

## Sesión 4 – Manejo de una base de datos con información multimedia en MongoDB.

Esta sesión la dividiremos en tres actividades. Cada actividad, a su vez, está dividida en varios ejercicios.

### Preámbulo.

Implementaremos una base de datos sencilla en MongoDB. Lo que vamos a construir en estas dos sesiones es una base de datos de películas, y la evaluación de ellas por distintos usuarios con un valor de cero a cinco. En la primera sesión incluiremos algunos datos de tipo tradicional, y en la segunda sesión otros datos multimedia. Los datos multimedia serán fotografías del póster vinculados a cada película. La información que tenemos es la siguiente:

1. Una tabla maestra, que hemos descargado de un dataset público de la web Kaggle que no está disponible en formato Json. La página es: <https://www.kaggle.com/rounakbanik/the-movies-dataset/>. Las obtuvimos en formato CSV, y la transformamos previamente a JSON para que fuese más amigable con MongoDB. Contiene información de la base de datos de películas de la web [www.imdb.com](http://www.imdb.com) (internet movies database).

Las columnas que nos proporciona esa fuente de datos gratuita son:

- adult: Si tiene contenido para adultos o no
- belongs\_to\_collection: Datos de la saga de la película, si pertenece.
- budget: Presupuesto
- genres: Género (Comedia, acción, etc.)
- homepage: Página web.
- id: Código único de la película.
- imdb\_id: Código único de la película en la base de datos IMDB.
- original\_language: Idioma original.
- original\_title: Título original.
- overview: Resumen.
- popularity: Índice de popularidad.
- poster\_path: Nombre del archivo del póster (no lo tenemos).
- production\_companies: Productoras.
- production\_countries: Países de producción.
- release\_date: Fecha de lanzamiento.
- revenue: Ingresos.
- runtime: Duración.
- spoken\_languages: Idiomas.
- status: Estado.
- tagline: Etiqueta.
- title: Título.
- video: Booleano que indicas si es de cine o de video.
- vote\_average: Promedio de evaluaciones entre todos los usuarios
- vote\_count: número de usuarios que la han votado.



2. Un conjunto de fotos de los pósteres de algunas de las películas (cada película puede tener ninguna, una o varias fotografías de póster). Las fotos tienen una descripción del póster.

Para el desarrollo de esta práctica asumiremos que la máquina está configurada como una del laboratorio con el sistema operativo Windows, y que tiene MongoDB versión 3.2 instalado, es decir:

C:\mongodb

**IMPORTANTE:** En esta práctica nos referiremos a la carpeta donde está MongoDB como "Program Files". En algunas versiones de Windows deberéis sustituir esta carpeta por "Archivos de programa".

Si queréis realizar esta práctica en otras máquinas, podéis descargar el servidor de bases de datos gratuitamente en la siguiente dirección web:

<https://www.mongodb.org/downloads>

Nosotros vamos a trabajar en el sistema operativo Windows, sin embargo, existen versiones para distintos sistemas operativos: Linux, MAC OS X, y Solaris. La versión del laboratorio es la 3.2, aunque a fecha de febrero 2020 ya van por la 4.2.3

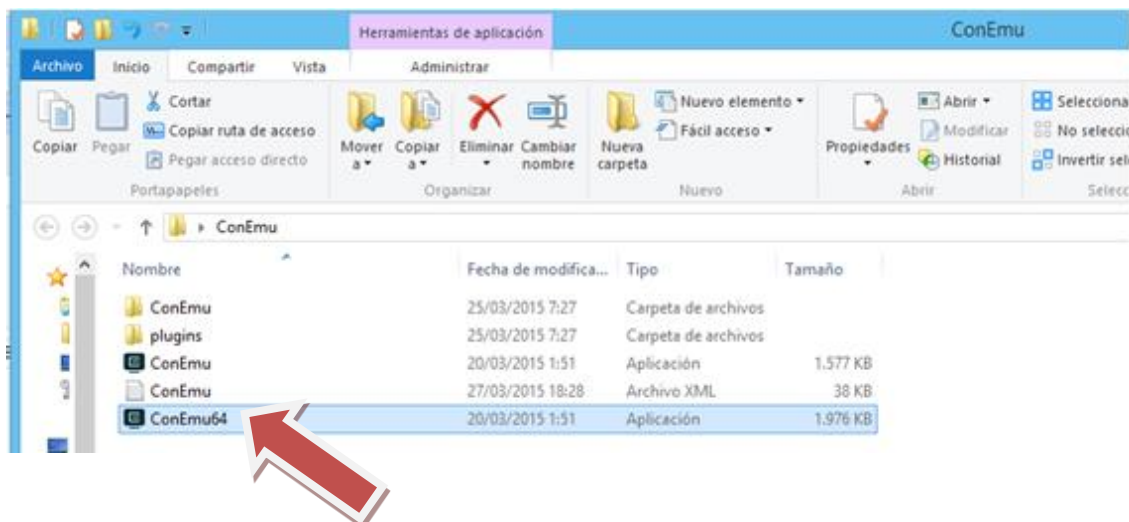
## Actividad 1: Configuración preliminar: Acondicionamiento del entorno de trabajo.

**Abstract:** Vamos a dejar el entorno del sistema preparado para poder realizar la práctica.

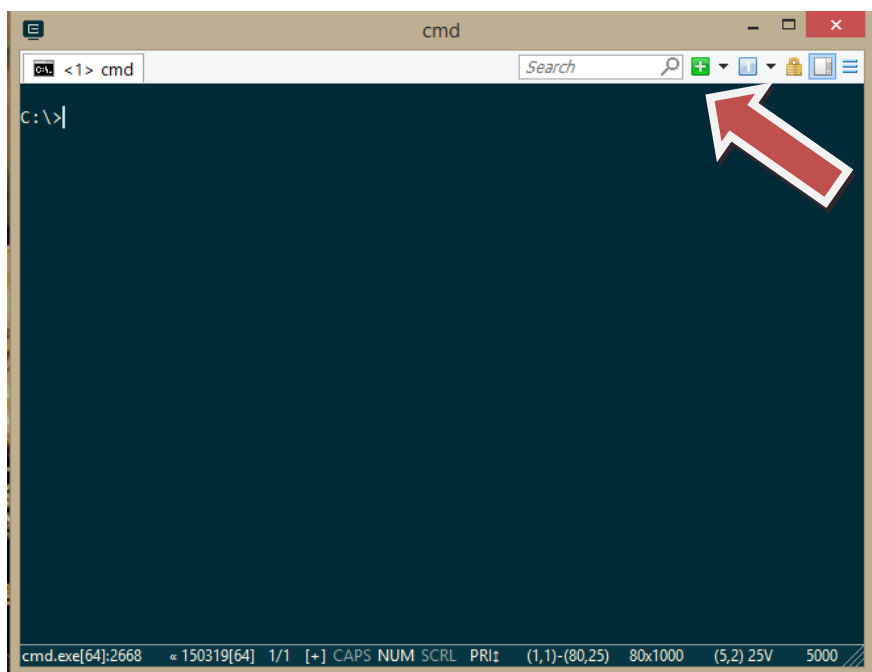
**Paso 1:** Descarga los materiales desde el campus virtual, y descomprímelos en una carpeta en el escritorio de Windows. Abre la carpeta de materiales, y verás el fichero "movies\_metadata.json":

Copia el archivo de trabajo en la carpeta c:\datosMongo. Comprueba que se ha copiado correctamente.

**Paso 2:** Desarrollaremos la práctica utilizando un emulador gratuito de consola de MS-DOS llamado "ConEmu", que se encuentra también entre los materiales descargables desde el Campus Virtual. Nos permitirá poder interactuar con varias consolas de comandos simultáneamente. Para hacerlo funcionar, tan solo debes ubicar la carpeta en el escritorio de tu máquina. No necesita instalación. Hecho esto, ejecuta esta aplicación:



El resultado debe ser similar a éste:



Crea tres pestañas, que usaremos más adelante:

- La <1> para servir para tener el servidor de bases de datos activo.
- La <2> para ejecutar sentencias de MongoDB.
- La <3> para ejecutar comandos del sistema operativo desde esta consola.

Las pestañas se crean pulsando el botón verde con la cruz blanca en el centro que se encuentra en la parte superior derecha de la aplicación.

**IMPORTANTE:** ConEmu permite pegar directamente del portapapeles los comandos a ejecutar. Por ello puedes copiarlos desde este documento, y pegar el texto ahí. En Windows 10 puedes hacer esto mismo con tres ventanas de comandos (cmd), no obstante, te podría resultar algo más incómodo.

## Actividad 2: Puesta en marcha del sistema.

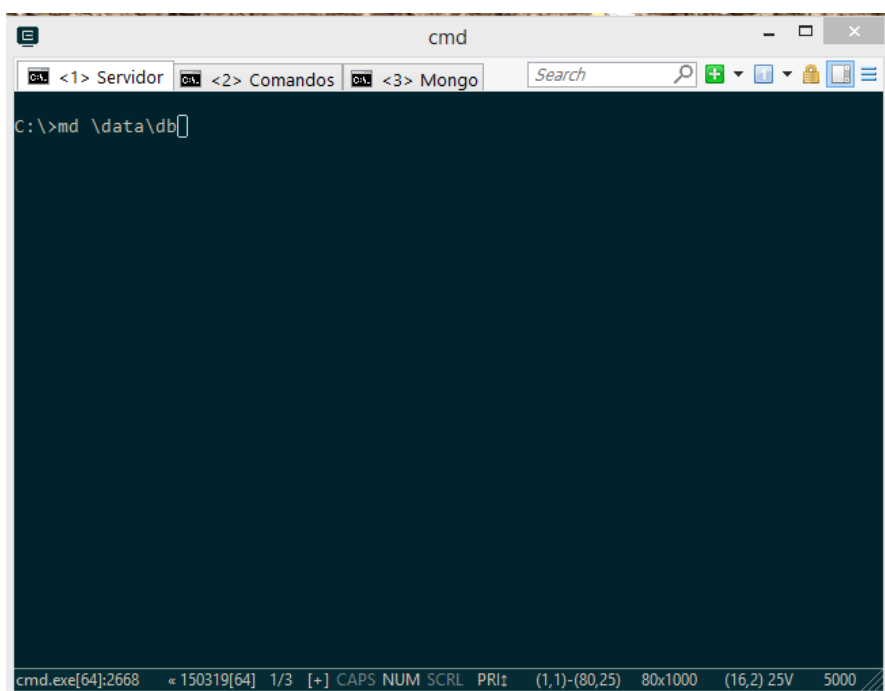
### Ejercicio 1: Arranque del servidor de base de datos.

**Abstract:** Vamos a poner en marcha el servidor de base de datos MongoDB.

MongoDB busca por defecto los archivos de bases de datos en la carpeta `c:\data\db`. Comprueba si existe, y si no crea esa carpeta desde la consola <1>. Para ello ejecutaremos el comando:

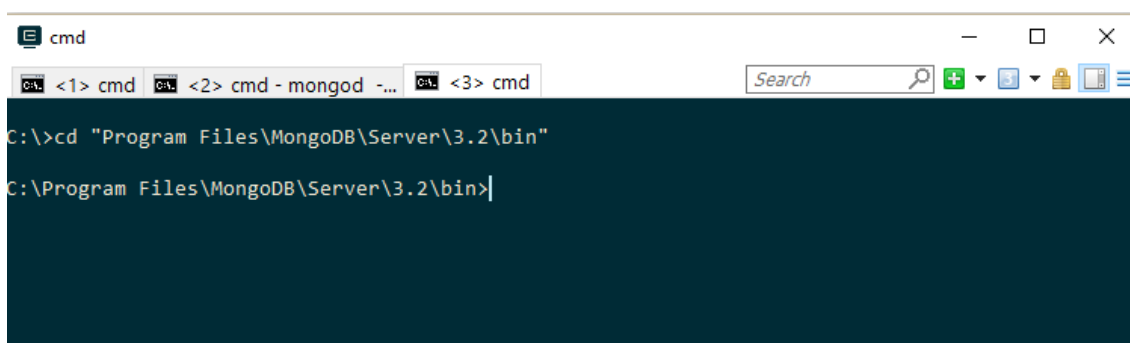
**`md c:\data\db`**

tal como se muestra en la siguiente imagen:



Hecho esto, movámonos a la carpeta donde se encuentra la aplicación del gestor de MongoDB mediante el comando:

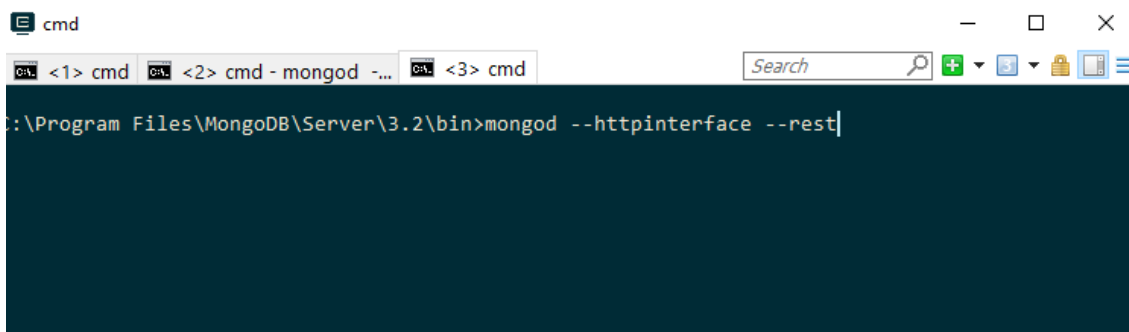
**`cd "c:\program files\mongoDB\server\3.2\bin"`**



Ejecutemos ahora el servidor de la base de datos mediante el siguiente comando:

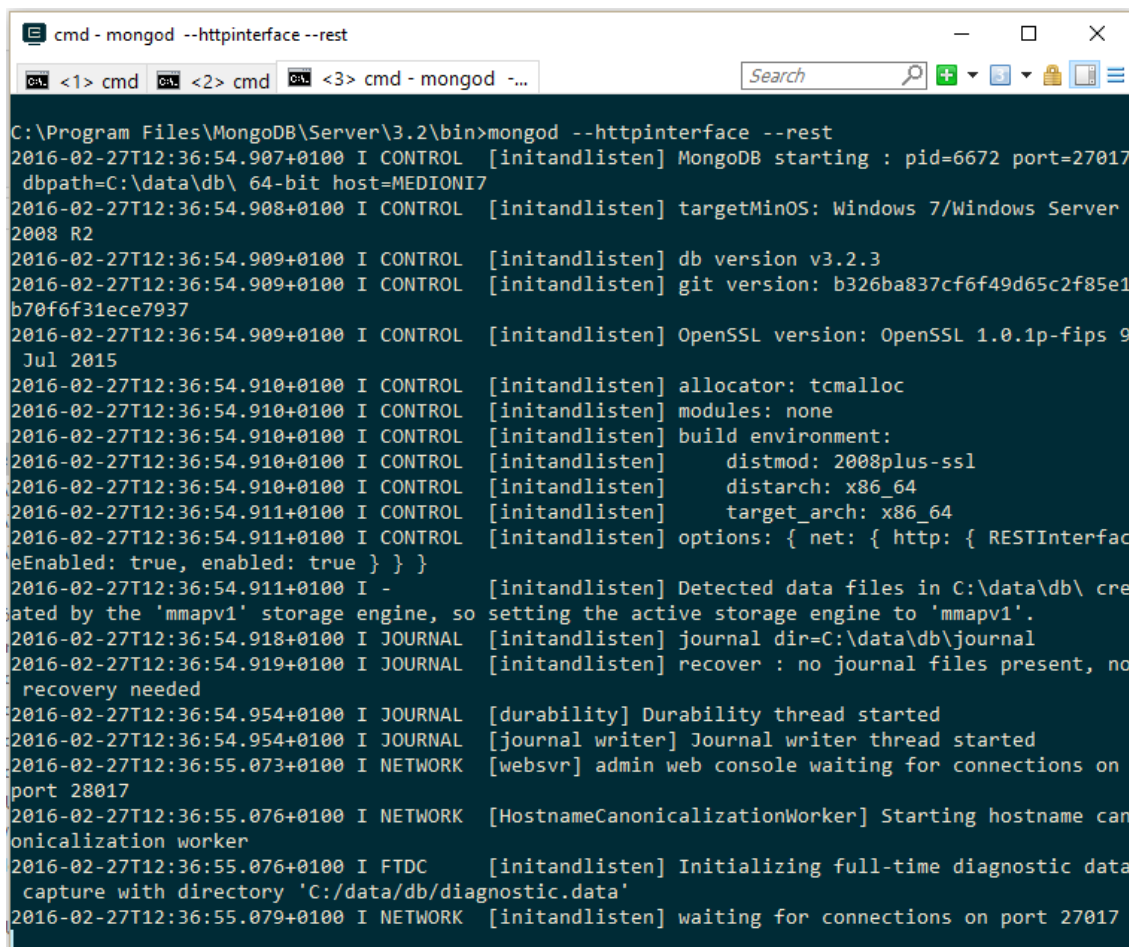
**mongod --httpinterface --rest**

Con este modo usaremos todos los parámetros del servidor por defecto, excepto las opciones `--httpinterface` y `--rest`. Ellas nos permitirán monitorizar su estado desde un navegador web por medio del protocolo http.



```
cmd
C:\Program Files\MongoDB\Server\3.2\bin>mongod --httpinterface --rest
```

El resultado debe ser algo similar al siguiente:



```
cmd - mongod --httpinterface --rest
C:\Program Files\MongoDB\Server\3.2\bin>mongod --httpinterface --rest
2016-02-27T12:36:54.907+0100 I CONTROL [initandlisten] MongoDB starting : pid=6672 port=27017
dbpath=C:\data\db\ 64-bit host=MEDIONI7
2016-02-27T12:36:54.908+0100 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server
2008 R2
2016-02-27T12:36:54.909+0100 I CONTROL [initandlisten] db version v3.2.3
2016-02-27T12:36:54.909+0100 I CONTROL [initandlisten] git version: b326ba837cf6f49d65c2f85e1
b70f6f31ece7937
2016-02-27T12:36:54.909+0100 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1p-fips 9
Jul 2015
2016-02-27T12:36:54.910+0100 I CONTROL [initandlisten] allocator: tcmalloc
2016-02-27T12:36:54.910+0100 I CONTROL [initandlisten] modules: none
2016-02-27T12:36:54.910+0100 I CONTROL [initandlisten] build environment:
2016-02-27T12:36:54.910+0100 I CONTROL [initandlisten] distmod: 2008plus-ssl
2016-02-27T12:36:54.910+0100 I CONTROL [initandlisten] distarch: x86_64
2016-02-27T12:36:54.911+0100 I CONTROL [initandlisten] target_arch: x86_64
2016-02-27T12:36:54.911+0100 I CONTROL [initandlisten] options: { net: { http: { RESTInterfac
eEnabled: true, enabled: true } } }
2016-02-27T12:36:54.911+0100 I - [initandlisten] Detected data files in C:\data\db\ cre
ated by the 'mmapv1' storage engine, so setting the active storage engine to 'mmapv1'.
2016-02-27T12:36:54.918+0100 I JOURNAL [initandlisten] journal dir=C:\data\db\journal
2016-02-27T12:36:54.919+0100 I JOURNAL [initandlisten] recover : no journal files present, no
recovery needed
2016-02-27T12:36:54.954+0100 I JOURNAL [durability] Durability thread started
2016-02-27T12:36:54.954+0100 I JOURNAL [journal writer] Journal writer thread started
2016-02-27T12:36:55.073+0100 I NETWORK [websvr] admin web console waiting for connections on
port 28017
2016-02-27T12:36:55.076+0100 I NETWORK [HostnameCanonicalizationWorker] Starting hostname can
onicalization worker
2016-02-27T12:36:55.076+0100 I FTDC [initandlisten] Initializing full-time diagnostic data
capture with directory 'C:/data/db/diagnostic.data'
2016-02-27T12:36:55.079+0100 I NETWORK [initandlisten] waiting for connections on port 27017
```

En este punto ya tendremos en marcha el servidor de base de datos con su configuración por defecto (dirección IP localhost, y puerto de escucha 27017), listo para recibir peticiones.

## Ejercicio 2. Monitorizar el servidor mediante un navegador Web.

**Abstract:** Vamos a ver en un navegador web el estado del servidor MongoDB que acabamos de poner en marcha.

Ejecuta un navegador web (el que quieras), y accede a la url:

**http://localhost:28017**

MongoDB escucha el protocolo http por el puerto sumadas 1000 unidades al que escucha peticiones de base de datos. En nuestro caso, el 28017, ya que el de base de datos es el 27017.

[illegible]

En este punto podremos ver el estado del servidor de base de datos.

Investiga por las pestañas para ver lo que nos muestra sobre el estado del servidor. Observa que mucha información nos la muestra en formato Json, el que utiliza MongoDB para almacenar los datos.

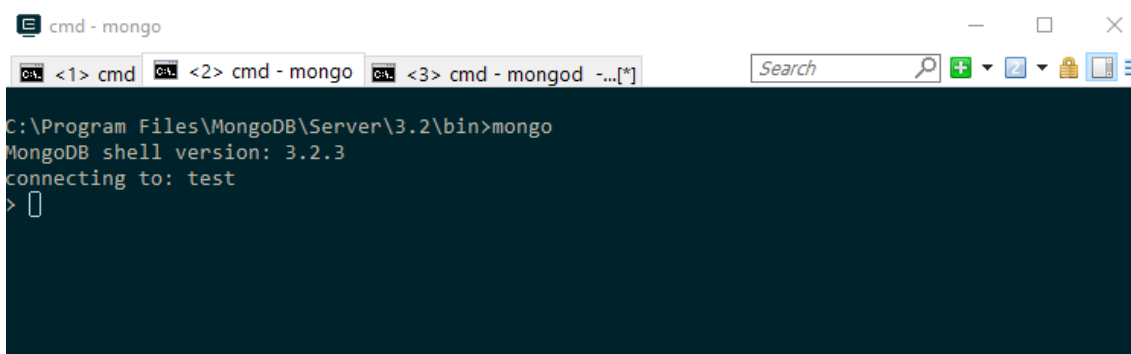
### Ejercicio 3. Poner en marcha la aplicación cliente de MongoDB que permite ejecutar comandos de base de datos.

**Abstract:** Vamos a ejecutar la aplicación Mongo.exe, que nos permitirá interactuar con el servidor de bases de datos MongoDB.

Activa la pantalla de comandos <2> pulsando sobre ella. Movámonos a la carpeta donde se encuentra la aplicación de interacción con el servidor MongoDB mediante el comando:

**cd "c:\program files\mongodb\server\3.2\bin"**

Ejecuta a continuación el programa "mongo.exe". El resultado es el siguiente:



```
cmd - mongo
C:\Program Files\MongoDB\Server\3.2\bin>mongo
MongoDB shell version: 3.2.3
connecting to: test
>
```

En este punto tenemos ya el servidor de base de datos aceptando peticiones, y el cliente listo para realizárselas.

### Actividad 3: Sentencias CRUD en MongoDB.

A partir de ahora, los ejercicios los realizarás tú. Consulta la documentación de la sesión para conocer los comandos que necesitarás para solucionar cada uno.

Nota importante: MongoDB es sensible al uso de mayúsculas y minúsculas. Cuando crees colecciones y bases de datos en él, asegúrate que usas los nombres tal como los has definido en su creación. Por ejemplo, la colección "**M**ovies" será distinta de la colección "**m**ovies".

### Ejercicio 4: Incorporar a tu servidor la información de la base de datos de ejemplo (CREATE).

**Abstract:** Vamos a crear a una base de datos que llamaremos "dbdm". Le añadiremos una colección de documentos con información sobre las películas de nuestro dataset, así como sus evaluaciones. Esta información la tenemos en unos archivos de texto, con formato Json.

Ejecuta en la pantalla <3> el comando para importar la colección a una base de datos en tu servidor MongoDB. Los parámetros del comando son:

BASEDATOS = dbdm

COLECCIÓN = movies

FICHERO = "c:\datosMongo\movies\_metadata.json" (debes incluir las comillas dobles)

Lo que queremos hacer es cargar en la base de datos "dbdm" una colección llamada "movies" en el servidor activo en nuestra máquina (el que hemos puesto en marcha en la actividad anterior).

Date cuenta de lo siguiente:

1. No hemos tenido que crear la base de datos. El hecho de importar una colección en ella la crea con la configuración por defecto.
2. No le hemos indicado la estructura de los documentos de la colección. La ha tomado directamente del archivo Json. Haciendo un símil con sistemas relacionales, no hemos tenido que ejecutar la instrucción "CREATE TABLE".



## Ejercicio 5: Ejecutando consultas sencillas (READ).

**Abstract:** Vamos a realizar consultas a la colección movies en la base de datos dbdm.

Lanzaremos un par de consultas sencillas a la base de datos. Aunque nuestro servidor ahora tan solo tiene una base de datos (dbdm), podría albergar muchas otras.

1. Ejecuta el comando correspondiente para seleccionar la base de datos dbdm.
2. Vamos a obtener todos los datos de los documentos de la colección movies cuyo lenguaje original sea el español. Ejecuta el comando correspondiente en la pantalla <2>. Los parámetros de la consulta son:

COLECCIÓN = movies  
FILTRO = "original\_language": "es"

El resultado debe ser similar al que muestra la imagen siguiente:

```
cmd - mongo
> use dbdm
switched to db dbdm
> find('movies', {'original_language': 'es'})
{
  "overview" : "Fabian (Dario Grandinetti) works as a bank executive. After one of his colleagues in the bank dies, he falls into crisis. His wife, Mariela (Carolina Peleritti), is a psychologist and has Olga (Mabel Rivera), an embittered woman who tries to take her frustration out on her, as one of her patients. Fabian will meet a famous blind author (Federico Luppi), who tries to help him. Following his advice, Fabian embarks himself on a relationship with Alicia (Antonella Costa), a young artist that he considers his inevitable love.",
  "popularity" : 0.277213,
  "poster_path" : "/zZimQkC6oCRXSzuymxFP4eRl3gR.jpg",
  "production_companies" : "[{'name': 'Adivina Producciones S.L.', 'id': 5470}]",
  "production_countries" : "[{'iso_3166_1': 'AR', 'name': 'Argentina'}, {'iso_3166_1': 'ES', 'name': 'Spain'}]",
  "release_date" : "2014-04-11",
  "revenue" : 0,
  "runtime" : 0,
  "spoken_languages" : "[{'iso_639_1': 'es', 'name': 'Español'}]",
  "status" : "Released",
  "tagline" : "",
  "title" : "Inevitable",
  "video" : "False",
  "vote_average" : 0,
  "vote_count" : 0
}
Type "it" for more
> |
```

La función "find" en MongoDB es equivalente a la sentencia SELECT en SQL. Estudiemos un poco el ejemplo:

1. La sentencia equivalente en SQL sería: `Select * from movies where original_language = 'es'`
2. La función pretty() nos muestra el resultado formateado. Podemos ejecutar la misma sentencia quitando esa función, y nos mostrará todos los caracteres seguidos, siendo muy complicada la interpretación de la información.

3. Los filtros se incluyen con formato JSON dentro del primer parámetro de la función find.
4. Esta sintaxis es muy similar a la de Java o JavaScript.
5. Find devuelve los 20 primeros resultados. Si queremos ver más, ejecutaremos el comando "it".

Vamos a complicar un poco la sentencia de búsqueda, y vamos a seleccionar únicamente el título original de la película (columna original\_title), y la media de votos obtenidos (columna vote\_average). Antes hemos seleccionado todas las columnas. Para ello usaremos la misma función, pero indicándole qué columnas queremos mostrar. Los parámetros de la consulta son:

COLECCIÓN = movies  
FILTRO = "original\_language": "es"  
MOSTRAR = "original\_title":1,"vote\_average":1

El resultado de la consulta debe ser similar al que se muestra en la siguiente imagen:

```
cmd - mongo
C:\> <1> Gestor[*] C:\> <2> cmd - mongo C:\> <3> cmd
Search
" _id" : ObjectId("5e47bbafc1a7d55b46ebadbfb"),
  "original_title" : "El gran amor del conde Drácula",
  "vote_average" : 6.2
}

" _id" : ObjectId("5e47bbafc1a7d55b46ebae04"),
  "original_title" : "Akelarre",
  "vote_average" : 3
}

" _id" : ObjectId("5e47bbafc1a7d55b46ebae18"),
  "original_title" : "El Hombre Sin Rostro",
  "vote_average" : 8
}

" _id" : ObjectId("5e47bbafc1a7d55b46ebae33"),
  "original_title" : "1898. Los últimos de Filipinas",
  "vote_average" : 6.5
}

" _id" : ObjectId("5e47bbafc1a7d55b46ebae34"),
  "original_title" : "Inevitable",
  "vote_average" : 0
}
Type "it" for more
> |
mongo.exe*[64]:6308  « 191012[64] 2/3 [+] NUM InpGrp PRI: 83x26 (3,4172) 25V 9664 100%
```

## Ejercicio 6: Incorporación de información a la base de datos (INSERT).

**Abstract:** Vamos a incorporar nuevos documentos a la colección movies.

Selecciona la pestaña <2> para interactuar con la base de datos.

Inserta en la colección movies el siguiente documento:

DATOS = `_id: 2, "title": "Pelicula DBDM", "belongs_to_collection": "DBDM"`

Consulta la base de datos para asegurarte que se ha insertado ese documento correctamente.

**Nota importante:** el campo "\_id" es una columna especial en MongoDB que hace las funciones de clave primaria automática. Todas las colecciones la tienen de manera obligatoria, y sus valores no se pueden repetir en la misma colección. Si no le damos valor, el gestor de la base de datos le asigna un valor único automático.

## Ejercicio 7: Modificación de información de la base de datos (UPDATE).

**Abstract:** Vamos a modificar información ya existente en la base de datos de películas.

Selecciona la pestaña <2> para interactuar con la base de datos de películas.

Modifica el documento que hemos insertado antes en colección movies con estos nuevos datos:

CONDICIONES = `_id:2`

DATOS = `_id: 2, "title": "Pelicula DBDM (Modificado)"`

**Nota importante:** MongoDB sustituirá el primer documento que encuentre que cumpla las CONDICIONES indicadas por lo que pongamos en DATOS, no únicamente esas columnas. Las columnas que no indiquemos se perderán. En resumen: se sustituye un documento completo por el otro. Para modificar parcialmente el documento, o modificar muchos documentos que cumplan esa condición, tendríamos que hacerlo usando el operador \$set, que no veremos en estas sesiones por ser algo más avanzado.

Consulta la base de datos para asegurarte que se ha cambiado ese documento. Observa si se sigue manteniendo el valor de la columna "belongs\_to\_collection" que hemos insertado en el ejercicio anterior.



## Ejercicio 8: Borrado de información de la base de datos (DELETE).

**Abstract:** Vamos a borrar un documento de la colección movies.

Selecciona la pestaña <2> para interactuar con la base de datos.

Borra el documento que hemos insertado y modificado antes en la colección movies:

CONDICIONES = \_id:2

Consulta la base de datos para asegurarte que ya no existe ese documento.

### Fin de la sesión.

Una vez terminados los ejercicios, deberemos cerrar adecuadamente el servidor. Para ello, ve a la pestaña <2> y ejecuta este comando:

**db.getSiblingDB("admin").shutdownServer()**

Los servidores de bases de datos deben ser desconectados de manera controlada, para que puedan terminar todas las transacciones pendientes, y guardar las cachés.

Hecho esto, cierra las tres sesiones de ConEmu.