrar a Mongo, crear la base de datos db bdNewPeli y luego la colección películas.

Paso 6: Iniciar otro cmd

Una vez instalado este paquete se puede comprobar que funciona correctamente ejecutando: `mongoimport --help`

Formato general de importación

```
mongoimport --db baseDatos --collection coleccion --host localhost:puerto --type csv --file archivo_a_importar
```

Ejemplo json:

```
mongoimport --db bdNewPeli --collection peliculas --jsonArray --host localhost:27017 -file C:\infoPeliculas.json
```

Ejemplo csv:

```
mongoimport --db bdPagos --collection pagos --type csv --host localhost:27017 --file C:\pagos.csv --headerline
```

Consulta de documentos

Se usa el método `find()` para consultar documentos.

Ejemplo: `db.libros.find();`

//muestra toda la información de la colección libros

Formato general:

```
db.collection.find({condición},{proyección})
```

Devuelve los campos indicados de los documentos que satisfacen las condiciones

- **condición:** especifica condiciones o filtros
- **proyección:** especifica los campos a mostrar

Veamos algunas observaciones al respecto:

- Las secciones condición y proyección son opcionales.
- Condición vacía devuelve todos los documentos (filas)
- Proyección vacía devuelve todos los campos (columnas)
- En la proyección para mostrar se usa un número con el campo distinto de 0 y para no mostrar un campo se usa un 0
- Existe la expresión `findOne()` que mostrará el primer documento que coincida con la condición o condiciones.

Consultas sin condiciones

SQL	NoSQL
Para la tabla productos: select * from productos; select id, nombre from productos; select nombre from cliente;	Para la colección productos: db.productos.find() db.productos.find({},{nombre:1}) db.productos.find({},{nombre:1,_id:0})

Consultas con condiciones

SQL	NoSQL
Para la tabla productos: select nombre from productos where situacion= 'A'; select nombre from productos where situacion<>'A'; select * from productos where situacion="A" or precio=5000;	Para la colección productos: db.productos.find({situacion:"A"},{nombre:1,_id:0}) db.productos.find({situacion:{\$ne:"A"}},{nombre:1,_id:0}) db.productos.find({\$or:[situacion:"A",{precio:5000}]})

Operadores relacionales

•	<u>\$eq</u>	- <u>igual</u>
•	\$lt	- menor que
•	\$lte	- menor o igual que
•	\$gt	- mayor que
•	<u>\$gte</u>	- mayor o igual que
•	\$ne	- distinto
•	\$in	- dentro de
•	\$nin	- no dentro de

Veamos algunos ejemplos:

- Mostrar todos los libros que tienen un precio mayor a 40000:

```
db.libros.find({ precio: { $gt:40000 } });
```

- Mostrar todos los libros que en el campo cantidad tiene 50 o más:

```
db.libros.find( { cantidad: { $gte : 50 } });
```

- Mostrar todos los libros que en el campo cantidad hay un valor distinto a 50:

```
db.libros.find( { cantidad: { $ne : 50 } });
```

- Mostrar todos los libros cuyo precio estén comprendidos entre 20000 y 45000:

```
db.libros.find( { precio: { $gte : 20000 , $lte : 45000 } } );
```

Operadores lógicos

Una condición «and» realiza una operación lógica «and» en una matriz de dos o más expresiones. Selecciona los documentos en los que se cumplen todas las condiciones de las expresiones.

```
{ $and: [ { <expresion1> }, { <expresion2> }, ... , { <expresionN> } ] }
```

Una condición «or» realiza una operación lógica «or» en una matriz de dos o más expresiones. Selecciona los documentos en los que al menos una de las expresiones es verdadera.

```
{ $or: [ { <expresion1> }, { <expresion2> }, ... , { <expresionN> } ] }
```

Operador realiza una operación lógica «nor» en una matriz utilizando una o varias expresiones. A continuación, selecciona los documentos que no cumplen las expresiones de la consulta. En términos más sencillos, hace lo contrario de la condición \$or

```
{ $nor: [ { <expresion1> }, { <expresion2> }, ... { <expresionN> } ] }
```

Operador realiza una operación lógica de «not» en una matriz para la expresión especificada. A continuación, selecciona los documentos que no coinciden con las expresiones de la consulta. Esto incluye los documentos que no contienen el campo.

```
{ campo: { $not: { <operador-expresion> } } }
```

Ordenamiento

```
consulta.sort(criterios)
```

criterios: documento con parejas {field: value} con value es 1 (ascendente) o -1 (descendente)

Veamos un ejemplo:

SQL	NoSQL
Para la tabla productos: select * from productos where situacion="A" order by nombre desc;	Para la colección productos: db.productos.find({situacion:"A"}).sort({nombre:- 1})

Count: Devuelve la cantidad de documentos que satisfacen las condiciones.

Ejemplo:

- **SQL:** Para la tabla productos
select count(*) from productos where situacion="A";
- **NoSQL:** Para la colección productos
db.productos.count({situacion:"A"})
db.productos.find({situacion:"A"}).count()

- ***distinct***

Devuelve los valores diferentes de cierto campo de los documentos que satisfacen las condiciones.

Ejemplo:

- **SQL:** Para la tabla productos
`select distinct(precio) from productos;`
- **NoSQL:** Para la colección productos
`db.productos.distinct("precio")`

- ***Búsquedas de textos***

Se utilizan para buscar coincidencias en donde la celda tenga una porción de una cadena.

- Su formato es:
`db.tu_coleccion.find({"campo": /. *busqueda.*/})`
- Para ignorar si son mayúsculas o minúsculas:
`db.tu_coleccion.find({"campo": /. *busqueda.*/i})`

Ejemplo:

- **SQL:** Para la tabla productos
`select nombre from productos where nombre like 'A%'`
- **NoSQL:** Para la colección productos
`db.productos.find({nombre: /. *A./ });`

Eliminación de documentos

Expresión	Función	Ejemplo
deleteOne	Borra el primer documento que cumple la condición(es) que se configuran.	db.collection.deleteOne({condición})
deleteMany	Borra todos los documentos que cumplen la condición(es) que se configuran.	db.collection.deleteMany({condición})
remove	Es otra alternativa para eliminar uno o más documentos. Acepta dos parámetros: la condición y solo un valor booleano que, si se establece en verdadero, elimina solo un documento.	db.collection.remove({condición, valor_booleano})

El resultado de la ejecución de cualquiera de estas formas, dará como resultado en MongoDB la cantidad de filas borradas.

Para la creación de condiciones de borrado se pueden utilizar los mismos operadores de consultas.

Veamos algunos ejemplos:

Eliminación de todos los documentos

Se pueden eliminar todos los documentos de una colección usando `deleteMany`, `remove` o `drop` sin condiciones.

Ejemplos:

```
db.libros.deleteMany({})
```

```
db.libros.remove({})
```

```
db.libros.drop()
```

Eliminación del primer documento con condición(es)

Ejemplos:

```
db.libros.deleteOne({copias:5});
```

```
db.libros.remove({copias:5}, true);
```

Eliminación de todos los documentos que cumplen condición(es)

Ejemplos:

```
db.libros.deleteMany({copias:5});
```

```
db.libros.remove({copias:5});
```

Mejores prácticas para eliminar documentos

Comprender lo que se está eliminando: Antes de eliminar cualquier documento, es importante comprender qué se está eliminando. Esto puede parecer obvio, pero muchos errores de eliminación ocurren porque el desarrollador no entendió completamente qué estaba eliminando. Por lo tanto, antes de eliminar cualquier documento, se debe comprender qué contiene y cómo se relaciona con otros documentos en la base de datos.

Utilizar el método deleteOne(): El método deleteOne() es el método más básico para eliminar documentos en MongoDB. Elimina el primer documento que coincide con los criterios de consulta dados. Aunque es sencillo de utilizar, se debe tener cuidado al utilizarlo ya que sólo elimina un documento a la vez. Si hay varios documentos que coinciden con los criterios de consulta, solo se eliminará el primero.

Utilizar el método deleteMany(): Si se necesita eliminar varios documentos que coinciden con un criterio de consulta, se puede utilizar el método deleteMany(). Este método elimina todos los documentos que coinciden con los criterios de consulta dados. Sin embargo, se debe tener cuidado al utilizarlo, ya que puede provocar la eliminación de demasiados documentos a la vez.

Utilizar el método `findOneAndDelete()`: El método `findOneAndDelete()` es similar al método `deleteOne()`, pero también devuelve el documento eliminado. Esto puede resultar útil si necesita una copia del documento que está eliminando. Sin embargo, este método es más lento que el método `deleteOne()` porque necesita devolver el documento antes de eliminarlo.

Utilizar el método `drop()`: Si se necesita eliminar una colección completa, se puede utilizar el método `drop()`. Este método elimina toda la colección, incluidos todos sus documentos e índices. Sin embargo, se debe tener cuidado al usarlo, ya que elimina permanentemente toda la colección.

Utilizar el método `remove()`: El método `remove()` es una forma antigua de eliminar documentos en MongoDB. Elimina todos los documentos que coinciden con los criterios de consulta dados. Sin embargo, este método ha quedado obsoleto en la versión 3.2 de MongoDB y no debe usarse en proyectos nuevos.

Hacer siempre una copia de seguridad antes de eliminar: Antes de eliminar cualquier documento, es una buena práctica realizar una copia de seguridad de la base de datos. Esto permite restaurar la base de datos si algo sale mal durante la eliminación. MongoDB ofrece varias herramientas para realizar copias de seguridad de bases de datos, incluidos `mongodump` y `mongoexport`.

Probar la consulta de eliminación antes de ejecutarla: Antes de ejecutar una consulta de eliminación, es una buena práctica probarla primero. Puede hacer esto ejecutando la consulta como una consulta de selección y verificando los resultados. Esto permite ver qué documentos se eliminarán antes de eliminarlos.

Actualización de documentos

Para actualizar datos, se utiliza el comando `update`. Este comando, en su forma básica, recibe dos parámetros: uno con la consulta para filtrar los documentos a modificar y otro con los elementos que se modificarán.

Existen diferentes usos de `update`. Veamos los formatos.

Para actualizar el primer documento que coincida con la consulta

```
db.colección.updateOne({condición}, { $set:{ campo y valor a actualizar}})
```

```
db.colección.update({condición}, {$set:{ campo y valor a actualizar}})
```

Para actualizar todos los documentos que coincidan con la consulta

```
db.colección.updateMany({condición}, {$set:{campo y valor a actualizar}})
```

```
db.colección.update({condición}, {$set:{campo y valor a actualizar}}, {multi:true})
```

Para insertar el documento si este no existe

```
db.colección.update({condición}, {campo y valor a actualizar}, {upsert:true})
```

El resultado de la ejecución de cualquiera de estas formas, dará como resultado en MongoDB la cantidad de filas actualizadas. Para la creación de condiciones de actualización se pueden utilizar los mismos operadores de consultas.

Ejemplos

```
db.personas.updateOne({sexo:"F"}, { $set:{ Edad:26}})
```

```
db.personas.update({sexo:"F"}, { $set:{ Edad:26}})
```

```
db.personas.updateMany({Sexo:"M"}, {$set:{Sueldo:570000}})
```

```
db.personas.update({Sexo:"M"}, {$set:{Sueldo:600000}}, {multi:true})
```

```
db.personas.update({Edad:25}, {Sueldo:580000}, {upsert:true})
```

Operaciones CRUD en subdocumentos

Documentos embebidos

Hasta ahora se ha trabajado definiendo documentos con una serie de campos que almacenan tipos de datos simples como enteros, reales, cadenas de caracteres y fechas.

Si se tiene que almacenar datos de una empresa, pero la dirección se debe discriminar por calle, número, comuna y ciudad, se tendría que crear un documento embebido o subdocumento.

Los documentos embebidos o subdocumentos permiten crear relaciones uno a muchos (1:N) presentes en las bases de datos relacionales, pero en un mismo documento. Veamos ejemplos.

- ***Insertión***

```
use base_semana4
db.empresa.drop()

db.clientes.insertOne(
{
  _id: 1,
  nombre: 'IPFactory',
  mail: 'informacion@ipfactory.com',
  direccion: {
```



```
    calle: 'Colon',
    numero: 620,
    comuna: 'Las Condes',
    ciudad: 'Santiago'
  }
}
)

db.clientes.insertOne(
{
  _id: 2,
  nombre: 'SubTerra',
  mail: 'contacto@subterra.com',
  direccion: {
    calle: 'miraflores',
    numero: 1200,
    comuna: 'Peñalolen',
    ciudad: 'Santiago'
  }
}
)

db.clientes.insertOne(
{
  _id: 3,
  nombre: 'TDrone',
  mail: 'servicio@pdrone.com',
  direccion: {
```

```
    calle: 'Acacias',
    numero: 894,
    comuna: 'Maipú',
    ciudad: 'Santiago'
  }
}
)

db.clientes.insertOne(
{
  _id: 4,
  nombre: 'SFacet',
  mail: 'personas@isfacet.com',
  direccion: {
    calle: 'Pasten',
    numero: 1350,
    comuna: 'Las Condes',
    ciudad: 'Santiago'
  }
}
)
```

Consultas

Para mostrar todas las empresas con dirección 'Colon' se debe realizar la siguiente consulta:

```
db.clientes.find({'direccion.calle':'Colon'})
```

Es obligatorio usar las comillas cuando se hace referencia a un subcampo de un documento embebido, ejemplo: 'direccion.calle'.

Para mostrar todas las empresas con donde su número está comprendido entre 1 y 1000 la siguiente consulta puede ser:

```
db.clientes.find({'direccion.numero':{$gte:0},'direccion.numero':{$lte:1000}})
```

Actualización

Actualizar la dirección, número y ciudad de TDrone a calle los aramos, número 455, comuna Machalí y ciudad Rancagua.

```
db.clientes.updateOne({nombre: 'TDrone'},{$set:{'direccion.calle': 'Los Aromos',  
'direccion.numero':455,'direccion.comuna':'Machalí','direccion.ciudad': 'Rancagua'}})
```

Eliminación

Borrar los clientes de la ciudad de Santiago.

```
db.clientes.deleteMany({'direccion.ciudad': 'Santiago'})
```

Arreglos o listas

Creación de arreglos por medio de variables

Formato:

```
var nombre_arreglo=[valor1, valor2,....,valorN]
```

Ejemplo:

```
var valores1 = [1,3,5]
```

```
var valores2 = [2,4,6]
```

Asignación de arreglos por medio de variables:

```
var usuario1 = {nombre: "Juan Perez", escala: valores2}
```

```
var usuario2 = {nombre: "Ana Moya", escala: valores1}
```

Ingresando documentos de tipo arreglo a la colección

```
db.ejemplo_arreglo.insert( usuario1)
```

```
db.ejemplo_arreglo.insert( usuario2)
```

Para verificar:

```
db.ejemplo_arreglo.find()
```

Agregando valores al arreglo con \$addToSet

Este método busca dentro de un arreglo el valor que se quiere ingresar, si está no lo inserta y si no se encuentra lo inserta en la última posición.

Ejemplo:

```
db.ejemplo_arreglo.update({}, {$addToSet: { escala : 4 } } )
```

Para verificar:

```
db.ejemplo_arreglo.find()
```

```
db.ejemplo_arreglo.updateMany({}, {$addToSet: { escala : 4 } } )
```

Para verificar:

```
db.ejemplo_arreglo.find()
```

Nota: Si se usa `update` busca solo la primera coincidencia, como en el primer documento hay un 4 no lo vuelve a insertar, por otro lado, al usar `updateMany` busca en todos los documentos e inserta el valor en el segundo documento al final de este.

Agregando valores al arreglo con \$push

Este método inserta en la última posición, si está o no está el valor a insertar

Ejemplo:

```
db.ejemplo_arreglo.update( {}, {$push : { escala : 4} })
```

```
db.ejemplo_arreglo.find();
```

```
db.ejemplo_arreglo.updateMany( {}, {$push : { escala : 4} })
```

```
db.ejemplo_arreglo.find();
```

Nota: Si se usa `update` busca solo la primera coincidencia, inserta el valor 4 al final del primer documento; por otro lado, al usar `updateMany` busca en todos los documentos e inserta el valor al final de cada documento.

Agregando más de un valor con `$each`

Este método permite insertar más de un valor al arreglo; solo inserta los valores que no se encuentran en el arreglo al final de este.

Ejemplo:

```
db.ejemplo_arreglo.updateMany ( {}, {$addToSet: { escala : {$each : [5,6]} } } )
```

```
db.ejemplo_arreglo.find();
```

Para insertar valores a un arreglo en una posición específica se puede usar `$position`

Ejemplo:

```
db.ejemplo_arreglo.updateMany( {}, {$push: { escala: {$each : [100, 101],
```

```
$position: 4 } } } )
```

```
db.ejemplo_arreglo.find();
```

Los índices en arreglos de MongoDB comienzan en 0

Para ordenar los valores del arreglo se usa `$sort`

Ejemplo:

```
db.ejemplo_arreglo.update( {}, { $push : { escala: { $each : [95, 97], $sort:1 } } } )
```

```
db.ejemplo_arreglo.find();
```

Eliminación de elementos de un arreglo

Para eliminar un elemento de un arreglo, se usa \$pull

Ejemplo:

```
db.ejemplo_arreglo.updateMany( {}, { $pull : { escala : 100 } } )
```

```
db.ejemplo_arreglo.find();
```

Se puede eliminar con intervalos de valores:

Ejemplo:

```
db.ejemplo_arreglo.updateMany( {}, { $pull : { escala : { $gte : 50 } } } )
```

```
db.ejemplo_arreglo.find();
```

Para eliminar más de un elemento de un arreglo, se usa \$pullAll

Ejemplo:

```
db.ejemplo_arreglo.updateMany( {}, { $pullAll : { escala : [4, 5] } } )
```

```
db.ejemplo_arreglo.find();
```

Consulta de arreglos

Para mostrar solo los elementos del arreglo

Ejemplo:

```
db.ejemplo_arreglo.find( {}, { _id:0 , escala:1 } )
```

Para mostrar una cantidad determinada de elementos del arreglo se usa \$slice

Ejemplo:

```
db.ejemplo_arreglo.find( {}, { _id:0, nombre:1, escala : { $slice : 2 } } )
```

Si el valor de la visualización es negativo mostrará los valores desde el final del arreglo.

Ejemplo:

```
db.ejemplo_arreglo.find( {}, { _id:0, nombre:1, escala : { $slice : -2 } } )
```

También se puede definir un intervalo de valores a mostrar

Ejemplo:

```
db.ejemplo_arreglo.find( {}, { _id: 0, nombre:1, escala: { $slice : [1, 3] } } )
```

En este ejemplo muestra los valores de la posición 1 a la 3

Para realizar comparaciones con elementos del arreglo

Ejemplo:

```
db.ejemplo_arreglo.find({ escala: 6 },{_id:0, escala:1})
```

Se puede usar \$all para buscar más de un valor en documentos

Ejemplo:

```
db.ejemplo_arreglo.find({ escala: {$all:[6,2]} },{_id:0, escala:1})
```

Usando el operador \$all se buscan varios elementos dentro de un arreglo, especificando como entrada un arreglo de elementos a buscar. En el ejemplo se está buscando todas las escalas que contengan 6 y 2. Sólo se devolverán los documentos que contengan ambos valores. Esto corresponde a un Y lógico.

También para realizar comparaciones con los elementos del arreglo se puede usar \$in o \$nin

Ejemplo:

```
db.ejemplo_arreglo.find( { escala: { $in : [ 6 ] } }, {_id:0, escala:1} )
```

Si se agregan valores después del 6, estos serán evaluados como un o lógico en la consulta

Ejemplo:

```
db.ejemplo_arreglo.find( { escala: { $nin : [ 2,5 ] } }, {_id:0, escala:1} )
```

Otro operador es \$size que busca arreglos con un determinado tamaño

Ejemplo:

```
db.ejemplo_arreglo.find( { escala: { $size:3 } },{_id:0, escala:1})
```

Dot Notation

Se utiliza en para realizar consultas en arreglos y en subdocumentos. Se basa en añadir un punto después del identificador del arreglo o subdocumento para realizar consultas sobre un índice en concreto del arreglo o sobre un campo concreto del subdocumento.

Ejemplo:

```
db.ejemplo_arreglo.find( { "escala.0": 2 },{_id:0, escala:1})
```

En el ejemplo buscamos todos los documentos que cumplan la condición de que el valor 0 del arreglo sea 2, el uso de comillas es obligatorio.

En síntesis, en este recurso de la asignatura aprendiste que MongoDB es una base de datos gratuita y multiplataforma que puede ser usada mediante comandos en distintos formatos gráficos, ya sea por medio de cmd o por medio de MongoDB Compass.

Por otro lado, es importante conocer la definición de arreglos y sus métodos, pues estos permiten generar colecciones de información mucho más cercanos a la realidad con una rapidez de procesamiento de consultas muy elevado. MongoDB tiene una serie de comandos para insertar, consultar, eliminar y actualizar documentos, los diferentes formatos están disponibles en las versiones de MongoDB pero, con el paso del tiempo, algunos de ellos han quedado obsoletos; se recomienda según la versión a utilizar verificar la documentación.

Bibliografía

- Iglesias, A; Cuadra, D; Castro, E. (2014). Desarrollo de bases de datos. RA-MA Editorial.
- Sarasa, A. (2026). *Introducción a las bases de datos NoSQL usando MongoDB*. Editorial UOC. <https://elibro.net/es/lc/inacap/titulos/58524>