

Tutorial - Bases de datos no relacionales con MongoDB

Contenido

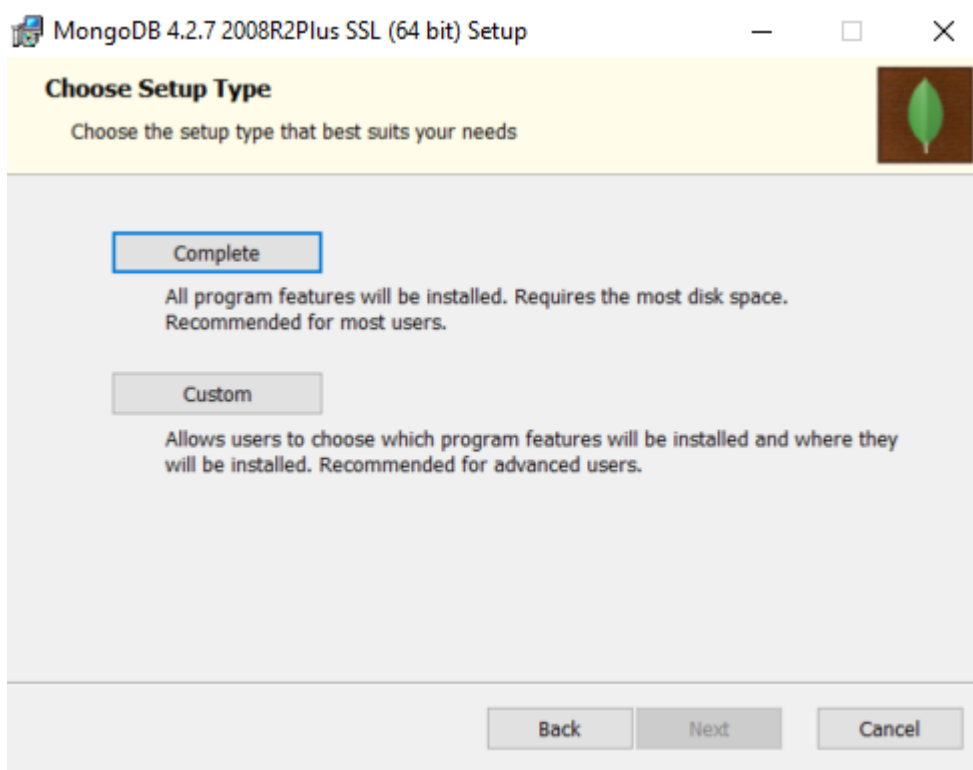
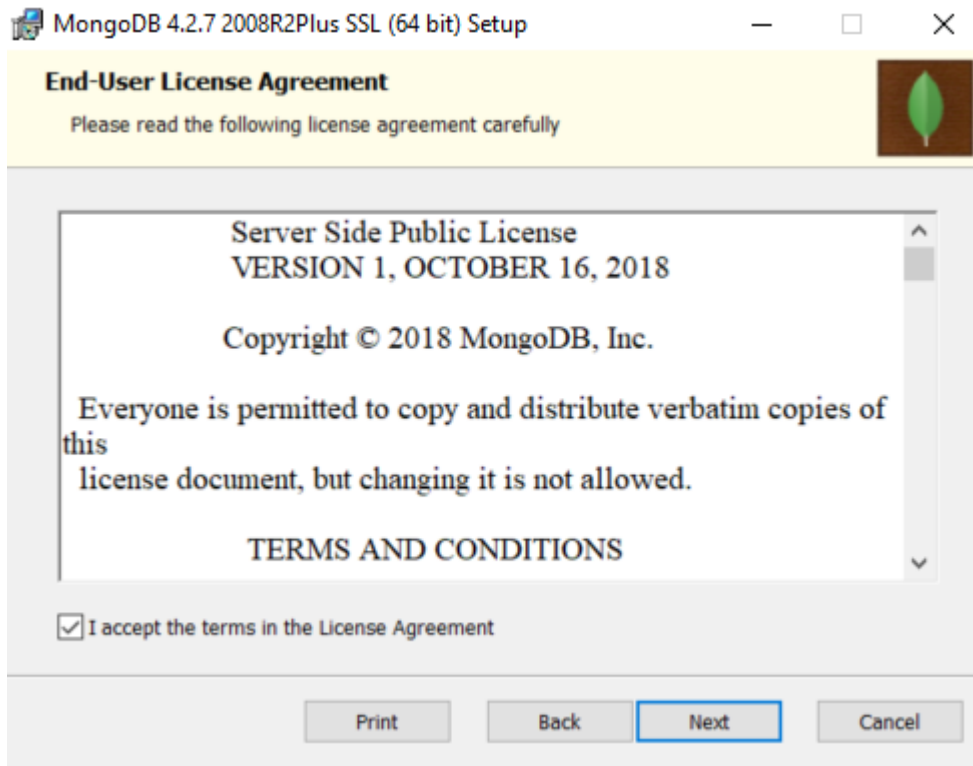
- 1. Instalación
- 2. Configuración
- 3. Ejemplo (Librería) - Base de datos No SQL en MongoDB
- 4. Referencias

1. Instalación (Windows)

1.1. Descargar el archivo de instalación en el sitio oficial de MongoDB

The screenshot shows the MongoDB download center page. The 'Server' tab is selected. Under 'MongoDB Community Server', the 'Version' dropdown is set to '4.2.7 (current release)', the 'OS' dropdown is set to 'Windows x64', and the 'Package' dropdown is set to 'MSI'. The 'Download' button is highlighted. Below the dropdowns, the download URL is displayed: https://fastdl.mongodb.org/win32/mongodb-win32-x86_64-2012plus-4.2.7-signed.msi. To the right, there is a list of links: Release notes, Changelog, All version binaries, Installation instructions, Download source (tgz), and Download source (zip). At the bottom left, a download bar shows the file 'mongodb-win32-x86_64-2012plus-4.2.7-signed.msi' with a size of 176/254 MB and a status of 'Fasten 29 s'.

1.2. Una vez descargado, ejecutar el archivo de instalación y seguir los pasos.



Service Configuration
Specify optional settings to configure MongoDB as a service.

☐ Install MongoD as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

Service Name:

Data Directory:

Log Directory:

< Back **Next >** Cancel

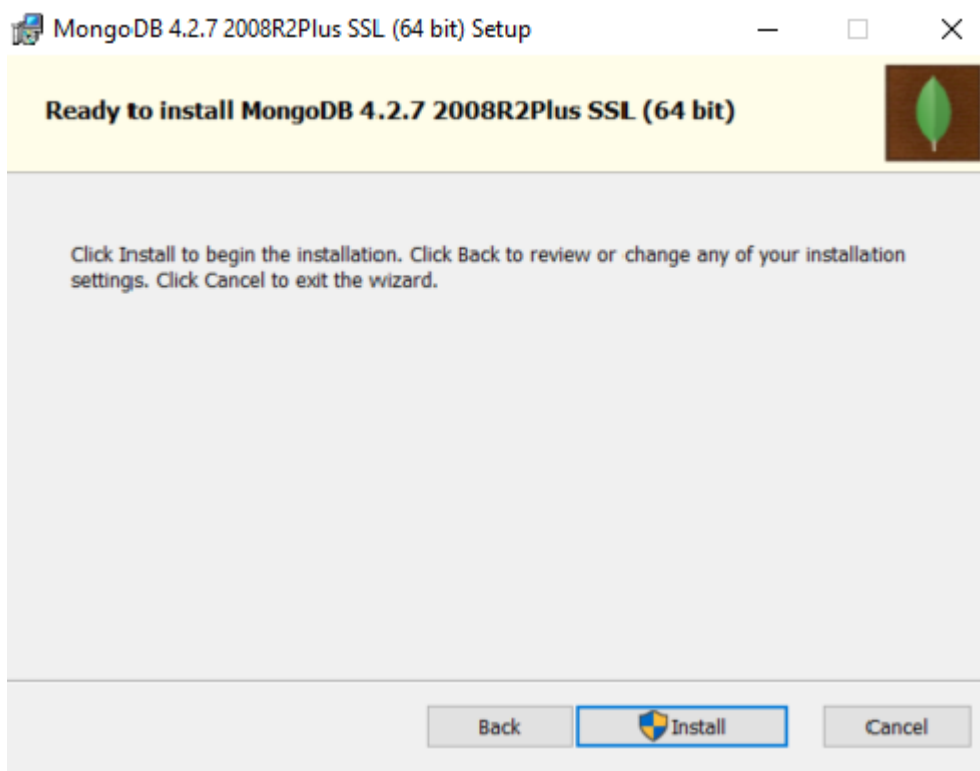
Recomendación: No seleccionar la opción **Instalar MongoDB como un servicio**, ya que el servicio **mongod** se activará automáticamente cada vez que inicie su sistema operativo y esto podría disminuir el rendimiento de su equipo. Para el ejercicio académico, es buena práctica activar manualmente el servicio únicamente cuando se requiera.

Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.

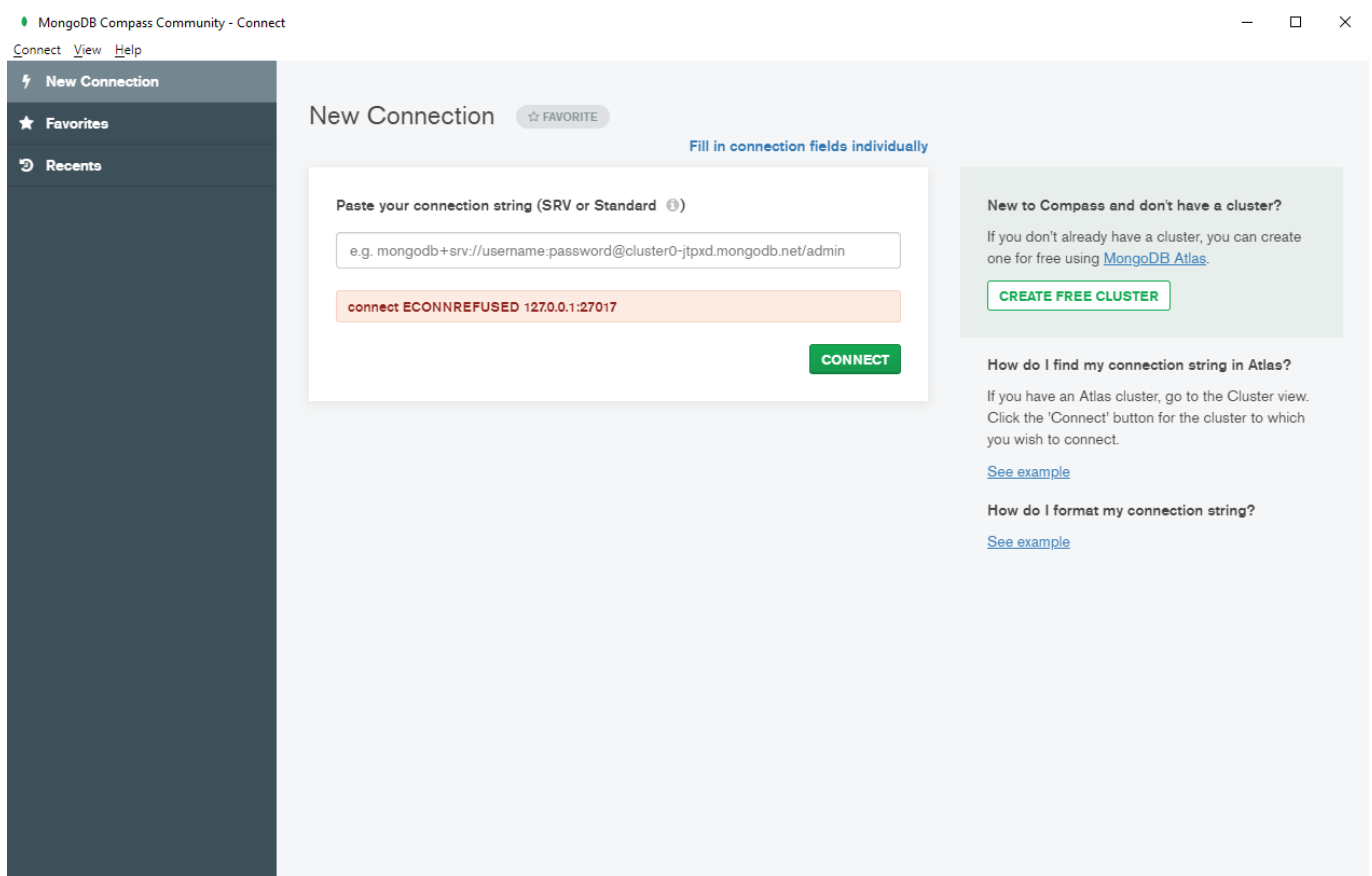
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: <https://www.mongodb.com/products/compass>

☒ Install MongoDB Compass

Back **Next** Cancel



Una vez finalice la instalación, se ejecutará automáticamente **MongoDB Compass Community**, una herramienta GUI que nos permitira visualizar y manipular de forma gráfica nuestras bases de datos. La utilizaremos mas adelante.

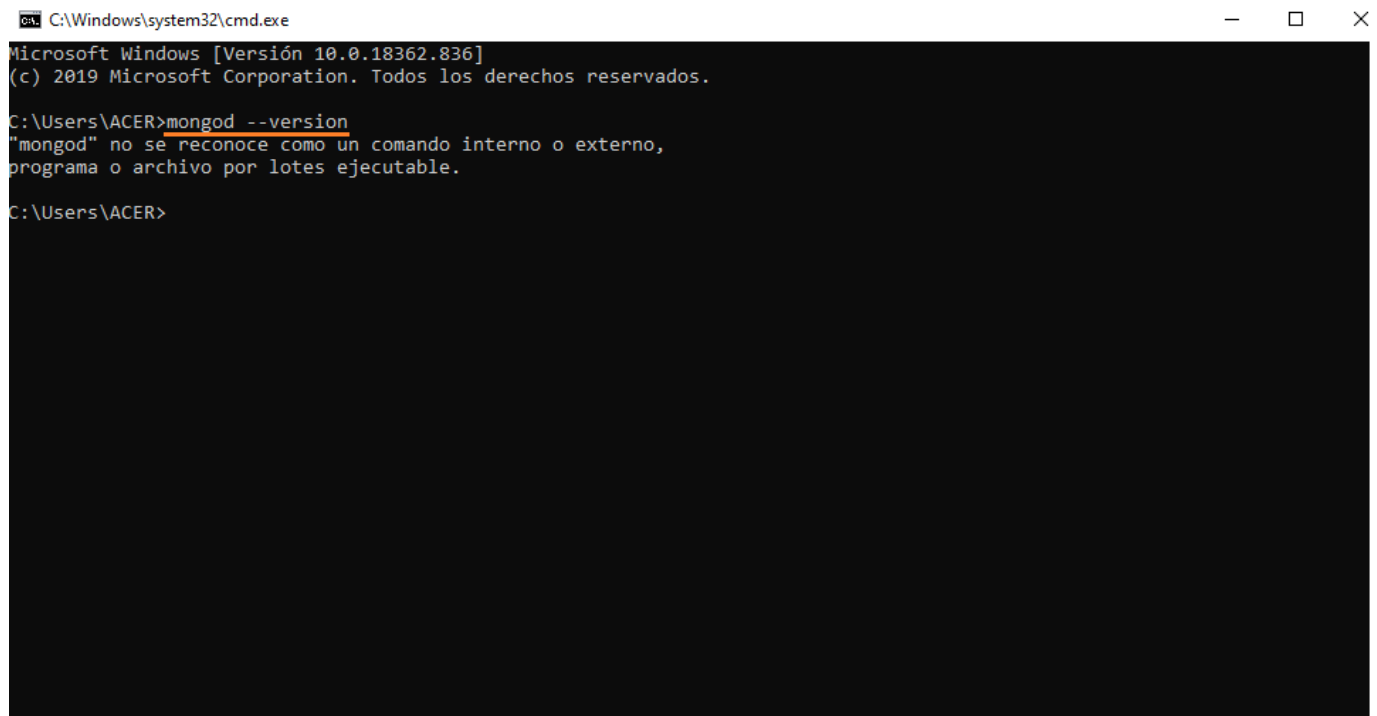


2. Configuración (Windows)

Cada vez que utilicemos el servicio, inicialmente debemos activarlo y posteriormente conectarnos a este. La activación del servicio se puede realizar a través de la consola de comandos (*CMD*) del sistema operativo y la conexión se puede realizar mediante la consola o a través de la herramienta GUI (*MongoDB Compass Community*).

Para activar el servicio de MongoDB a través de la consola de comandos, necesitamos configurar la ruta donde se encuentra el archivo de activación para que el sistema operativo lo reconozca.

2.1. Abrir la consola de comandos *CMD* y ejecutar ***mongod --version***. Si no se encuentra configurada la ruta de ubicación del archivo nos debe aparecer lo siguiente.



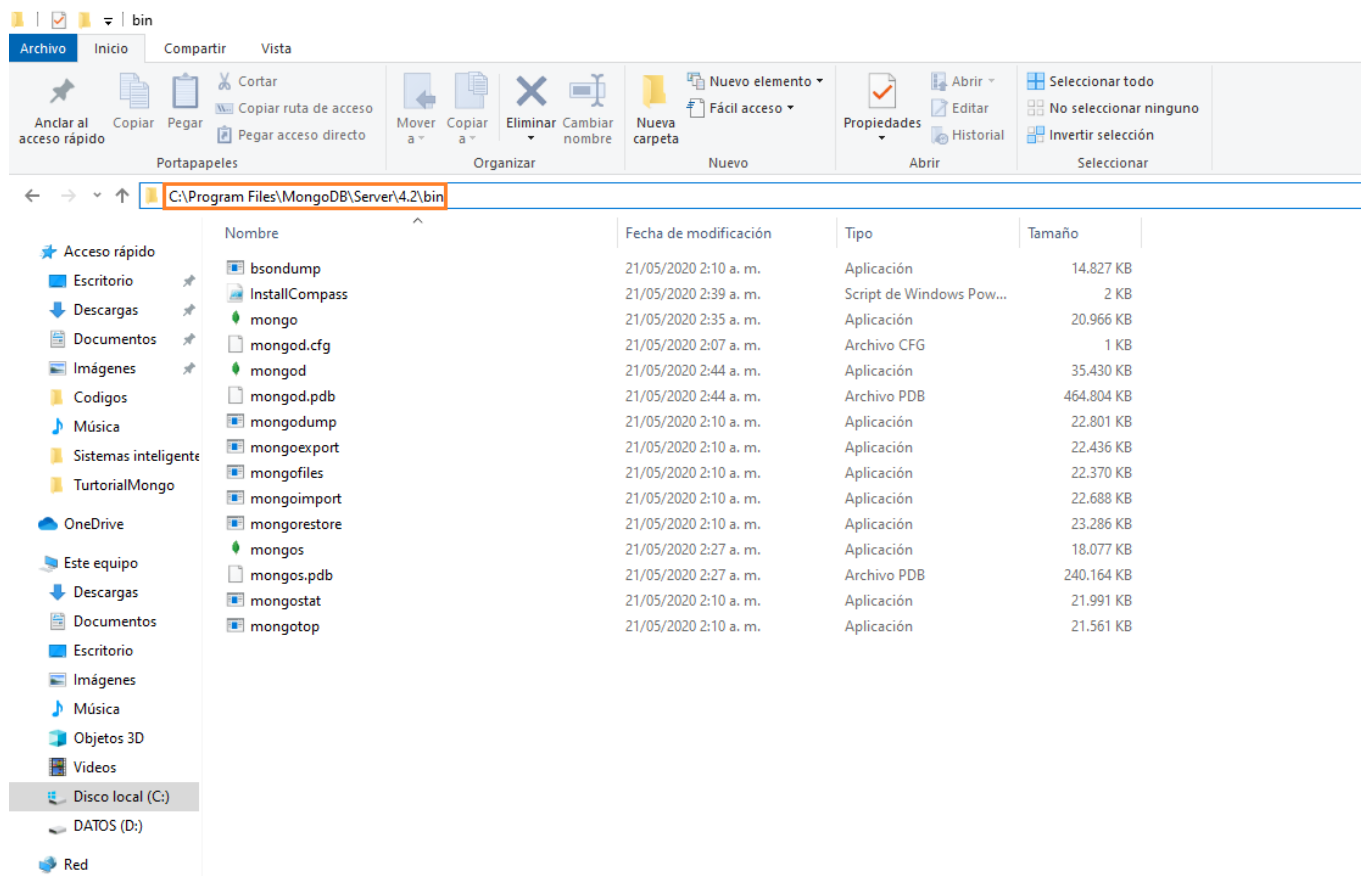
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\ACER>mongod --version
"mongod" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

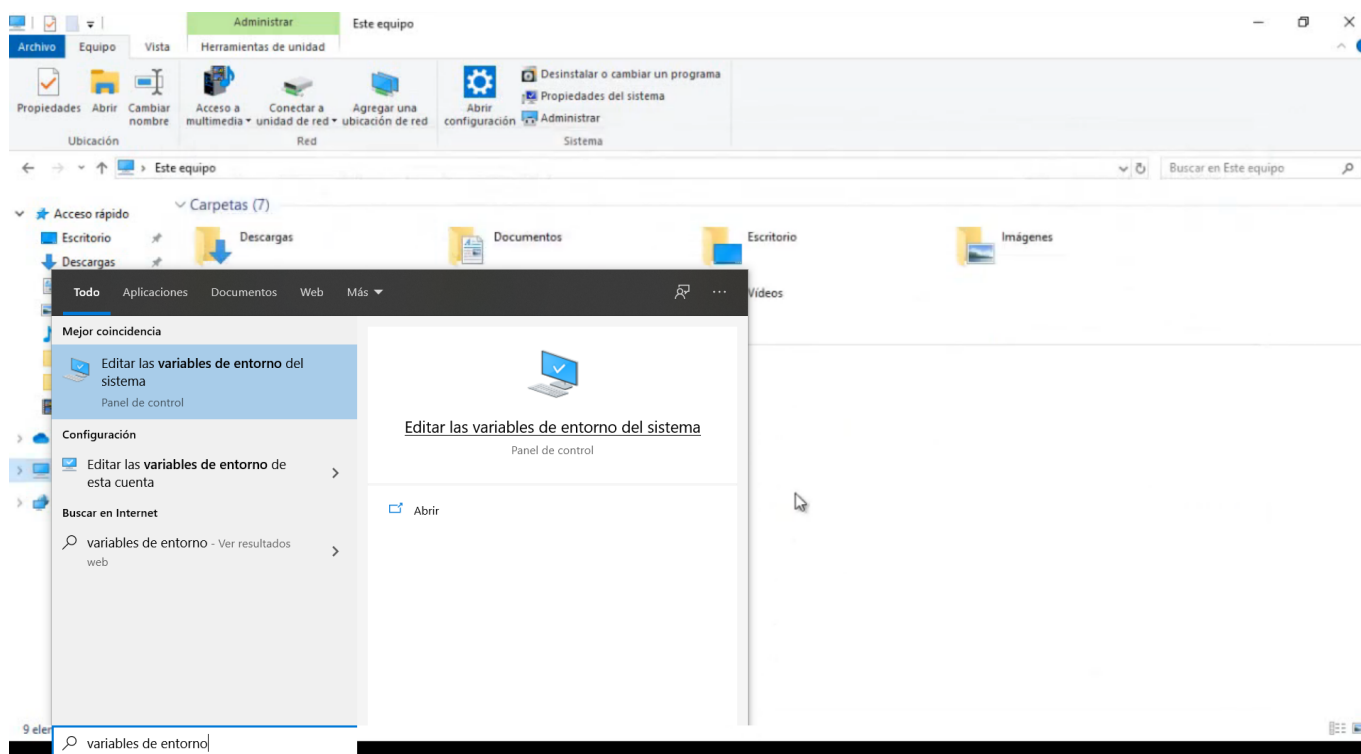
C:\Users\ACER>
```

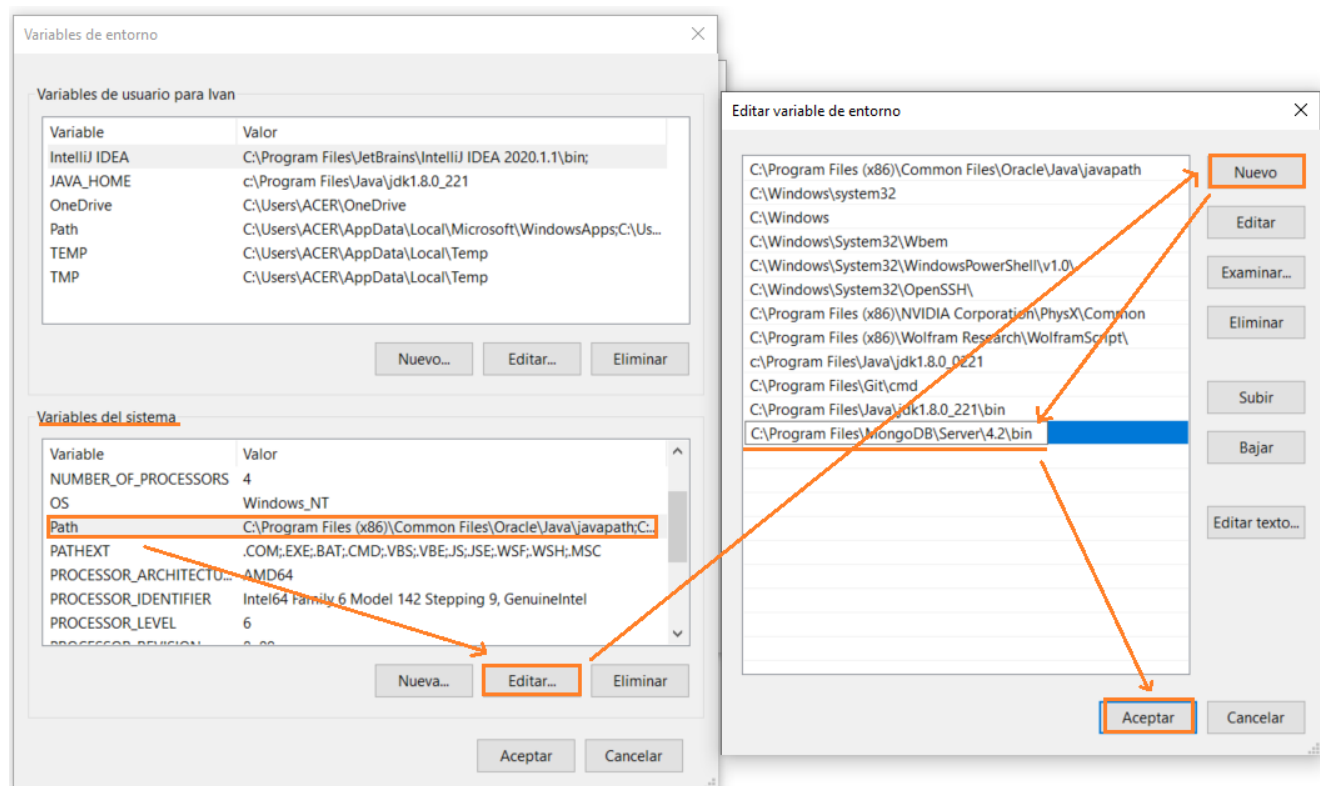
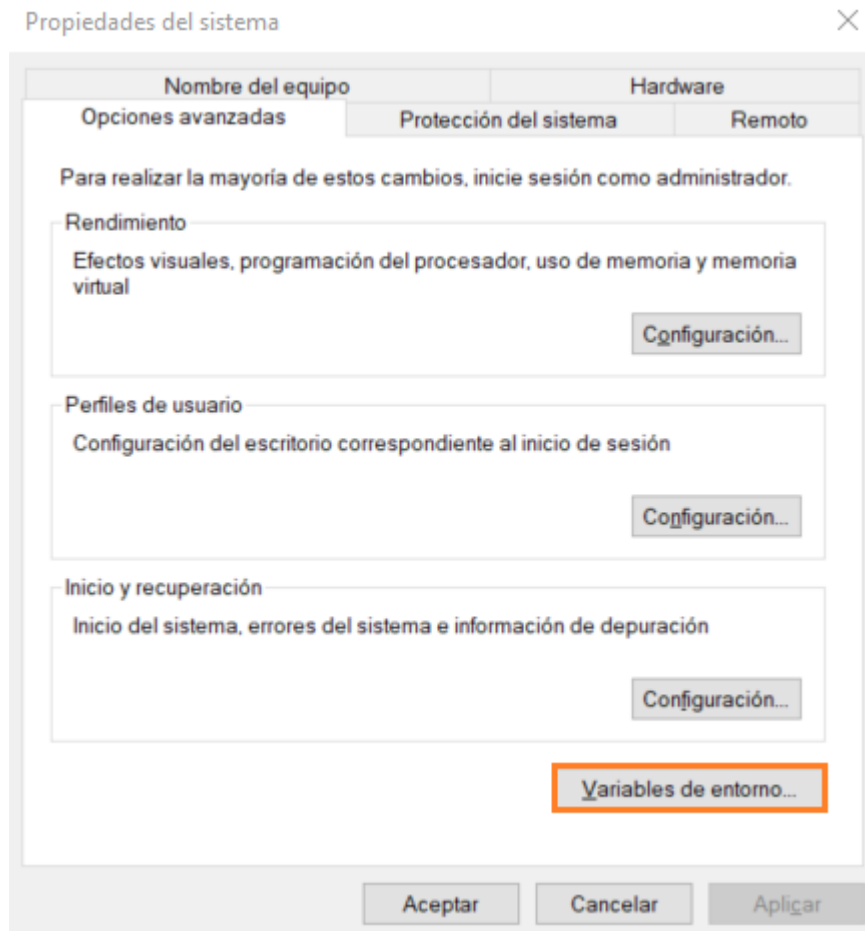
Para solucionarlo vamos a adicionar la dirección donde se encuentra el archivo de activación al path de rutas de windows.

2.2. Buscar la carpeta ***bin*** que contiene el archivo de activación ***mongod*** y copiar esa ruta. Normalmente la carpeta se aloja en la dirección ***C:\Program Files\MongoDB\Server\4.2\bin***



2.3. Adicionar esta ruta a las variables de entorno del sistema





2.4. Verificar nuevamente en la consola de comandos que se encuentre la ruta del archivo de activación **mongod** `mongod --version`. Nos debe aparecer lo siguiente.

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\ACER>mongod --version
db version v4.2.7
git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
allocator: tcmalloc
modules: none
build environment:
  distmod: 2012plus
  distarch: x86_64
  target_arch: x86_64

C:\Users\ACER>

```

¡Ya está configurado MongoDB para ejecutarse en la consola de comandos.!

2.5. Ahora, para activar el servicio ***mongod***, ejecutamos en la consola de comandos: **mongod**.

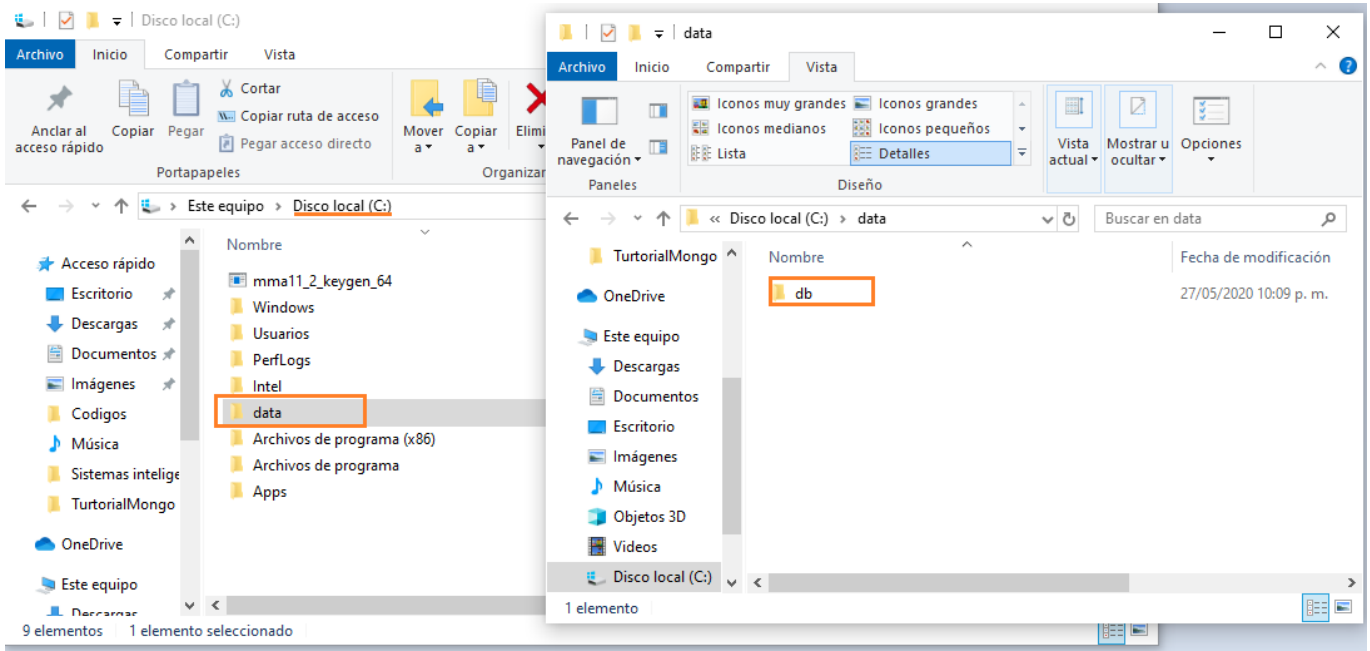
```

C:\Windows\system32\cmd.exe
C:\Users\ACER>mongod ← para encender el servicio de mongo
2020-05-27T22:02:34.293-0500 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-05-27T22:02:34.934-0500 W ASIO [main] No TransportLayer configured during NetworkInterface startup
2020-05-27T22:02:34.940-0500 I CONTROL [initandlisten] MongoDB starting : pid=3756 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-VMG8L9D
2020-05-27T22:02:34.940-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-05-27T22:02:34.940-0500 I CONTROL [initandlisten] db version v4.2.7
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten] git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten] allocator: tcmalloc
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten] modules: none
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten] build environment:
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten]   distmod: 2012plus
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten]   distarch: x86_64
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten]   target_arch: x86_64
2020-05-27T22:02:34.941-0500 I CONTROL [initandlisten] options: {}
2020-05-27T22:02:34.953-0500 I STORAGE [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found. Create the missing directory or specify another path using (1) the --dbpath command line option, or (2) by adding the 'storage.dbPath' option in the configuration file., terminating
2020-05-27T22:02:34.957-0500 I NETWORK [initandlisten] shutdown: going to close listening sockets...
2020-05-27T22:02:34.958-0500 I - [initandlisten] Stopping further Flow Control ticket acquisitions.
2020-05-27T22:02:34.958-0500 I CONTROL [initandlisten] now exiting
2020-05-27T22:02:34.958-0500 I CONTROL [initandlisten] shutting down with code:100

C:\Users\ACER> ← No se mantuvo encendido el servicio

```

Sin embargo, vemos que el servicio se activa por un momento y luego vuelve a quedar inactivo. Esto pasa porque no encuentra la carpeta donde puede mantener activa la información del servicio. Para solucionarlo, vamos a crear la carpeta ***data*** en la ruta **C:** y dentro de esta carpeta creamos la carpeta ***db***.



Una vez creadas las carpetas **data** y **db**, activamos nuevamente el servicio en la consola con el comando **mongod** y ya debería permanecer activo.

```

C:\Windows\system32\cmd.exe - mongod
Microsoft Windows [versión 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\ACER>mongod
2020-05-27T22:13:55.081-0500 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-05-27T22:13:55.085-0500 I ASIO [main] No TransportLayer configured during NetworkInterface startup
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] MongoDB starting : pid=10992 port=27017 dbpath=C:\data\db 64-bit host=DESKTOP-VW68L9D
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] db version v4.2.7
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] git version: 51d9fe12b5d19720e72dcd7db0f2f1dd9a19212
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] allocator: tcmalloc
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] modules: none
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] build environment:
2020-05-27T22:13:55.087-0500 I CONTROL [initandlisten] distmod: 2012plus
2020-05-27T22:13:55.088-0500 I CONTROL [initandlisten] distarch: x86_64
2020-05-27T22:13:55.088-0500 I CONTROL [initandlisten] target_arch: x86_64
2020-05-27T22:13:55.088-0500 I CONTROL [initandlisten] options: {}
2020-05-27T22:13:55.813-0500 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=3521M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress]
2020-05-27T22:13:55.968-0500 I STORAGE [initandlisten] WiredTiger message [1590635635:988705][10992:140732486803024], txn-recover: Set global recovery timestamp: (0, 0)
2020-05-27T22:13:55.924-0500 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-05-27T22:13:55.942-0500 I STORAGE [initandlisten] Timestamp monitor starting
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-05-27T22:13:55.965-0500 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-05-27T22:13:55.965-0500 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2020-05-27T22:13:55.966-0500 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2020-05-27T22:13:55.966-0500 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2020-05-27T22:13:55.967-0500 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2020-05-27T22:13:55.968-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-27T22:13:55.989-0500 I STORAGE [initandlisten] createCollection: admin.system.version with provided UUID: edbd7684-a320-40ee-88bb-6921e2b55baa and options: { uuid: UUID("edbd7684-a320-40ee-88bb-6921e2b55baa") }
2020-05-27T22:13:56.020-0500 I INDEX [initandlisten] index build: done building index_id_0 on ns admin.system.version
2020-05-27T22:13:56.034-0500 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-05-27T22:13:56.035-0500 I COMMAND [initandlisten] Setting featureCompatibilityVersion to 4.2
2020-05-27T22:13:56.038-0500 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-05-27T22:13:56.046-0500 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2020-05-27T22:13:56.053-0500 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-05-27T22:13:56.060-0500 I STORAGE [initandlisten] createCollection: local.startup_log with generated UUID: 28060037-ecde-488b-a8ed-ae4de4776644 and options: { capped: true, size: 10485760 }
2020-05-27T22:13:56.074-0500 I INDEX [initandlisten] index build: done building index_id_0 on ns local.startup_log
2020-05-27T22:13:56.075-0500 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-05-27T22:13:56.673-0500 I FTDC [initandlisten] failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 'El objeto especificado no se encontró en el equipo.'
2020-05-27T22:13:56.674-0500 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2020-05-27T22:13:56.685-0500 I SHARDING [LogicalSessionCacheReap] Marking collection config.system.sessions as collection version: <unsharded>
2020-05-27T22:13:56.686-0500 I NETWORK [listener] Listening on 127.0.0.1
2020-05-27T22:13:56.686-0500 I NETWORK [listener] waiting for connections on port 27017
2020-05-27T22:13:56.690-0500 I CONTROL [LogicalSessionCacheReap] Sessions collection is not set up; waiting until next sessions reap interval: config.system.sessions does not exist
2020-05-27T22:13:56.692-0500 I STORAGE [LogicalSessionCacheRefresh] createCollection: config.system.sessions with provided UUID: 4ca610ca-1246-4bd0-ae84-39b9288e3fea and options: { uuid: UUID("4ca610ca-1246-4bd0-ae84-39b9288e3fea") }
2020-05-27T22:13:56.705-0500 I INDEX [LogicalSessionCacheRefresh] index build: done building index_id_0 on ns config.system.sessions
2020-05-27T22:13:56.718-0500 I INDEX [LogicalSessionCacheRefresh] index build: starting on config.system.sessions properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessions", expireAfterSeconds: 1800 } using method: hybrid
2020-05-27T22:13:56.718-0500 I INDEX [LogicalSessionCacheRefresh] build may temporarily use up to 200 megabytes of RAM
2020-05-27T22:13:56.719-0500 I INDEX [LogicalSessionCacheRefresh] index build: collection scan done, scanned 0 total records in 0 seconds
2020-05-27T22:13:56.721-0500 I INDEX [LogicalSessionCacheRefresh] index build: inserted 0 keys from external sorter into index in 0 seconds
2020-05-27T22:13:56.732-0500 I INDEX [LogicalSessionCacheRefresh] index build: done building index lsidTTLIndex on ns config.system.sessions
2020-05-27T22:13:57.009-0500 I SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>

El servicio se mantiene encendido .

```

Nota: Es importante aclarar que esta consola no debemos cerrarla mientras nos encontremos trabajando con **MongoDB**.

2.6. Finalmente, cada vez que vayamos a trabajar con **MongoDB** vamos a realizar dos pasos:

- Activar el servicio **MongoD** con el comando **mongod**.
- Conectarnos al servicio **MongoD** (en una nueva consola de comandos) con el comando **mongo**.

```
C:\Windows\system32\cmd.exe - mongod
Microsoft Windows [Versi3n 10.0.18362.836]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\ACER>mongod
2020-05-27T22:13:55.801-0500 I CONTROL [main]
-sslDisabledProtocols 'none'
2020-05-27T22:13:55.805-0500 W ASIO [main]
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten]
64-bit host=DESKTOP-VMG8L9D
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] MongoDB shell version v4.2.7
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] Implicit session: session { "id" : UUID("3662e5d7-0dee-4719-a88c-a3f8712a1a91") }
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] MongoDB server version: 4.2.7
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] Welcome to the MongoDB shell.
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] For interactive help, type "help".
2020-05-27T22:13:55.807-0500 I CONTROL [initandlisten] For more comprehensive documentation, see
2020-05-27T22:13:55.808-0500 I CONTROL [initandlisten] http://docs.mongodb.org/
2020-05-27T22:13:55.808-0500 I CONTROL [initandlisten] Questions? Try the support group
2020-05-27T22:13:55.808-0500 I CONTROL [initandlisten] http://groups.google.com/group/mongodb-user
2020-05-27T22:13:55.813-0500 I STORAGE [initandlisten] Server has startup warnings:
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten]
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is
unrestricted.
2020-05-27T22:13:55.964-0500 I CONTROL [initandlisten]
2020-05-27T22:13:55.965-0500 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-05-27T22:13:55.965-0500 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2020-05-27T22:13:55.966-0500 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2020-05-27T22:13:55.966-0500 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2020-05-27T22:13:55.967-0500 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2020-05-27T22:13:55.967-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-27T22:13:55.968-0500 I CONTROL [initandlisten]
---
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()
---
> ← ya estamos conectados al servicio
```

Para verificar que estamos conectados a Mongo podemos ejecutar el siguiente comando: `show dbs`

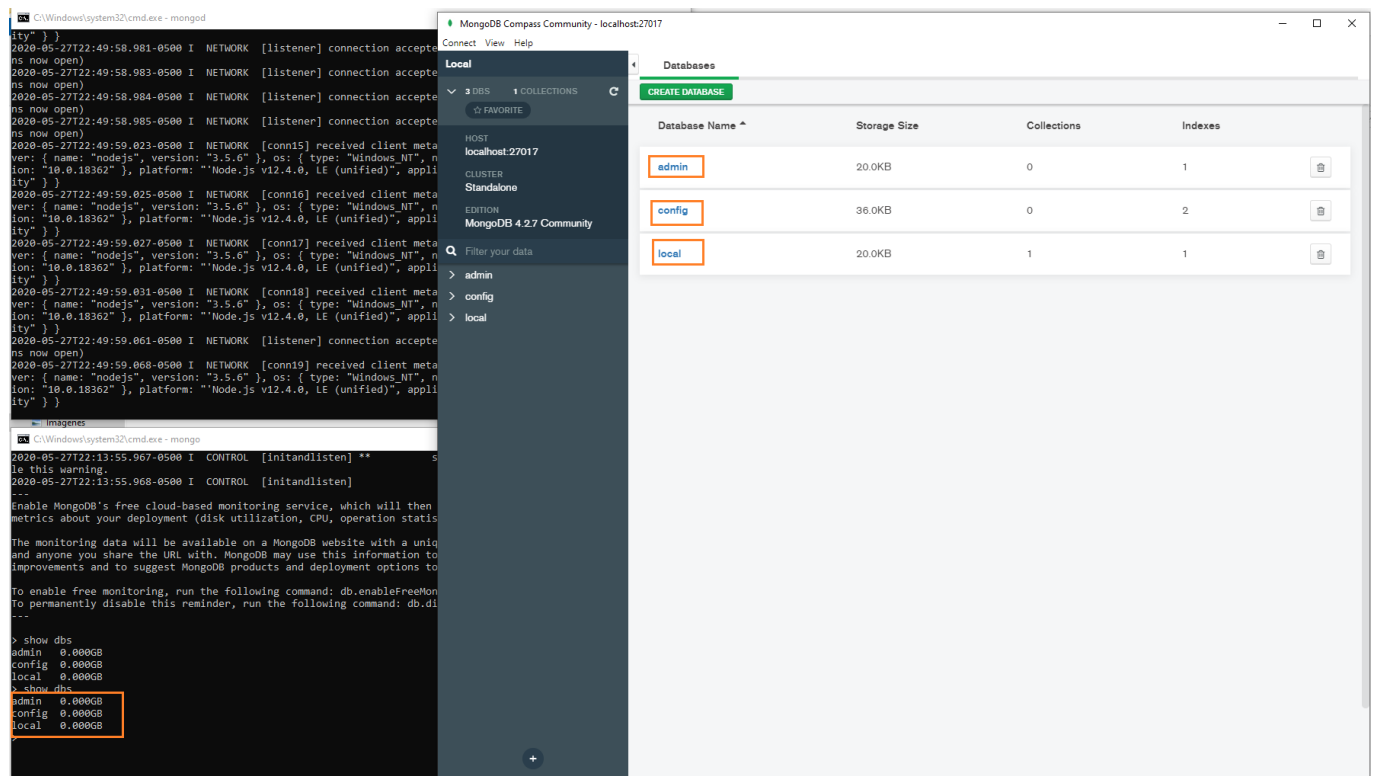
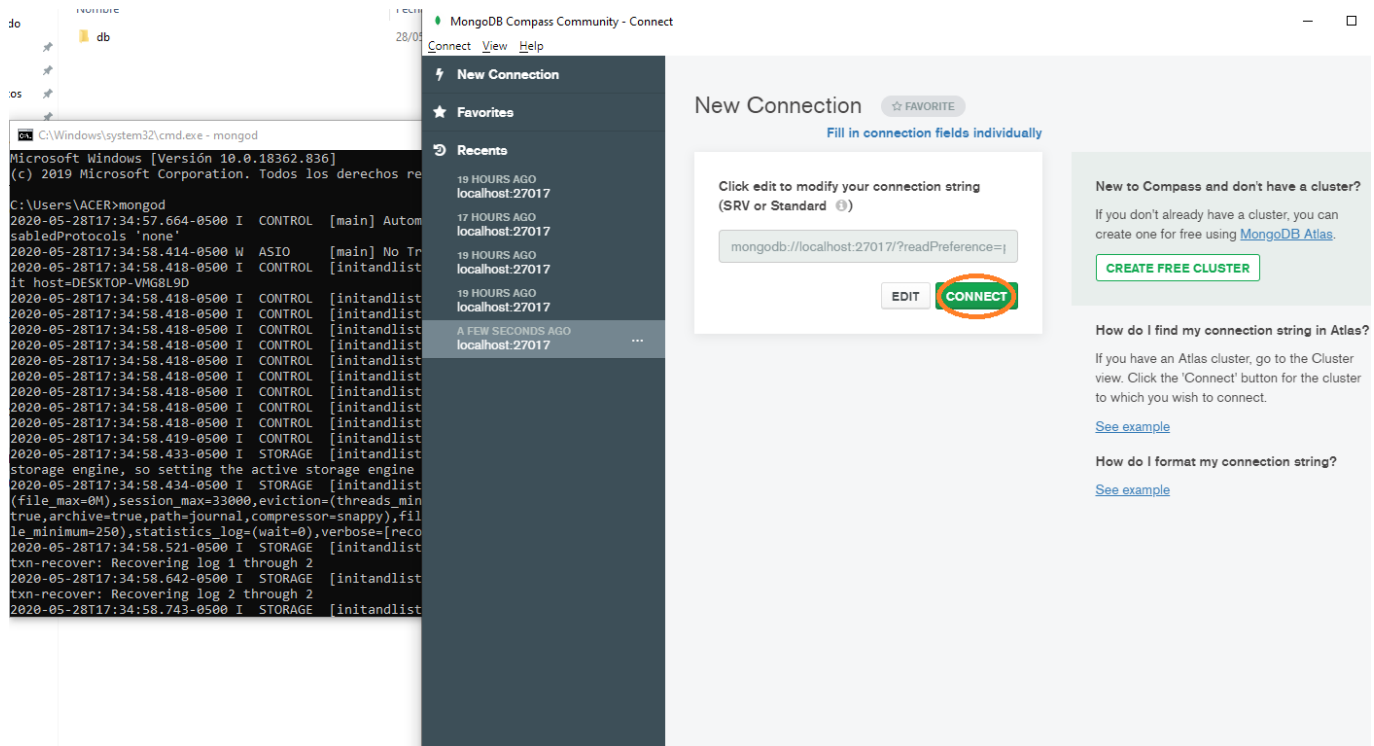
```
C:\Windows\system32\cmd.exe - mongo
2020-05-27T22:13:55.967-0500 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-05-27T22:13:55.968-0500 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display 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()
---

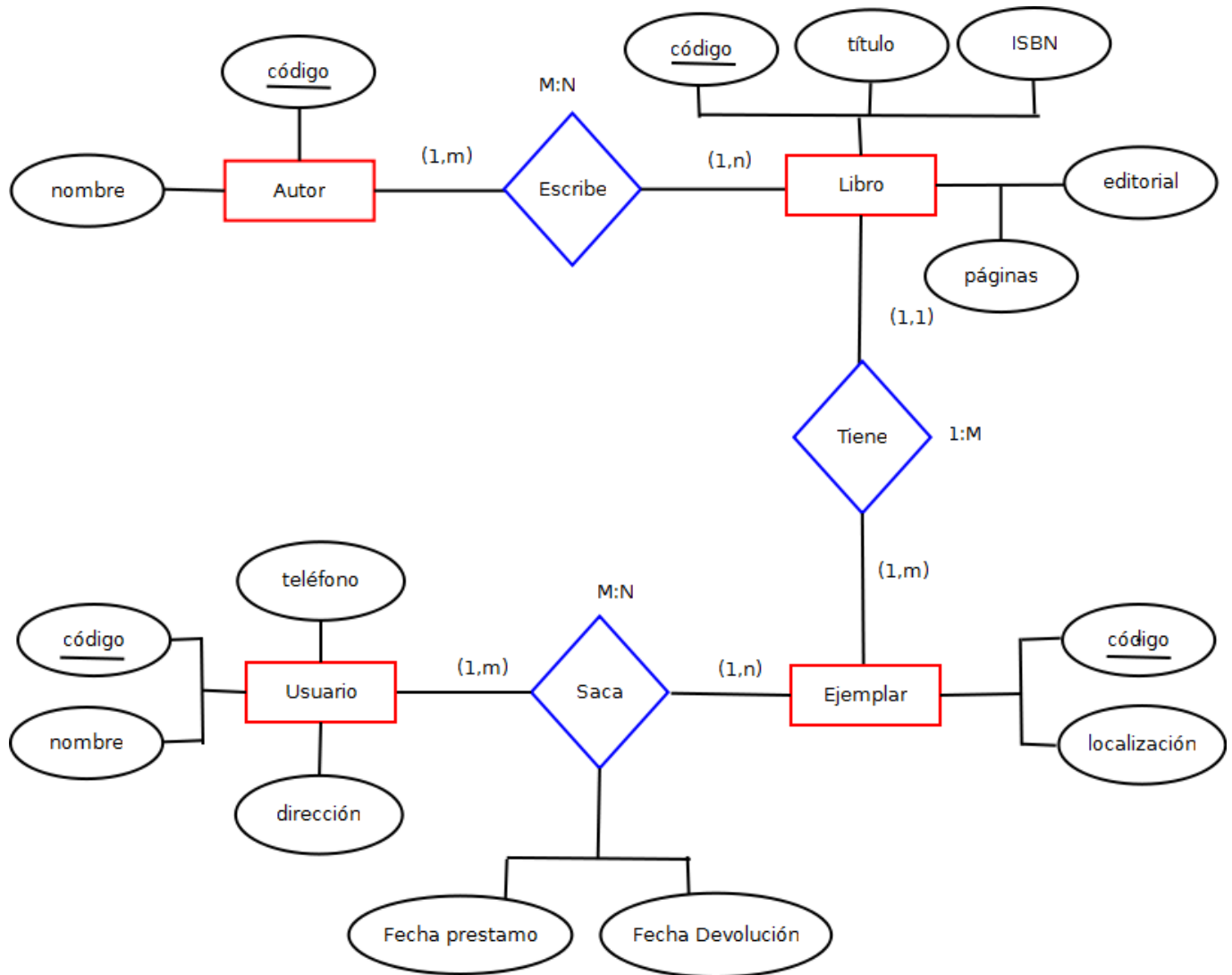
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>
```

Si queremos conectarnos a Mongo a traves de la herramienta GUI (**MongoDB Compass Community**) lo podemos hacer asi:



3. Ejemplo (Libreria) - Base de datos No SQL en MongoDB

Ahora vamos a construir en MongoDB, un ejemplo de una base de datos no relacional para una Libreria. A continuacion presentamos el modelo entidad-relacion correspondiente:

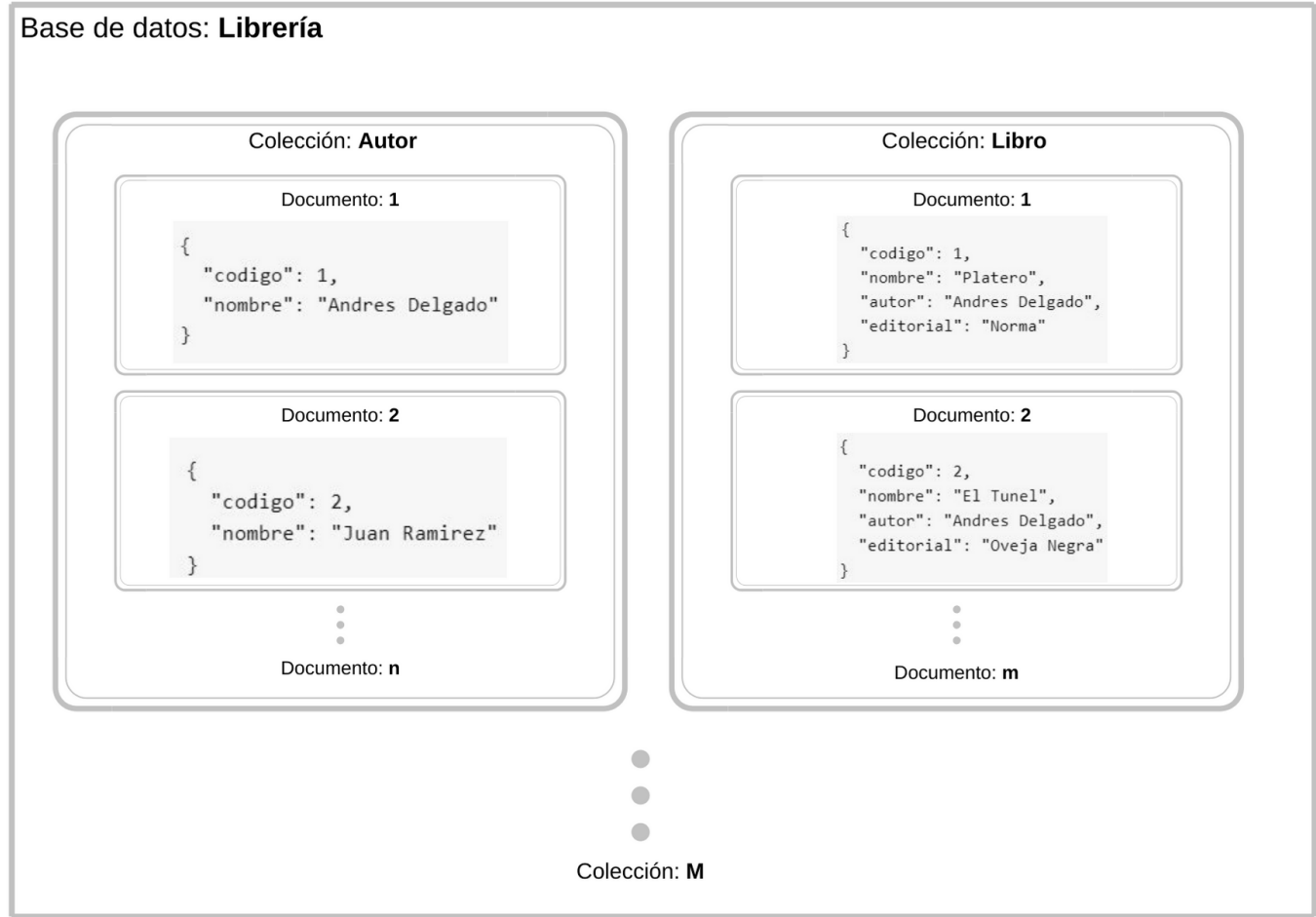


Para representar el modelo ER de la librería en MongoDB resulta de gran utilidad presentar una equivalencia del modelo **SQL** con el modelo **NoSQL**.

SQL Terms/Concepts	MongoDB Terms/Concepts
Database	Database
Table	Collection
Row	Document or BSON Document
Column	Field
Table joins	\$lookup, embedded documents
Primary key: Specify any unique column or column combination as primary key.	Primary key: In MongoDB, the primary key is automatically set to the <code>_id</code> field.

En MongoDB y en general en los modelos no relacionales hay tres conceptos importantes que debemos tener presentes: **base de datos**, **colecciones** y **documentos**. A continuación para el ejemplo de la Librería, se presenta un esquema que muestra como se organizará la información a través de estos tres elementos.

MongoDB



Antes de crear nuestra base de datos vamos a citar algunos comandos que serán de gran utilidad para trabajar con las bases de datos en la consola de MongoDB.

Comando	Funcionamiento
db	Muestra la base de datos actual que estamos utilizando.
show dbs	Muestra las bases de datos que tenemos actualmente.
help	Muestra comandos generales que podemos utilizar.
db.help()	Obtener ayuda con los metodos relacionados a las bases de datos.

3.1. Creación de la base de datos

Para crear una base de datos, utilizamos el comando `use Nombre_base_de_datos` . Con este comando le estamos diciendo a MongoDB que base de datos vamos a utilizar. Sin embargo, Mongo no va a crear la base de datos hasta que adicionemos como minimo una colección.

3.1.1. Crear la base de datos de la Libreria

- Crear base de datos

```
> use Libreria
> db.createCollection("Autor")
```

- Verificar que se creó correctamente la base de datos

```
> show dbs
```

- (Opcional) Si requiero eliminar la base de datos actual

```
> db.DropDatabase()
```



```
C:\Windows\system32\cmd.exe - mongo
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
>
> db
test
>
> use Libreria
switched to db Libreria
>
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
>
> db
Libreria
>
> db.createCollection("Autor")
{ "ok" : 1 }
>
> show dbs
Libreria 0.000GB
admin    0.000GB
config  0.000GB
local    0.000GB
>
```

Visualizar bases de datos existentes

Visualizar base de datos actual

Seleccionar nueva base de datos

La base de datos no ha sido creada aún

Crear primera coleccion en la base de datos

La base de datos ha sido creada

3.2. Adicionar colecciones a la base de datos

Si queremos adicionar colecciones a una base de datos utilizamos: `db.collection("Nombre_colección")`. También podemos crear colecciones de forma implícita, insertando documentos con el comando: `db.Nombre_colección.insert({Documento_BSON})`.

3.2.1 Crear las colecciones necesarias para la Libreria

- Crear las colecciones Autor, Libro, Usuario y Ejemplar.

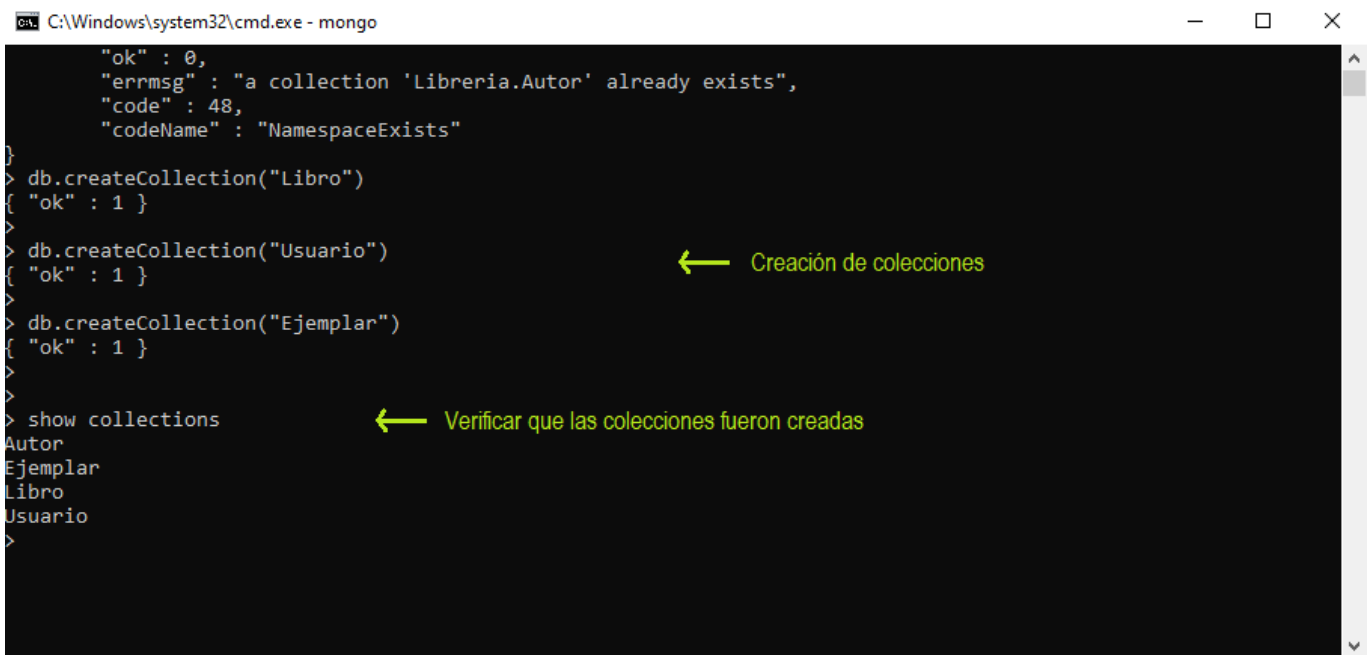
```
> db.createCollection("Autor")
> db.createCollection("Libro")
> db.createCollection("Usuario")
> db.createCollection("Ejemplar")
```

- Verificar que se hayan creado correctamente las colecciones en la base de datos.

```
> show collections
```

- (Opcional) Si requiero eliminar una colección de la base de datos actual

```
> db.Nombre_Colección.Drop()
```



```
C:\Windows\system32\cmd.exe - mongo

{"ok" : 0,
  "errmsg" : "a collection 'Libreria.Autor' already exists",
  "code" : 48,
  "codeName" : "NamespaceExists"
}
> db.createCollection("Libro")
{ "ok" : 1 }
> db.createCollection("Usuario")
{ "ok" : 1 }
> db.createCollection("Ejemplar")
{ "ok" : 1 }
> show collections
Autor
Ejemplar
Libro
Usuario
```

← Creación de colecciones

← Verificar que las colecciones fueron creadas

3.3. Adicionar documentos a las colecciones de la base de datos

Los documentos que se adicionan en las colecciones, son objetos JSON, que mongoDB traduce a objetos binarios (BSON) para hacer mas eficiente operaciones como la busqueda de valores. Estos objetos se componen de parejas Campo-Valor. A continuación presentamos un ejemplo de un documento JSON, que representa la informacion de un usuario.

```
{
  "nombre": "Ivan Delgado",
  "telefono": 2955208,
  "estado": false,
  "direccion": ["Direccion1", "Direccion2"],
  "fecha_creacion": new Date("12/04/2020")
}
```

Los valores en un documento pueden ser diferentes tipos de datos (Strings, numeros, booleanos, matrices, objetos..).

Relaciones en MongoDB

A diferencia de las base de datos SQL, las bases de datos NoSQL, no requieren organizar la información de forma estructurada, lo que puede traer ventajas y desventajas asociadas dependiendo del problema que se quiera resolver. A pesar que en MongoDB no existen relaciones entre elementos, se pueden hacer aproximaciones para los distintos tipos de cardinalidad que pueden encontrarse en un modelo entidad-relacion.

- **Relaciones 1 a 1**

Este tipo de información se da, por ejemplo, entre un edificio y su plano, un estudiante y su historia académica, un trabajador y su curriculum...

Para el caso del estudiante y su historia académica, podemos vincular el identificador del libro en el ejemplar, o al revés. También podríamos incrustar un documento dentro de otro y tener toda la información en un único documento. Hay que tener en consideración como va a ser leído, como va a crecer o si es necesario tener en cuenta la atomicidad.

- **Relaciones 1 a N (entendiendo que N son muchos)**

Este tipo de relación se daría por ejemplo entre una ciudad y sus ciudadanos, un concierto y sus entradas, entre otros.

Si incrustáramos por ejemplo a las personas en el documento inicial, el documento incrementaría su tamaño hasta superar el límite permitido que son 16Mb.

Si incrustamos la información de la ciudad en cada uno de los documentos de la colección persona, vamos a tener una gran probabilidad de generar inconsistencias si tratamos de actualizar la información de la colección ciudad.

Una opción válida sería tener dos colecciones, una que contiene la información de la ciudad y otra que contenga a las N personas, y para cada persona incrustar un atributo que contenga el id de la ciudad a la que pertenece. A continuación presentamos un ejemplo.

Colección Ciudades

```
[
  {
    "_id" : 1,
    "nombre": "Bogota",
    "temperatura": 15,
  },
  {
    "_id" : 2,
    "nombre": "Cali",
    "Temperatura": 30,
  }
]
```


Colección Personas

```
[
  {
    "nombre": "Andres Perez",
    "edad": 30,
    "Ciudad": 1
  },
  {
    "nombre": "Nicolas Gonzalez",
    "edad": 45,
    "Ciudad": 1
  }
]
```

- **Relaciones 1 a N (entendiendo que N son pocos)**

Al contrario del caso anterior, en ocasiones N pueden ser pocos documentos. Por ejemplo si queremos modelar la relacion entre un post y sus comentarios, observamos que no hay muchos comentarios en un mismo Post. (podrian ser del orden de 1 a 10), y como cada comentario esta vinculado exclusivamente a un único Post y rara vez va a ser editado, una buena solución sería incrustar los N comentarios en el post correspondiente. A continuación presentamos un ejemplo.

Colección Posts

```
[
  {
    "_id": 1,
    "Contenido_post": "mensaje posteado",
    "comentarios": [
      {
        "_id": 5,
        "comentario": "respuesta_1"
      },
      {
        "_id": 6,
        "comentario": "respuesta_2"
      }
    ]
  },
]
```

- **Relaciones N a N (Pocos a Pocos o Muchos a Muchos)**

Este tipo de relación, que se daría entre Libros y Autores, o entre Estudiantes y Profesores... es importante analizarla en cada caso concreto, porque en realidad (en estos ejemplos por ejemplo) podrían entenderse como relaciones Pocos a Pocos en gran parte de los casos del mundo real.

Una opción sería, por ejemplo, incrustar los identificadores de cada uno de ellos en un array de identificadores... pero no es lo ideal, ya que suele crear una vulnerabilidad de inconsistencia a la hora de crear los datos en ambas direcciones, ya que no se mantienen juntos (aunque desde el punto de vista del rendimiento, podría ser interesante).

Libros

```
[
  {
    "_id":1,
    "titulo": "Historia del tiempo",
    "autores": [3, 7]
  },
]
```

Autores

```
[
  {
    "_id":3,
    "titulo": "Joseph Fontana",
    "libros": [1, 12, 24]
  },
  {
    "_id":7,
    "titulo": "Gonzalo Ponton",
    "libros": [1]
  }
]
```

Otra opción (para el ejemplo de los libros y autores) es, en lugar de un array de identificadores, incrustar el Libro dentro del Autor, pero esto nos hará duplicar libros y será vulnerable en caso de actualización.

Autores

```
[
  {
    "_id":3,
    "titulo": "Joseph Fontana",
    "libros":[
      {
        "titulo": "Historia del tiempo",
        "Editorial": "Critica"
      },
      {
        "titulo": "El tunel",
        "Editorial": "Alpha&Omega"
      },
    ],
  },
]
```

```

    ]
  },
  {
    "_id": 7,
    "titulo": "Gonzalo Ponton",
    "libros": [
      {
        "titulo": "Historia del tiempo",
        "Editorial": "Critica"
      }
    ]
  }
]

```

Una tercera alternativa sería tener ambos objetos como objetos de primera clase de un único documento general.

Autores-Libros

```

[
  {
    "_id": 7,
    "autor": "Gonzalo Ponton",
    "libro": {
      "titulo": "Historia del tiempo",
      "Editorial": "Critica"
    }
  },
  {
    "_id": 7,
    "autor": "Joseph Fontana",
    "libro": {
      "titulo": "Historia del tiempo",
      "Editorial": "Critica"
    }
  }
]

```

- **Conclusión:**

Es normal pensar que si utilizamos documentos embebidos de mongo estaremos desnormalizando los datos y generando problemas por no respetar la tercera forma normal. Pero en realidad, mientras no dupliquemos los datos, no crearíamos vulnerabilidades.

- En relaciones **uno a uno** es perfectamente viable embeber los datos de forma segura porque no se duplican los datos.
- En relaciones **uno a muchos**, funciona bien si embebes los **muchos** en los **unos**.

- En relaciones **muchos a muchos**, hay que vincularlos a través de un array de identificadores en los documentos. Por algunas razones se podría querer embeber documentos, pero como resultado se duplicaría mucho contenido.

3.3.1. Insertar documentos en las colecciones (Continuemos con nuestro ejemplo de la Librería...)

Para insertar documentos en las colecciones podemos utilizar los siguientes comandos:

Comando	Funcionamiento
<code>db.NameCollection.insert()</code>	Inserta un documento a la colección
<code>db.NameCollection.insertOne()</code>	Inserta un documento a la colección
<code>db.NameCollection.insertMany()</code>	Inserta multiples documentos a una colección

- **3.3.1.1.** Insertar algunos documentos en la colección Autor

```
db.Autor.insertMany([
  {
    "autor": "Gonzalo Ponton",
    "libros": []
  },
  {
    "autor": "Ana Villamil",
    "libros": []
  },
  {
    "autor": "Gabriel Garcia Marquez",
    "libros": []
  }
])
```

Validar que se hayan insertado correctamente los documentos con el comando

`db.Autor.find().pretty()`. Obtendremos como resultado:

```
{
  "_id" : ObjectId("5ed302e92c3fc91319be5394"),
  "autor" : "Gonzalo Ponton",
  "libros" : [ ]
}
{
  "_id" : ObjectId("5ed302e92c3fc91319be5395"),
  "autor" : "Ana Villamil",
  "libros" : [ ]
}
{
  "_id" : ObjectId("5ed330aaa855a6eecd3f609"),
  "autor" : "Gabriel Garcia Marquez",
```

```
"libros" : [ ]
}
```

- **3.3.1.2.** Insertar algunos documentos en la colección Libro

```
db.Libro.insertMany([
  {
    "titulo": "El Tune1",
    "isbn": "978-92-95055",
    "editorial": "Planeta",
    "paginas": 35,
    "autores":[]
  },
  {
    "titulo": "Cien años de soledad",
    "isbn": "978-92-12345",
    "editorial": "Oveja Negra",
    "paginas": 350,
    "autores":[]
  }
])
```

Validar que se hayan insertado correctamente los documentos con el comando

`db.Libro.find().pretty()`. Obtendremos como resultado:

```
{
  "_id" : ObjectId("5ed3157d2c3fc91319be539a"),
  "titulo" : "El Tune1",
  "isbn" : "978-92-95055",
  "editorial" : "Planeta",
  "paginas" : 35,
  "autores" : [ ]
}
{
  "_id" : ObjectId("5ed3157d2c3fc91319be539b"),
  "titulo" : "Cien años de soledad",
  "isbn" : "978-92-12345",
  "editorial" : "Oveja Negra",
  "paginas" : 350,
  "autores" : [ ]
}
```

Ahora vamos a adicionar a cada libro su(s) autor(es). (Para el ejercicio asumimos que los autores de "El tune1" son Ana Villamil y Gonzalo Ponton y de "Cien años de soledad" Gabriel Garcia Marquez).

```
db.Libro.update(
  {"titulo" : "Cien años de soledad"},
  {"$push: {autores: 'Gabriel Garcia Marquez'}}
```

```
{
  $set : {"autores" : [ObjectId("5ed330aaa855a6eecfd3f609")]}
}
)
```

```
db.Libro.update(
  {"titulo" : "El Tune1"},
  {
    $set : {"autores" : [ObjectId("5ed302e92c3fc91319be5395"),
                        ObjectId("5ed302e92c3fc91319be5394") ]}
  }
)
```

Para validar que la información de los autores ha sido actualizada en cada libro, utilizamos el comando `db.Libro.find().pretty()`. Obtendremos como resultado:

```
{
  "_id" : ObjectId("5ed3157d2c3fc91319be539a"),
  "titulo" : "El Tune1",
  "isbn" : "978-92-95055",
  "editorial" : "Planeta",
  "paginas" : 35,
  "autores" : [
    ObjectId("5ed302e92c3fc91319be5395"),
    ObjectId("5ed302e92c3fc91319be5394")
  ]
}
{
  "_id" : ObjectId("5ed3157d2c3fc91319be539b"),
  "titulo" : "Cien años de soledad",
  "isbn" : "978-92-12345",
  "editorial" : "Oveja Negra",
  "paginas" : 350,
  "autores" : [
    ObjectId("5ed330aaa855a6eecfd3f609")
  ]
}
```

Ahora vamos a adicionar a cada autor su(s) libro(s).

```
db.Autor.update(
  {"autor" : "Gonzalo Ponton"},
  {
    $set : {"libros" : [ObjectId("5ed3157d2c3fc91319be539a")]}
  }
)
```

```
db.Autor.update(
  {"autor" : "Ana Villamil"},
  {
    $set : {"libros" : [ObjectId("5ed3157d2c3fc91319be539a")]}
  }
)
```

```
db.Autor.update(
  {"autor" : "Gabriel Garcia Marquez"},
  {
    $set : {"libros" : [ObjectId("5ed3157d2c3fc91319be539b")]}
  }
)
```

Para validar que la información de los libros ha sido actualizada en cada autor, utilizamos el comando `db.Autor.find().pretty()`. Obtendremos como resultado:

```
{
  "_id" : ObjectId("5ed302e92c3fc91319be5394"),
  "autor" : "Gonzalo Ponton",
  "libros" : [
    ObjectId("5ed3157d2c3fc91319be539a")
  ]
}
{
  "_id" : ObjectId("5ed302e92c3fc91319be5395"),
  "autor" : "Ana Villamil",
  "libros" : [
    ObjectId("5ed3157d2c3fc91319be539a")
  ]
}
{
  "_id" : ObjectId("5ed330aaa855a6eecd3f609"),
  "autor" : "Gabriel Garcia Marquez",
  "libros" : [
    ObjectId("5ed3157d2c3fc91319be539b")
  ]
}
```

- **3.3.1.3.** Insertar algunos documentos en la colección Usuario

```
db.Usuario.insertMany([
  {
    "nombre": "Juan Perez",
    "telefono": "3145985225",
    "direccion": "cra 5 # 23-22"
```

```

    },
    {
      "autor": "Maria Andrade",
      "telefono": "3107985835",
      "direccion": "calle 5 # 2-52"
    }
  ])

```

Validar que se hayan insertado correctamente los documentos con el comando `db.Usuario.find().pretty()`. Obtendremos como resultado:

```

{
  "_id" : ObjectId("5ed30c9a2c3fc91319be5396"),
  "nombre" : "Juan Perez",
  "telefono" : "3145985225",
  "direccion" : "cra 5 # 23-22"
}
{
  "_id" : ObjectId("5ed30c9a2c3fc91319be5397"),
  "autor" : "Maria Andrade",
  "telefono" : "3107985835",
  "direccion" : "calle 5 # 2-52"
}

```

- **3.3.1.4.** Insertar algunos documentos en la colección Ejemplar (Vamos a crear dos ejemplares por cada libro, y cada ejemplar estará ubicado en una biblioteca diferente). Los valores que se están insertando en el campo "libro", corresponden al id de los libros previamente insertados.

```

db.Ejemplar.insertMany([
  {
    "libro": ObjectId("5ed3157d2c3fc91319be539a"),
    "Localizacion": "Biblioteca Virgilio Barco"
  },
  {
    "libro": ObjectId("5ed3157d2c3fc91319be539a"),
    "Localizacion": "Biblioteca el Tintal"
  },
  {
    "libro": ObjectId("5ed3157d2c3fc91319be539b"),
    "Localizacion": "Biblioteca Virgilio Barco"
  },
  {
    "libro": ObjectId("5ed3157d2c3fc91319be539b"),
    "Localizacion": "Biblioteca el Tintal"
  },
])

```


Validar que se hayan insertado correctamente los documentos con el comando `db.Ejemplar.find().pretty()`. Obtendremos como resultado:

```
{
  "_id" : ObjectId("5ed319a12c3fc91319be539c"),
  "libro" : ObjectId("5ed3157d2c3fc91319be539a"),
  "Localizacion" : "Biblioteca Virgilio Barco"
}
{
  "_id" : ObjectId("5ed319a12c3fc91319be539d"),
  "libro" : ObjectId("5ed3157d2c3fc91319be539a"),
  "Localizacion" : "Biblioteca el Tintal"
}
{
  "_id" : ObjectId("5ed319a12c3fc91319be539e"),
  "libro" : ObjectId("5ed3157d2c3fc91319be539b"),
  "Localizacion" : "Biblioteca Virgilio Barco"
}
{
  "_id" : ObjectId("5ed319a12c3fc91319be539f"),
  "libro" : ObjectId("5ed3157d2c3fc91319be539b"),
  "Localizacion" : "Biblioteca el Tintal"
}
```

- **3.3.1.5.** Vamos a crear una colección para registrar la información relacionada con los préstamos de los ejemplares.

```
> db.createCollection("HistorialPrestamos")
```

Insertar algunos documentos en la colección HistorialPrestamos. Cada una de estas inserciones simularán el registro del préstamo de ejemplares. Los valores de los campos "usuario" y "ejemplar", contienen el id de algunos de los documentos de las colecciones de Usuario y Ejemplar.

```
db.HistorialPrestamos.insertMany([
  {
    "usuario": ObjectId("5ed30c9a2c3fc91319be5396"),
    "ejemplar": ObjectId("5ed319a12c3fc91319be539c"),
    "FechaPrestamo": new Date(20/04/2020),
    "FechaDevolucion": new Date(25/04/2020)
  },
  {
    "usuario": ObjectId("5ed30c9a2c3fc91319be5396"),
    "ejemplar": ObjectId("5ed319a12c3fc91319be539e"),
    "FechaPrestamo": new Date(20/05/2020),
    "FechaDevolucion": new Date(29/05/2020)
  },
  {
    "usuario": ObjectId("5ed30c9a2c3fc91319be5397"),
```

```

    "ejemplar": ObjectId("5ed319a12c3fc91319be539d"),
    "FechaPrestamo": new Date(02/01/2020),
    "FechaDevolucion": new Date(15/01/2020)
  }
])

```

Validar que se hayan insertado correctamente los documentos con el comando `db.HistorialPrestamos.find().pretty()`. Obtendremos como resultado:

```

{
  "_id" : ObjectId("5ed3211d2c3fc91319be53a0"),
  "usuario" : ObjectId("5ed30c9a2c3fc91319be5396"),
  "ejemplar" : ObjectId("5ed319a12c3fc91319be539c"),
  "FechaPrestamo" : ISODate("1970-01-01T00:00:00Z"),
  "FechaDevolucion" : ISODate("1970-01-01T00:00:00Z")
}
{
  "_id" : ObjectId("5ed3211d2c3fc91319be53a1"),
  "usuario" : ObjectId("5ed30c9a2c3fc91319be5396"),
  "ejemplar" : ObjectId("5ed319a12c3fc91319be539e"),
  "FechaPrestamo" : ISODate("1970-01-01T00:00:00Z"),
  "FechaDevolucion" : ISODate("1970-01-01T00:00:00Z")
}
{
  "_id" : ObjectId("5ed3211d2c3fc91319be53a2"),
  "usuario" : ObjectId("5ed30c9a2c3fc91319be5397"),
  "ejemplar" : ObjectId("5ed319a12c3fc91319be539d"),
  "FechaPrestamo" : ISODate("1970-01-01T00:00:00Z"),
  "FechaDevolucion" : ISODate("1970-01-01T00:00:00Z")
}

```

3.3.2. Realizando consultas en MongoDB

Finalmente vamos a realizar algunas consultas sobre nuestra base de datos.

- **3.3.2.1.** Consultar la informacion del libro Cien años de Soledad.

```
db.Libro.findOne({"titulo":"Cien años de soledad"})
```

Resultado:

```

{
  "_id" : ObjectId("5ed3157d2c3fc91319be539b"),
  "titulo" : "Cien años de soledad",
  "isbn" : "978-92-12345",
  "editorial" : "Oveja Negra",
  "paginas" : 350,

```

```
    "autores" : [
      ObjectId("5ed330aaa855a6eecd3f609")
    ]
  }
```

- **3.3.2.2.** Consultar cuantos ejemplares hay del libro Cien años de Soledad.

```
libro = db.Libro.findOne({"titulo":"Cien años de soledad"})
numEjemplares = db.Ejemplar.find({"libro":libro._id}).count()
print ("El libro "+libro.titulo+" tiene "+numEjemplares+" ejemplares")
```

Resultado:

```
El libro Cien años de soledad tiene 2 ejemplares
```

- **3.3.2.3.** Consultar que libros le han prestado a Juan Perez

```
juan = db.Usuario.findOne({"nombre":"Juan Perez"})
db.HistorialPrestamos.find({"usuario":juan._id}).forEach(
  (prestamo)=>{
    ejemplar = db.Ejemplar.findOne({"_id":prestamo.ejemplar})
    libro = db.Libro.findOne({"_id":ejemplar.libro})
    print("Libro: " + libro.titulo)
  }
)
```

Resultado:

```
Libro: El Tunel
Libro: Cien años de soledad
```

Nota: A través de la herramienta GUI (Mongo Compass Community), puede gestionar casi que al mismo nivel de detalle toda la información relacionada con las bases de datos, colecciones y documentos. A continuación podrá observar desde la interfaz gráfica la construcción de la base de datos de la Librería.

MongoDB Compass Community - localhost:27017/Libreria

Connect View Help

Local

4 DBS6 COLLECTIONS

☆ FAVORITE

HOSTlocalhost:27017

CLUSTERStandalone

EDITIONMongoDB 4.2.7 Community

Filter your data

Libreria

Autor

Ejemplar

HistorialPrestamos

Libro

Usuario

admin

config

local

Collections

CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Autor	3	78.0 B	234.0 B	1	36.0 KB	
Ejemplar	4	82.5 B	330.0 B	1	20.0 KB	
HistorialPrestamos	3	113.0 B	339.0 B	1	20.0 KB	
Libro	2	151.0 B	302.0 B	1	20.0 KB	
Usuario	2	100.5 B	201.0 B	1	36.0 KB	

MongoDB Compass Community - localhost:27017/Libreria.Autor

Connect View Collection Help

Local

4 DBS6 COLLECTIONS

☆ FAVORITE

HOSTlocalhost:27017

CLUSTERStandalone

EDITIONMongoDB 4.2.7 Community

Filter your data

Libreria

Autor

Ejemplar

HistorialPrestamos

Libro

Usuario

admin

config

local

Libreria.Autor Documents

Libreria.Autor

DOCUMENTS 3

TOTAL SIZE 234B

AVG. SIZE 78B

INDEXES 1

TOTAL SIZE 36.0KB

AVG. SIZE 36.0KB

Documents

Aggregations

Explain Plan

Indexes

FILTER

OPTIONS

FIND

RESET

ADD DATA

VIEW

Displaying documents 1 - 3 of 3

REFRESH

_id: ObjectId("5ed302e92c3fc91319be5394")

autor: "Gonzalo Ponton"

libros: Array

_id: ObjectId("5ed302e92c3fc91319be5395")

autor: "Ana Villamil"

libros: Array

_id: ObjectId("5ed330aaa855a6eecd3f609")

autor: "Gabriel Garcia Marquez"

libros: Array

28 / 30

Libreria.Libro

Documents

Documents

Aggregations

Explain Plan

Indexes

DOCUMENTS 2

TOTAL SIZE 302B

AVG. SIZE 151B

INDEXES 1

TOTAL SIZE 20.0KB

AVG. SIZE 20.0KB

FILTER

OPTIONS

FIND

RESET

...

ADD DATA

VIEW

Displaying documents 1 - 2 of 2

REFRESH

	_id ObjectId	titulo String	isbn String	editorial String	paginas Double	autores Array
1	Sed3157d2c3fc91319be539a	"El Tune1"	"978-92-95055"	"Planeta"	35	[] 2 elements
2	Sed3157d2c3fc91319be539b	"Cien años de soledad"	"978-92-12345"	"Oveja Negra"	350	[] 1 elements

The screenshot shows the MongoDB Compass web interface. At the top, there's a breadcrumb 'Libreria.Libro' and a 'Documents' tab. The main header shows 'Libreria.Libro' and 'DOCUMENTS 2'. Below the header, there are tabs for 'Documents', 'Aggregations', 'Explain Plan', and 'Indexes'. A 'FILTER' button is on the left. Below the filter, there's an 'ADD DATA' button and a 'VIEW' section with icons for list, JSON, and grid views. The main area displays two JSON documents. The first document represents a book 'El Tunel' by 'Planeta' with 35 pages and two authors. The second document represents 'Cien años de soledad' by 'Oveja Negra' with 350 pages and one author.

```
{
  "_id": {
    "$oid": "5ed3157d2c3fc91319be539a"
  },
  "titulo": "El Tunel",
  "isbn": "978-92-95055",
  "editorial": "Planeta",
  "paginas": 35,
  "autores": [
    {
      "$oid": "5ed302e92c3fc91319be5395"
    },
    {
      "$oid": "5ed302e92c3fc91319be5394"
    }
  ]
}

{
  "_id": {
    "$oid": "5ed3157d2c3fc91319be539b"
  },
  "titulo": "Cien años de soledad",
  "isbn": "978-92-12345",
  "editorial": "Oveja Negra",
  "paginas": 350,
  "autores": [
    {
      "$oid": "5ed330aaa855a6eecd3f609"
    }
  ]
}
```

1. Referencias

1. [Guia de instalación y configuración](#)
2. [Guia de referencia y documentación oficial](#)
3. [Relaciones en MongoDB](#)