



Universidad Tecnológica de Panamá
Facultad de Ingeniería de Sistemas Computacionales
BASE DE DATOS I
Laboratorio N°10



Facilitador: Víctor A. Fuentes T.

Grupo: 1IF-131

Estudiante: Johel Heraclio Batista Cárdenas

Cédula: 8-914-587

A. TÍTULO DE LA EXPERIENCIA:

Laboratorio No.10. Bases de Datos No Relacionales (NoSQL)

B. TEMAS:

- a. Instalación de MongoDB
- b. Creación de base de datos en modo consola
- c. Opciones de colecciones, métodos y opciones de lectura / escritura
- d. Uso de MongoDB Compass

C. OBJETIVO(S):

- Instalar y manejar la estructura básica de una base de datos documental MongoDB y sus comandos básicos desde el manejo de consola (Shell) y en el modo gráfico con Compass.

D. METODOLOGÍA:

Para presentar el informe de los resultados obtenidos, haga captura de pantalla desde mostrando la instrucción y el resultado de la consulta generada por la misma.

Copie estas capturas de pantalla en la sección G (RESULTADOS) de esta guía, en el número mostrado en la sección E (PROCEDIMIENTO). Corte y sólo presente el área de trabajo donde aparece la instrucción y el resultado obtenido.

Recuerde colocar el texto de las sentencias que está utilizando en sus resultados.

E. PROCEDIMIENTO O ENUNCIADO DE LA EXPERIENCIA:

Todo lo indicado en color verde corresponde a acciones que usted deberá ejecutar. La primera sección es una serie de ejemplos explicativos que debe realizar para familiarizarse con el uso de las funciones y sintaxis que se tratan en este tema.

Base de Datos NoSQL

Las bases de datos no relaciones o NoSQL surgen de la necesidad de gestionar grandes cantidades de información y se distinguen porque no cumplen con el tradicional esquema entidad-relación.

Son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor similar a tablas Hash, Mapeo de Columnas, Documentos o Grafos.

Las bases de datos NoSQL se puede definir como “un conjunto de conceptos que permite el procesamiento rápido y eficiente de conjuntos de datos con un enfoque en el rendimiento, la confiabilidad y la agilidad”

Las bases de datos basadas en documentos se refieren a aquellas que almacenan sus datos en forma de documentos y dentro de estos documentos, posibilitando anidar otros documentos relacionados, como si se tratase de carpetas físicas (Pollo Cattaneo, López Nocera, & Daián Rottoli, 2014).

Las bases de datos basadas en documentos ofrecen un gran rendimiento y opciones de escalabilidad horizontal y son similares a los registros en bases de datos relacionales, pero son mucho más flexibles, ya que no tienen esquema (Ameya Nayak, 2013).

A diferencia de las bases de datos tipo llave-valor, estos sistemas usualmente admiten índices secundarios y múltiples tipos de documentos (objetos) por base de datos, y documentos o listas anidados (Cattell, 2011).

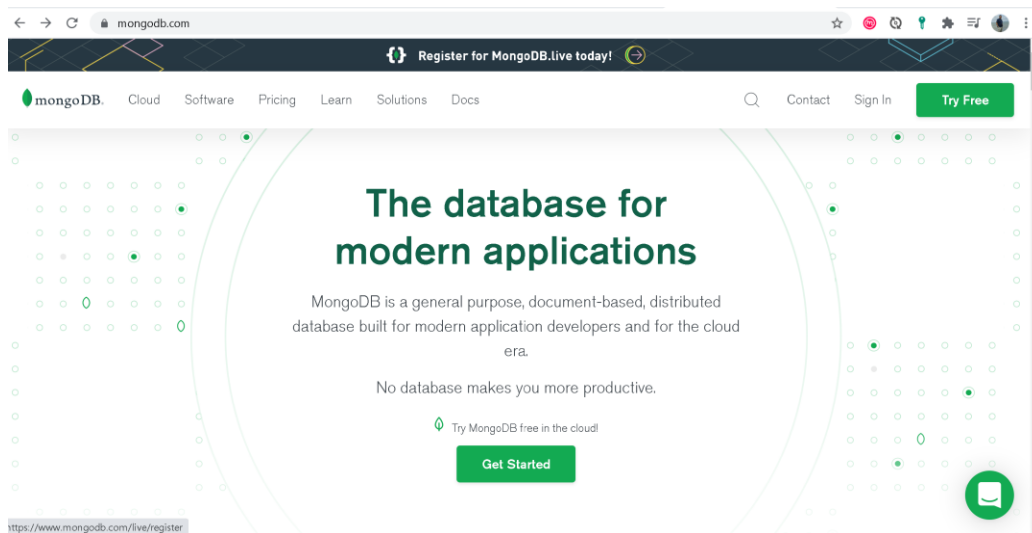
Algunos de los ejemplos más destacados de este tipo de base de datos son: MongoDB, CouchDB, SimpleDB, entre otros.

En el desarrollo de este laboratorio se exploran las bases de datos documentales, en particular el sistema MongoDB, el cual es uno de los de mayor uso comercial en la actualidad, pero no es el único.

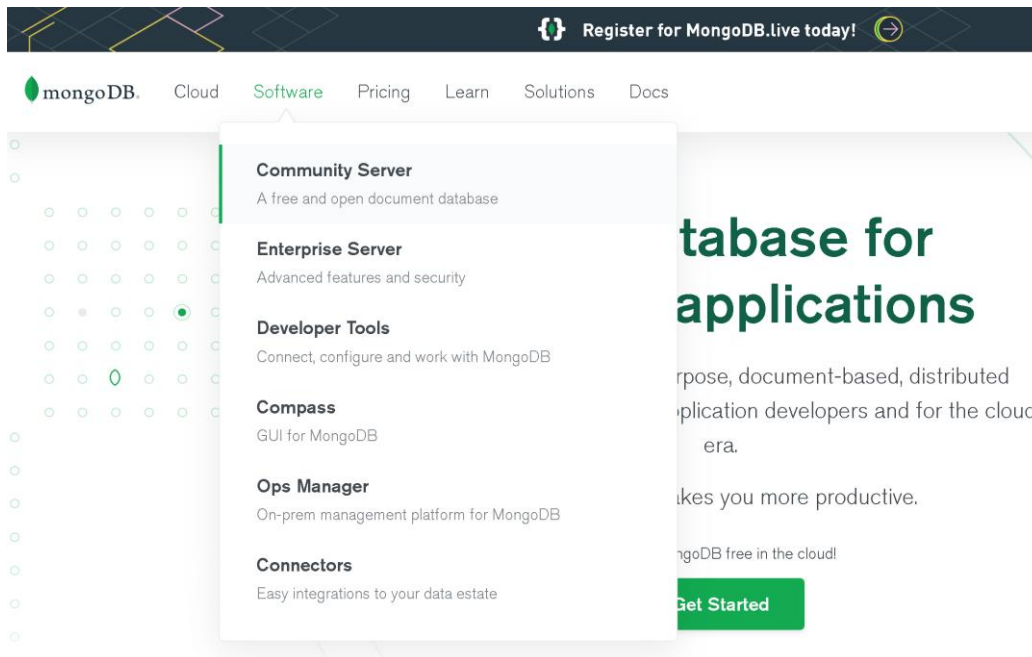


mongoDB®

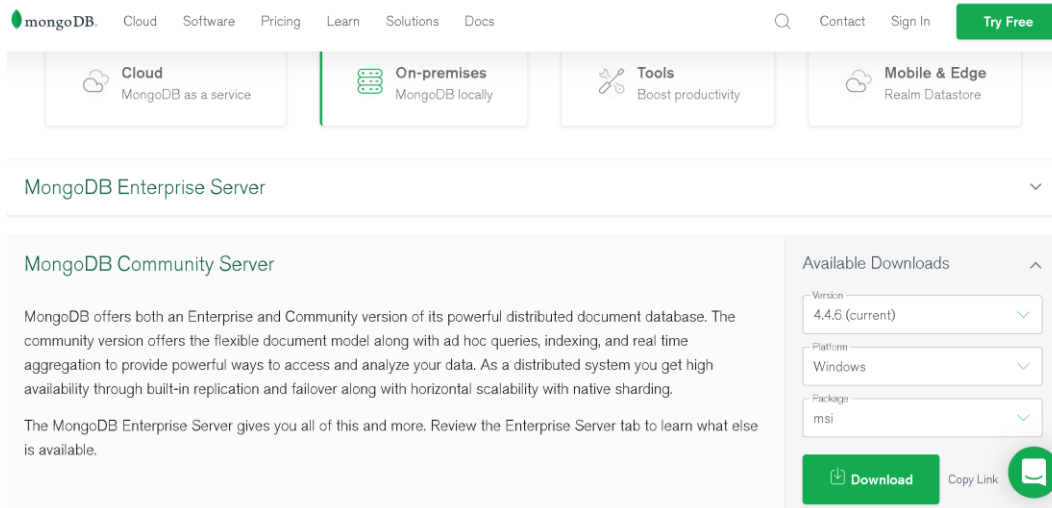
Ingresa a al sitio oficial de MongoDB <https://www.mongodb.com/>



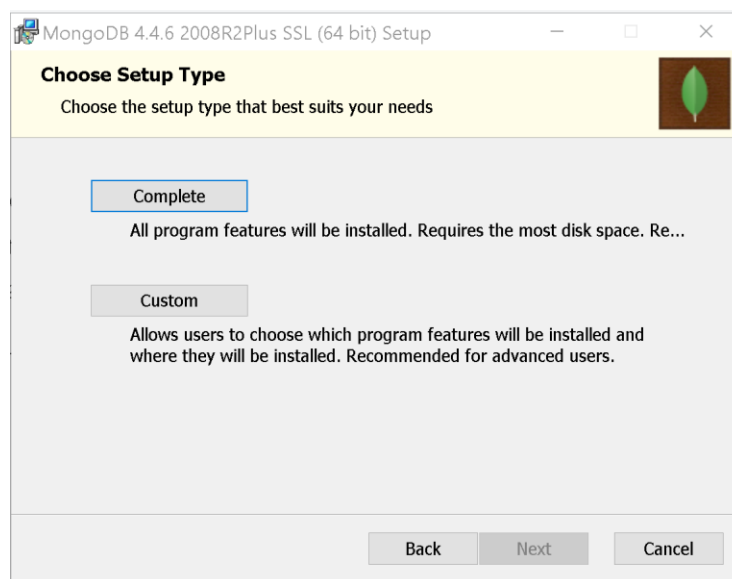
Ingresa a la sección de Software y escoger la opción de Community Server



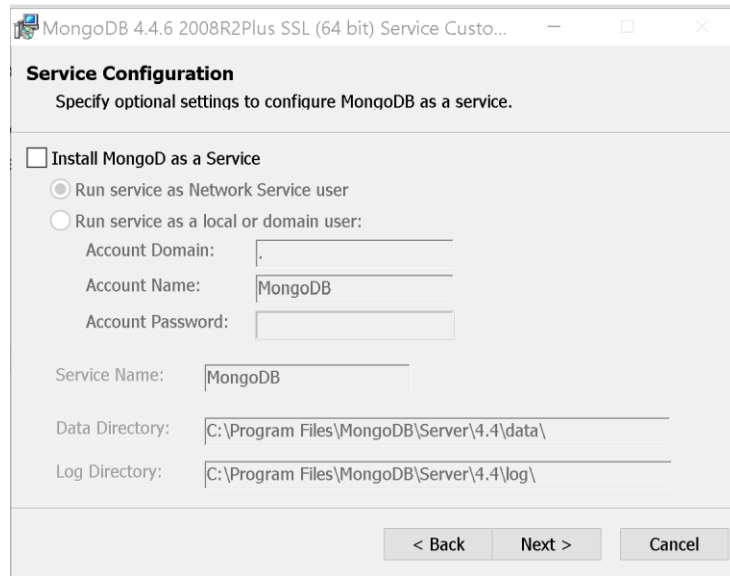
En la opción que se presenta del lado derecho de la pantalla, escoge el sistema operativo sobre el cual vas a realizar el proceso de instalación de la herramienta:



Cuando ha finalizado el proceso de descarga del instalador, ejecuta la aplicación para proceder a instalar MongoDB en tu computadora. En el proceso de instalación, basta con que des siguientes en las diferentes opciones del wizard. En la sección donde pregunta el tipo de instalación, se recomienda la versión completa:



Cuando se continúa con el proceso, existe una opción que pregunta si deseas instalar la plataforma como un servicio. Si activas esta opción, cada vez que inicies tu sistema operativo se ejecutarán los servicios de MongoDB. Si no deseas que eso ocurra basta con que desmarques la opción tal y como se presenta en la siguiente pantalla:



Al seleccionar la opción de instalación completa, también se instala MongoDB Compass, la cual es la interfaz gráfica de usuario de la plataforma. Asegúrese de tenerlo marcado:



Una vez se complete el proceso de instalación de MongoDB, aparecerá en pantalla una serie de acuerdos de licenciamiento que debe aceptar, dar siguiente en las opciones de visualización y se recomienda activar las actualizaciones de la plataforma con todas sus aplicaciones.

Privacy Settings

To enhance the user experience, Compass can integrate with 3rd party services, which requires external network requests. Please choose from the settings below:

- ☒ **Enable Product Feedback Tool**
Enables a tool for sending feedback or talking to our Product and Development teams directly from Compass.
- ☒ **Enable Geographic Visualizations**
Allow Compass to make requests to a 3rd party mapping service.
- ☒ **Enable Crash Reports**
Allow Compass to send crash reports containing stack traces and unhandled exceptions.
- ☒ **Enable Usage Statistics**
Allow Compass to send anonymous usage statistics.
- ☒ **Enable Automatic Updates**
Allow Compass to periodically check for new updates.

With any of these options, none of your personal information or stored data will be submitted.
Learn more:[MongoDB Privacy Policy](#)

Start Using Compass

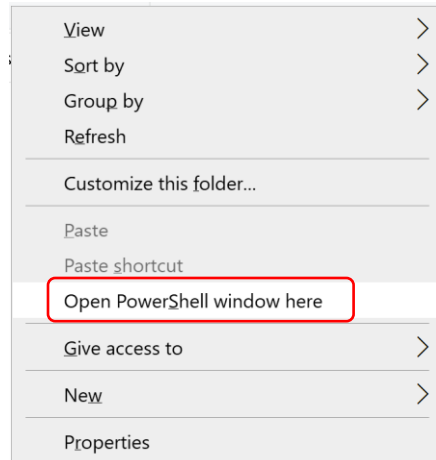
Ahora puedes cerrar la ventana anterior y dar clic en la opción de Finalizar del wizard de instalación de la aplicación.

El siguiente paso es comprobar que el proceso de instalación fue correcto. Para ello se necesita abrir una consola de Windows si están en dicho sistema operativo, o una terminal dependiendo del sistema que esté usando. Presiona el botón de Windows + r para que aparezca, y luego escriba el comando cmd. Cuando tenga la consola abierta, escriba `mongod --version` y luego presiona enter:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Victor>mongod --version
```

Si al ejecutar el comando, le aparece que no se reconoce `mongod`, es necesario ir a la carpeta donde se guardó la aplicación. Usualmente debe ingresar al disco duro, archivos de programa, escoger MongoDB y navegar hasta la carpeta bin, donde estarán las aplicaciones de la plataforma. Una vez allí, presiona la tecla shift y clic derecho y se debe abrir la siguiente ventana en la cual escoges la opción Abrir la ventana de Power Shell aquí:



Una vez abierta la terminal de Windows en la carpeta correspondiente, utiliza el siguiente comando para validar la versión de MongoDB: `./mongod.exe --version` y debe aparecer la versión que tiene instalada:

```
Windows PowerShell
PS C:\Program Files\MongoDB\Server\4.4\bin> ./mongod.exe --version
db version v4.4.6
Build Info: {
  "version": "4.4.6",
  "gitVersion": "72e66213c2c3eab37d9358d5e78ad7f5c1d0d0d7",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```



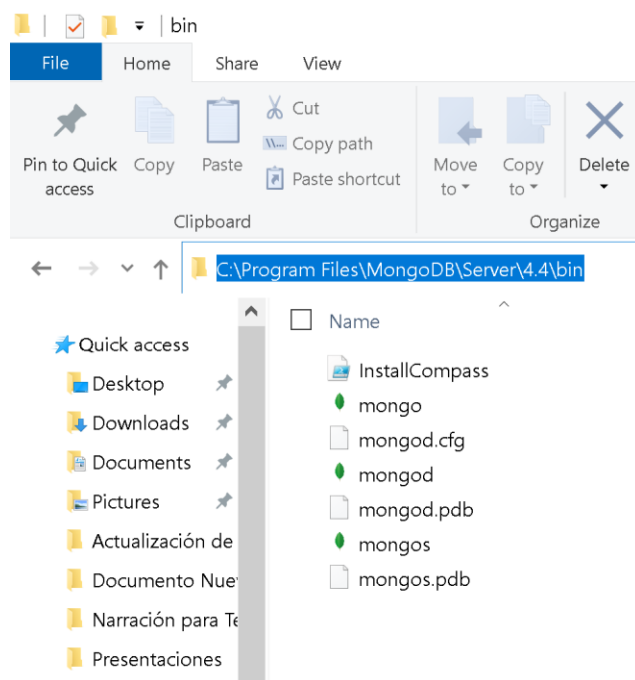
```
PowerShell 7.2.5
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

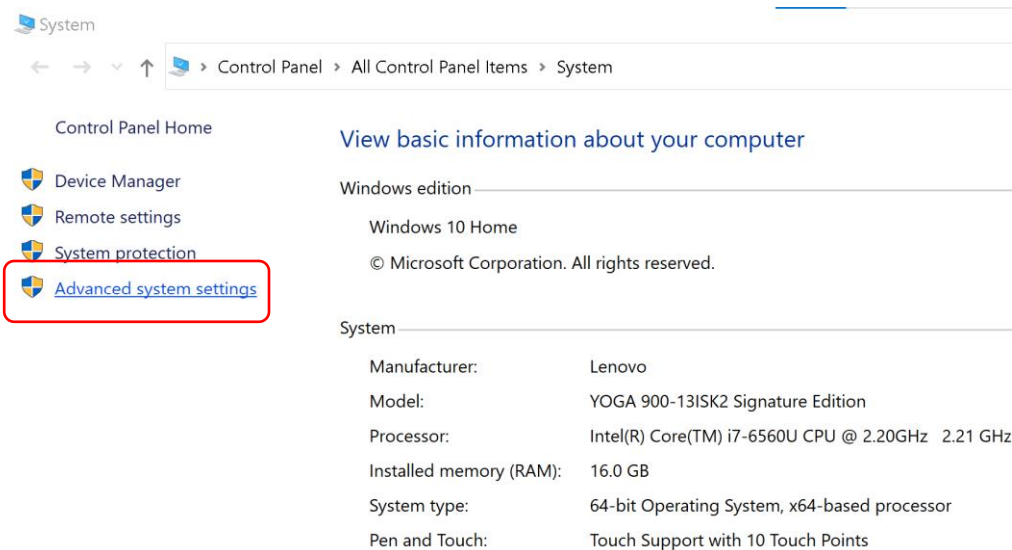
PS C:\Program Files\MongoDB\Server\5.0\bin> ./mongod.exe --version
db version v5.0.9
Build Info: {
  "version": "5.0.9",
  "gitVersion": "6f7dae919422dcd7f4892c10ff20cdc721ad00e6",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

Evidencia de Ejecución en PoweShell

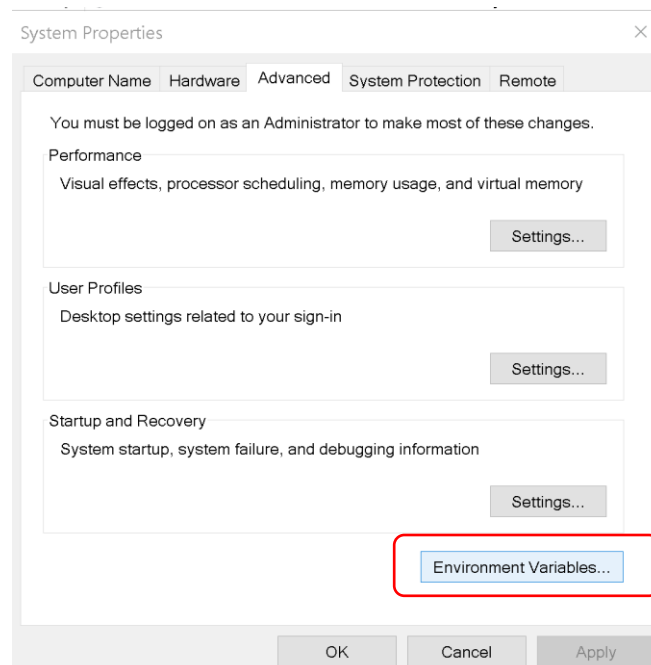
Para ejecutar MondoDB, deberá acceder siempre a la carpeta donde se tiene guardado el programa, la cual ya visitó, sin embargo, se puede configurar el path del sistema operativo para garantizar el acceso directo a MongoDB sin la necesidad de ingresar de la forma anterior. Para ello, copie en primer lugar la ruta de la carpeta que necesita y en donde se guardó la aplicación. Para nuestro caso particular es la siguiente ruta: C:\Program Files\MongoDB\Server\4.4\bin como se muestra en la siguiente imagen:



Ahora utilice el explorador de archivos y vaya a Mi Equipo y con el botón derecho escoja propiedades. Una vez dentro de la ventana respectiva, del lado izquierdo superior, escoja Opciones Avanzadas del Sistema:

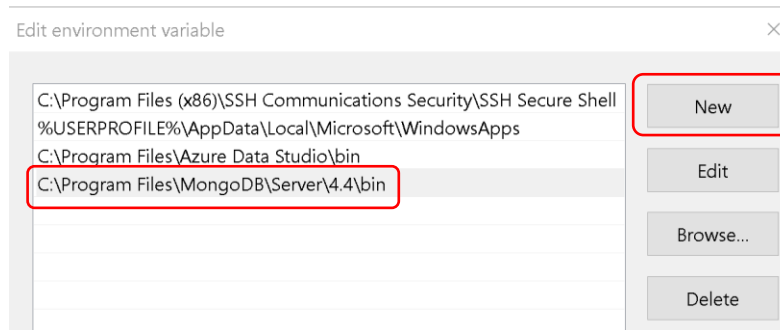


En la ventana que aparece, escoja la opción Variables del Entorno:



Ahora escoja las variables del usuario y en la opción Path, dé un doble clic para que aparezca la ventana de rutas. Dé un clic en Nuevo y pegue en este lugar la ruta de acceso a la carpeta donde

está MongoDB. Luego dé clic en aceptar (OK) en todas las ventanas y ya se habrá creado el acceso.

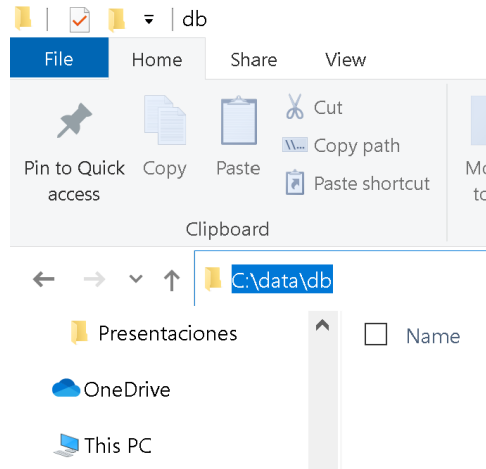


Puede comprobar que se tiene el acceso abriendo una consola (Botón Windows + r) y ejecute el comando `mongod --version`

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Victor>mongod --version
db version v4.4.6
Build Info: {
  "version": "4.4.6",
  "gitVersion": "72e66213c2c3eab37d9358d5e78ad7f5c1d0d0d7",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

Para que el servicio de MongoDB se mantenga en ejecución una vez sea llamado, se deben crear dos carpetas. La primera la debe crear en el directorio C:/ la cual debe llamarse “data” y dentro de esta carpeta creará otra carpeta llamada “db”. De esta forma se accederá a los servicios y la base de datos de forma correcta quedando la ruta de esta forma:



CREACIÓN DE UNA BASE DE DATOS EN MONGODB

Para crear la primera base de datos en MongoDB abra una consola de Windows (⌨+R), ejecute el cmd y especifique con el botón derecho que lo va a ejecutar como administrador.

Luego, usa el comando mongod. En el resultado se debe verificar que se esté escuchando el servidor y esperando por conexiones en el puerto 27017:

```
C:\WINDOWS\system32\cmd.exe - mongod
:936953][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 5 through 5"}
{"t":{"sdate":"2021-07-04T10:43:22.036-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:36606][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Main recovery loop: starting at 4/8960 to 5/256"}}
{"t":{"sdate":"2021-07-04T10:43:22.183-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:182610][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 4 through 5"}
{"t":{"sdate":"2021-07-04T10:43:22.283-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:282892][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 5 through 5"}
{"t":{"sdate":"2021-07-04T10:43:22.378-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:377313][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] set global recovery timestamp: (0, 0)}
{"t":{"sdate":"2021-07-04T10:43:22.378-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:377313][14836:140718482543952], txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] set global oldest timestamp: (0, 0)}
{"t":{"sdate":"2021-07-04T10:43:22.382-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message","attr":{"message":"[1625413402
:381773][14836:140718482543952], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 1, snapshot max: 1 snapshot count: 0, oldest tim
estamp: (0, 0), meta checkpoint timestamp: (0, 0)}
{"t":{"sdate":"2021-07-04T10:43:22.408-05:00"},"s":"I", "c":"RECOVERY", "id":23987, "ctx":"initandlisten","msg":"WiredTiger recoveryTimestamp","attr":{"recoveryTim
estamp":{"timestamp":{"t":"0","i":"0"}}}}
{"t":{"sdate":"2021-07-04T10:43:22.412-05:00"},"s":"I", "c":"STORAGE", "id":4366408, "ctx":"initandlisten","msg":"No table logging settings modifications are requir
ed for existing WiredTiger tables","attr":{"loggingEnabled":true}}
{"t":{"sdate":"2021-07-04T10:43:22.415-05:00"},"s":"I", "c":"STORAGE", "id":22262, "ctx":"initandlisten","msg":"Timestamp monitor starting"}
{"t":{"sdate":"2021-07-04T10:43:22.432-05:00"},"s":"H", "c":"CONTROL", "id":22120, "ctx":"initandlisten","msg":"Access control is not enabled for the database. Rea
d and write access to data and configuration is unrestricted","tags":{"startupWarnings"}}
{"t":{"sdate":"2021-07-04T10:43:22.432-05:00"},"s":"H", "c":"CONTROL", "id":22140, "ctx":"initandlisten","msg":"This server is bound to localhost. Remote systems
will be unable to connect to this server. Start the server with --bind_ip address to specify which IP addresses it should serve responses from, or with --bind_ip al
l to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning","tags":{"startupWarnings"}}
{"t":{"sdate":"2021-07-04T10:43:22.441-05:00"},"s":"I", "c":"STORAGE", "id":20536, "ctx":"initandlisten","msg":"Flow Control is enabled on this deployment"}
{"t":{"sdate":"2021-07-04T10:43:22.767-05:00"},"s":"I", "c":"FTDC", "id":20625, "ctx":"initandlisten","msg":"Initializing full-time diagnostic data capture","a
tt":{"dataDirectory":"C:/data/db/diagnostic.data"}}
{"t":{"sdate":"2021-07-04T10:43:22.782-05:00"},"s":"I", "c":"NETWORK", "id":23015, "ctx":"listener","msg":"listening on","attr":{"address":"127.0.0.1"}}
{"t":{"sdate":"2021-07-04T10:43:22.783-05:00"},"s":"I", "c":"NETWORK", "id":23016, "ctx":"listener","msg":"Waiting for connections","attr":{"port":27017,"ssl":"of
f"}}
{"t":{"sdate":"2021-07-04T10:44:22.419-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"WTCheckpointThread","msg":"WiredTiger message","attr":{"message":"[16254
13462:419248][14836:140718482543952], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 3, snapshot max: 3 snapshot count: 0, oldest
t timestamp: (0, 0), meta checkpoint timestamp: (0, 0)}
{"t":{"sdate":"2021-07-04T10:45:22.454-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"WTCheckpointThread","msg":"WiredTiger message","attr":{"message":"[16254
13522:453097][14836:140718482543952], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 5, snapshot max: 5 snapshot count: 0, oldest
t timestamp: (0, 0), meta checkpoint timestamp: (0, 0)}
{"t":{"sdate":"2021-07-04T10:46:22.493-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"WTCheckpointThread","msg":"WiredTiger message","attr":{"message":"[16254
13582:493245][14836:140718482543952], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 6, snapshot max: 6 snapshot count: 0, oldest
t timestamp: (0, 0), meta checkpoint timestamp: (0, 0)}}
```

Proceda a abrir otra consola como administrador y ejecute el comando mongo. La respuesta debe incluir al final el símbolo >, lo cual significa que el servicio está listo y en espera de instrucciones:

```
C:\WINDOWS\system32\cmd.exe - mongo
C:\Users\Victor>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("54a7745d-e9ad-4200-ae81-5dbcac74a23c") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
  2021-07-04T10:43:22.432-05:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
  2021-07-04T10:43:22.432-05:00: This server is bound to localhost. Remote systems will be unable to connect to th
is server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or wi
th --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to di
sable this warning
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

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

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

Puedes usar el comando db para verificar las bases de datos en la que está trabajando actualmente. Si es la primera vez que se está usando la aplicación, la base de datos que aparecerá creada se denomina test y es la BD a la que se conecta la aplicación por defecto.

```
---
> db
test
>
```

Recuerde que Mongo DB utiliza una arquitectura funcional que se ejemplifica de la siguiente manera:

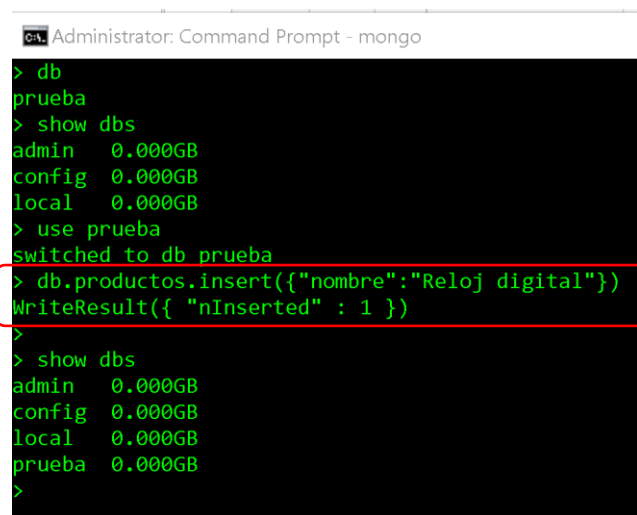


Si desea listar las bases de datos que hay instaladas en el sistema, puede usar el comando show dbs, tal como se muestra a continuación y que son las bases de datos por defecto:

```
C:\WINDOWS\system
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
>
```

Para crear la primera base de datos, utilice el comando “use” seguido del nombre de la base de datos. Para este caso se ha creado la base de datos prueba (use prueba). Inmediatamente, MongoDB lo posiciona en la base de datos que se creó, sin embargo, para que exista la base de datos se debe insertar al menos un elemento y lo pueda listar con el comando show dbs (Note que, si en este momento utiliza este comando, no mostrará la base de datos prueba).

Se procede a insertar un dato en la base de datos. Recuerde que MongoDB es una base de datos documental y por lo tanto debo crear una colección para luego trabajar con el documento. En este ejemplo, tenemos la base de datos prueba, se crea una colección que se llamará “productos” y luego se inserta en productos el documento respectivo con el valor que se desea. La colección entonces es productos y el documento tendrá el dato nombre. Para ello se usa la siguiente estructura:

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt - mongo". The window has a black background with green text. The commands and output are as follows:

```
> db
prueba
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use prueba
switched to db prueba
> db.productos.insert({"nombre":"Reloj digital"})
WriteResult({ "nInserted" : 1 })
>
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
prueba   0.000GB
>
```

The line containing the insert command and its output is highlighted with a red rectangular box.

El programa le muestra el resultado de este proceso en la segunda línea que aparece señalada en rojo. Si deseas ver las colecciones de la base de datos en la que se encuentra, use el comando show collections.

También se pueden usar métodos para la creación de colecciones, para borrarlas y cargar datos. Algunas de estas opciones se muestran en el siguiente código:

```

Administrator: Command Prompt - mongo
>
> show collections
productos
>
> db.createCollection("clientes")
{ "ok" : 1 }
> db.createCollection("vendedores")
{ "ok" : 1 }
> db.createCollection("departamentos")
{ "ok" : 1 }
> db.createCollection("pedidos")
{ "ok" : 1 }
>
> show collections
clientes
departamentos
pedidos
productos
vendedores
>
> db.pedidos.drop()
true
>
> show collections
clientes
departamentos
productos
vendedores
>

```

Se están creando nuevas colecciones usando la instrucción `db.createCollection("nombre")`

Se muestran las colecciones creadas con `show collections`

Use la estructura `db.nombrecolección.drop()` para borrar. Asegúrese que está en la base de datos correcta.

Hasta este momento, las colecciones que se han creado están vacías, a excepción de la colección `productos` que tiene un elemento. Para visualizar este contenido, utilice `db.productos.find()` y mostrará su contenido. Note que el resultado incluye un id que ha sido creado por MongoDB.

```

> db.productos.find()
{ "_id" : ObjectId("60e1f216aac412083d89275b"), "nombre" : "Reloj digital" }
>

```

A continuación, se procede a crear un documento nuevo en la colección `productos`, con la diferencia que se usa un elemento indexado, es decir un documento dentro de otro. Observe la siguiente estructura:

```

(
{
  "nombre": "Tablet",
  "precio": 119.95,
  "marca": "Samsung",
  "color": "gris",
  "garantía": true,
  "detalles": {

```

```

    "pais": "Korea del Sur",
    "ciudad": "Seúl",
    "fecha_producción": new Date("05/10/2021")
  }
}
)

```

```

> db.productos.insert(
... {
...   "nombre": "Tablet",
...   "precio": 119.95,
...   "marca": "Samsung",
...   "color": "gris",
...   "garantia": true,
...   "detalles": {
...     "pais": "Korea del Sur",
...     "ciudad": "Seúl",
...     "fecha_producción": new Date("05/10/2021")
...   }
... }
... )
WriteResult({ "nInserted" : 1 })
>

```

Para visualizar el contenido del documento, use la sintaxis

Si desea visualizar el documento en el estilo de JavaScript Object Notation (JSON) puede agregar un método adicional a la sintaxis: `db.productos.find()`

```

> db.productos.find()
{ "_id" : ObjectId("60e1f216aac412083d89275b"), "nombre" : "Reloj digital" }
{ "_id" : ObjectId("60e20541aac412083d89275c"), "nombre" : "Tablet", "precio" : 119.95, "marca" : "Samsung", "color" :
gris", "garantia" : true, "detalles" : { "pais" : "Korea del Sur", "ciudad" : "Seúl", "fecha_producción" : ISODate("202
-05-10T05:00:00Z") } }
>

```

Existe un método adicional que muestra la información en el formato JSON que se usó para crear el documento. El método se agrega luego del find y queda de la siguiente forma:

`db.productos.find().pretty()`

```

> db.productos.find().pretty()
{ "_id" : ObjectId("60e1f216aac412083d89275b"), "nombre" : "Reloj digital" }
{
  "_id" : ObjectId("60e20541aac412083d89275c"),
  "nombre" : "Tablet",
  "precio" : 119.95,
  "marca" : "Samsung",
  "color" : "gris",
  "garantia" : true,
  "detalles" : {
    "pais" : "Korea del Sur",
    "ciudad" : "Seúl",
    "fecha_producción" : ISODate("2021-05-10T05:00:00Z")
  }
}
>

```


Hasta el momento se ha visto que pueden insertarse datos de forma independiente o por separado, manejando diferente estructura o esquema. Esta es una de las características que resaltan en base de datos documentales. Una de las opciones que también se tiene para insertar datos, es la posibilidad de insertarlos mediante listas, lo cual serían varios documentos a la vez. Para ello deben usarse los símbolos de [] y dentro de ellos se tienen los documentos delimitados por { } y separados por coma de la siguiente forma:

```
db.productos.insert ([
  {
    "nombre": "Audífonos",
    "precio": 49.95,
    "marca": "Aiwa",
    "color": "negro",
    "garantia": true,
    "detalles": {
      "pais": "Japón",
      "ciudad": "Nagasaki",
      "fecha_producción": new Date("12/23/2020")
    }
  },
  {
    "nombre": "Monitor",
    "precio": 149.85,
    "marca": "Dell",
    "color": "negro",
    "garantia": true
  },
  {
    "nombre": "Celular",
    "precio": 459.95,
    "marca": "LG",
    "color": "negro",
    "conservación": "reconstruido",
    "garantia": false,
    "detalles": {
      "pais": "China"
    }
  }
])
```

Una vez ejecutada la inserción, debe aparecer el siguiente mensaje de éxito, señalando lo realizado:

```
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 3,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>
```

Liste los documentos de la colección productos (muestre la captura de pantalla respectiva).

Inserte una cantidad de documentos en las colecciones de su base de datos y use diferentes esquemas. La cantidad señalada para cada colección aparece en paréntesis:

- clientes (4)
- departamentos (3)
- vendedores (2)

Búsqueda de valores en MongoDB

Al igual que el modelo relacional, este tipo de aplicaciones también brinda una serie de métodos que permiten listar o extraer registros, en estos casos documentos o secciones. Existen diferentes formas de combinar métodos para conseguir la información necesaria.

Para el primer criterio de búsqueda lo que haremos será listar todos los documentos de la colección productos que tengan la información de color en negro, usando el siguiente comando: `db.productos.find({"color":"negro"})`. Note que se utilizan las llaves dentro del método find.

La salida que se obtiene es la siguiente. Recuerde que puedes visualizar de una mejor forma con el método pretty.

```
Administrator: Command Prompt - mongo
> db.productos.find({"color":"negro"}).pretty()
{
  "_id" : ObjectId("60e23efbaac412083d89275d"),
  "nombre" : "Audifonos",
  "precio" : 49.95,
  "marca" : "Aiwa",
  "color" : "negro",
  "garantia" : true,
  "detalles" : {
    "pais" : "Japón",
    "ciudad" : "Nagasaki",
    "fecha_producción" : ISODate("2020-12-23T05:00:00Z")
  }
}
{
  "_id" : ObjectId("60e23efbaac412083d89275e"),
  "nombre" : "Monitor",
  "precio" : 149.85,
  "marca" : "Dell",
  "color" : "negro",
  "garantia" : true
}
{
  "_id" : ObjectId("60e23efbaac412083d89275f"),
  "nombre" : "Celular",
  "precio" : 459.95,
  "marca" : "LG",
  "color" : "negro",
  "onservación" : "reconstruido",
  "garantia" : false,
  "detalles" : {
    "pais" : "China"
  }
}
>
```

Puedes agregar un criterio adicional dentro del método. Por ejemplo, si desea listas aquellos productos que sean de color negro y cuya marca sea Aiwa, se usaría lo siguiente:

```
db.productos.find({"color":"negro","marca":"Aiwa"}).pretty()
```

Ordenamiento

Para ordenar de forma alfabética la salida obtenida, basta con se utilice el método sort y se establezca el criterio de ordenamiento en verdadero, usando el valor de 1 tal y como se muestra en la siguiente sintaxis:

```
db.productos.find({"color":"negro"}).pretty().sort({nombre:1})
```

Observe que la salida obtenida está ordenada atendiendo al nombre del producto:

```

Administrator: Command Prompt - mongo
> db.productos.find({"color":"negro"}).pretty().sort({nombre:1})
{
  "_id" : ObjectId("60e23efbaac412083d89275d"),
  "nombre" : "Audifonos",
  "precio" : 49.95,
  "marca" : "Aiwa",
  "color" : "negro",
  "garantia" : true,
  "detalles" : {
    "pais" : "Japón",
    "ciudad" : "Nagasaki",
    "fecha_producción" : ISODate("2020-12-23T05:00:00Z")
  }
}
{
  "_id" : ObjectId("60e23efbaac412083d89275f"),
  "nombre" : "Celular",
  "precio" : 459.95,
  "marca" : "LG",
  "color" : "negro",
  "onservación" : "reconstruido",
  "garantia" : false,
  "detalles" : {
    "pais" : "China"
  }
}
{
  "_id" : ObjectId("60e23efbaac412083d89275e"),
  "nombre" : "Monitor",
  "precio" : 149.85,
  "marca" : "Dell",
  "color" : "negro",
  "garantia" : true
}
>

```

Existen algunos métodos que le permiten obtener los primeros documentos de una colección, se puede contar la cantidad de documentos usando un count a la colección, lo cual es equivalente a lo que se utilizó en algunos laboratorios usando el modelo relacional.

Se pueden usar opciones de concatenación para mostrar parte de la información de los documentos tal y como se muestra en la siguiente sentencia:

```
db.productos.find().forEach(productos => print("Nombre del Producto: " + productos.nombre))
```

Se usa el método forEach (para cada elemento) en conjunto con la salida en la consola, donde se concatena una cadena con los resultados de la consulta:

```

> db.productos.find().forEach(productos => print("Nombre del Producto: " + productos.nombre))
Nombre del Producto: Reloj digital
Nombre del Producto: Tablet
Nombre del Producto: Audifonos
Nombre del Producto: Monitor
Nombre del Producto: Celular
>

```

Actualización de datos

Para actualizar los valores de un documento se puede utilizar el método update, sin embargo, es importante saber que existen diferentes formas de realizar la actualización deseada.

Si se usa el método update con un criterio de búsqueda agregando lo que quiere actualizar, el método va a borrar todo el contenido del documento y dejará dicho documento solo con lo que ha indicado en el update.

Por ejemplo, si se busca actualizar un producto cuyo precio sea 49.95 y se lo coloca el nuevo precio en 59.95, toda la demás información que tenía ese documento se elimina y solo quedará el campo precio. Para evitar este inconveniente, se agrega la cláusula \$set en el precio. Observe el siguiente ejemplo:

```
db.productos.update({"nombre":"Audifonos"}, {$set:{"precio":59.95}})
```

```
> db.productos.update({"nombre":"Audifonos"}, {$set:{"precio":59.95}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.productos.find({"nombre":"Audifonos"})
{ "_id" : ObjectId("60e23efbaac412083d89275d"), "nombre" : "Audifonos", "precio" : 59.95, "marca" : "Aiwa", "color" : "negro", "garantia" : true, "detalles" : { "pais" : "Japón", "ciudad" : "Nagasaki", "fecha_producción" : ISODate("2020-12-23T05:00:00Z") } }
> db.productos.find({"nombre":"Audifonos"}).pretty()
{
  "_id" : ObjectId("60e23efbaac412083d89275d"),
  "nombre" : "Audifonos",
  "precio" : 59.95,
  "marca" : "Aiwa",
  "color" : "negro",
  "garantia" : true,
  "detalles" : {
    "pais" : "Japón",
    "ciudad" : "Nagasaki",
    "fecha_producción" : ISODate("2020-12-23T05:00:00Z")
  }
}
```

También es posible que la actualización permita crear un nuevo documento en la colección, por lo cual MongoDB los considera una actualización. Para ello se utiliza el método update. A continuación, se agrega un documento en la colección productos:

```
db.productos.update({"nombre":"Bocinas"}, {$set: {"precio":39.49}}, {upsert:true})
```

En este caso, el método upsert permite que se inserte el documento en la colección aunque no exista:

```
> db.productos.update({"nombre":"Bocinas"}, {$set: {"precio":39.49}}, {upsert:true})
WriteResult({
  "nMatched" : 0,
  "nUpserted" : 1,
  "nModified" : 0,
  "_id" : ObjectId("60e260c7b40753c32ceb3c06")
})
>
```

Otro de los métodos que puede serle útil al momento de estar trabajando las actualizaciones es renombrar un campo o atributo.

Para este ejemplo particular, use el método find y pretty para darse cuenta de que uno de los campos está más definido en cuanto a nombre. Hay un documento en donde el campo observación está mal escrito (se escribió onservación).

Para realizar la corrección del nombre del campo se utiliza el método \$rename de la siguiente forma:

```
db.productos.update({"nombre":"Celular"}, {$rename: {"onservación" : "observación"}})
```

```
> db.productos.update({"nombre":"Celular"}, {$rename: {"onservación" : "observación"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Note que MongoDB le indica que uno de los documentos ha sido modificado.

Muestre cómo queda el documento respectivo luego de esta modificación usando el método find y pretty.

Actualice uno de los documentos de cualquier colección en su base de datos.

Eliminar Documentos

Para proceder a eliminar documentos de las colecciones se usa el método remove. Se debe especificar la colección y un criterio de búsqueda.

A continuación, se procede a eliminar el documento cuyo nombre es Reloj Digital. Utilice el siguiente formato:

```
db.productos.remove({"nombre" : "Reloj digital"})
```

```
> db.productos.remove({"nombre" : "Reloj digital"})
WriteResult({ "nRemoved" : 1 })
>
```

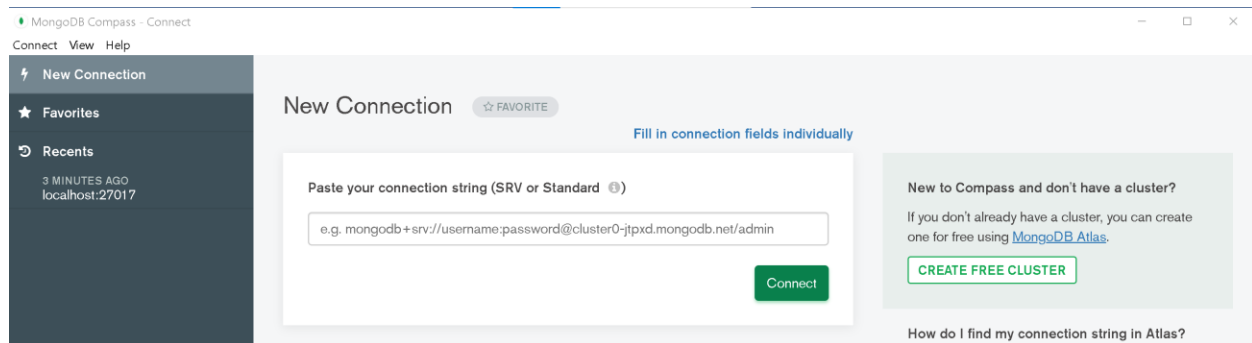
Si necesita eliminar todos los documentos de una colección, no es necesario enviar parámetros en el método remove, para lo cual puede usarlo de la siguiente forma:

`db.productos.remove({ })` → **si lo ejecuta, se eliminarán todos sus documentos de la colección.**

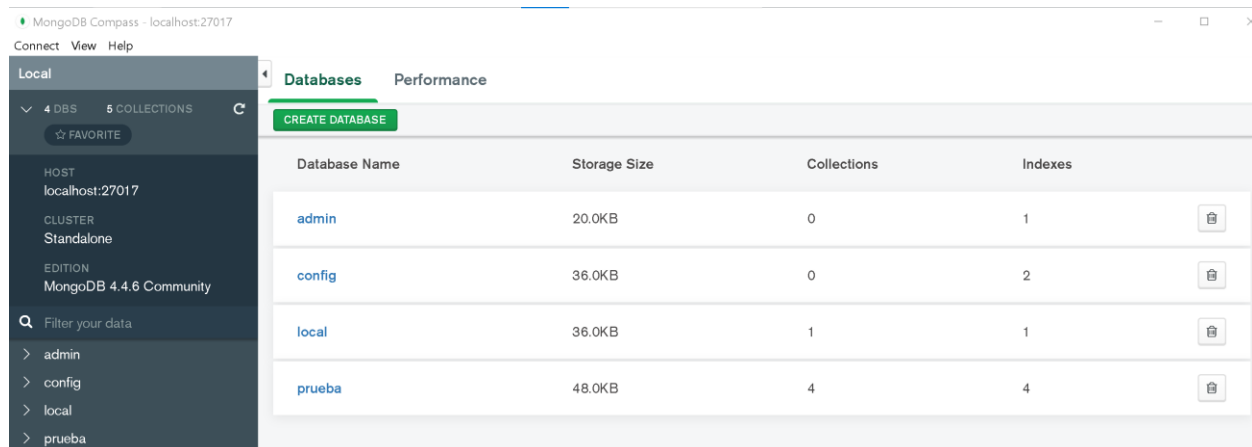
Usando el modo gráfico en MongoDB Compass

Compass es el modo gráfico de Mongo. Para ingresar a la aplicación busque el acceso directo creado o en su lista de programas.

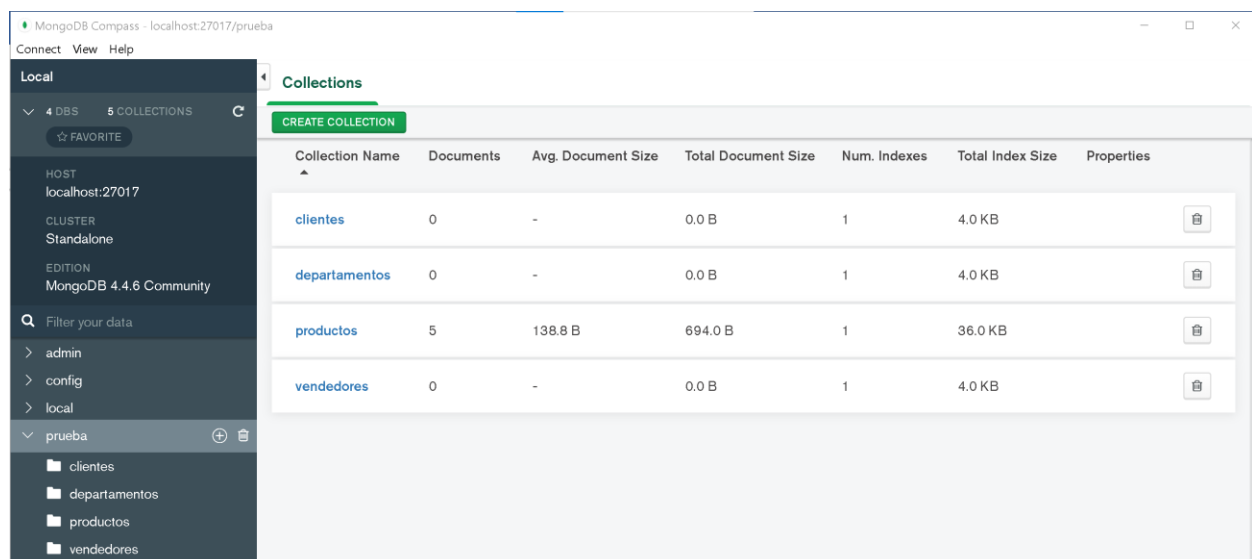
La ventana iniciar permite establecer una conexión con el servidor para lo cual debe usar la opción Connect en la ventana New Connection tal y como se muestra a continuación:



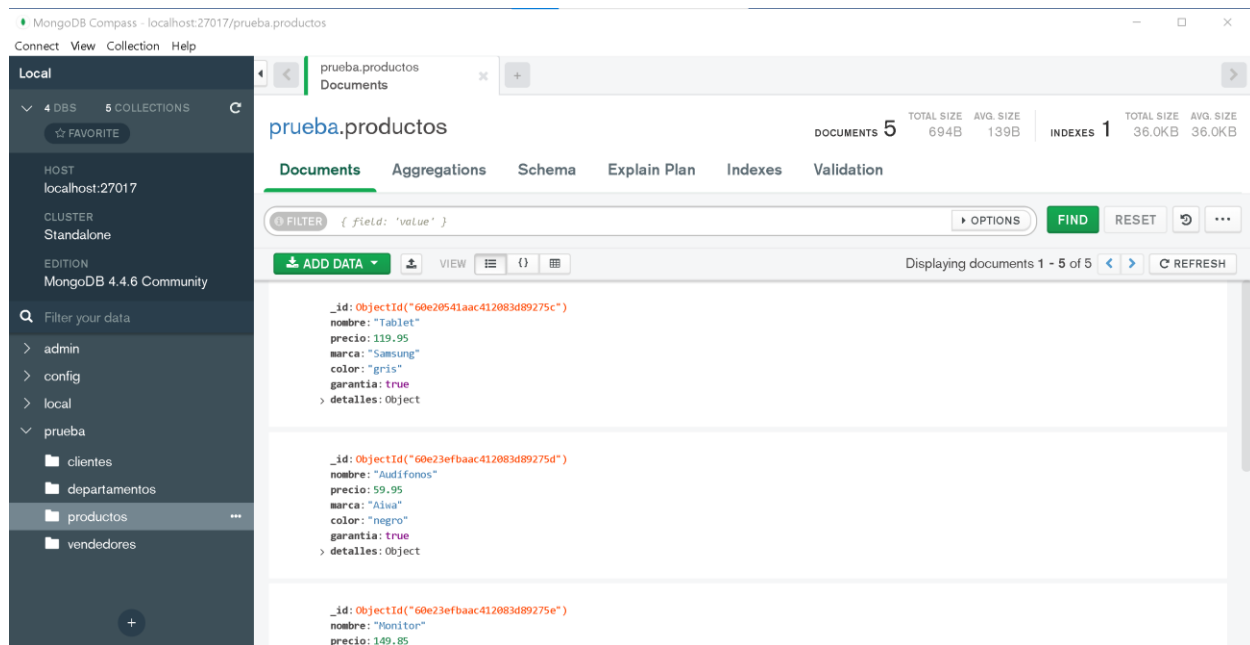
Una vez se establezca la conexión, deben aparecer las bases de datos creadas. Para esta actividad, se creó la base de datos prueba por lo cual, al abrir esta base de datos, se visualizan las colecciones existentes.



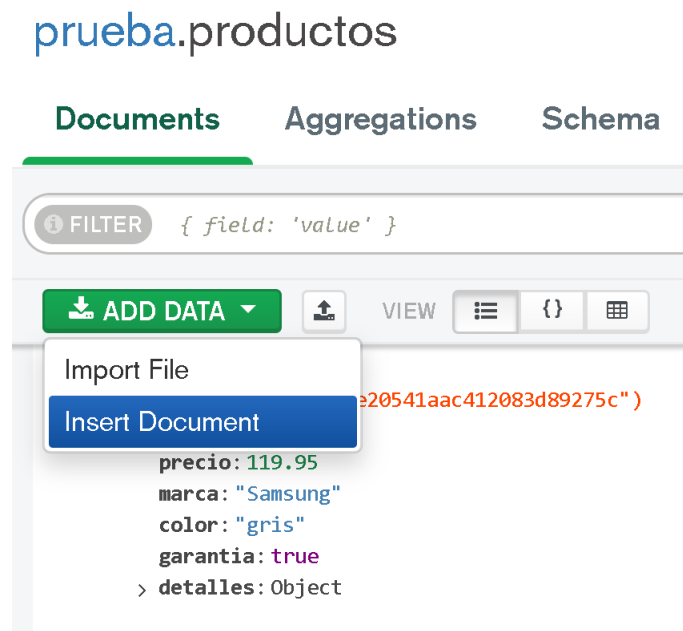
Al dar clic en prueba se muestran las 4 colecciones creadas:



Ahora puede ingresar a la colección que desee y observar los campos que contienen:



Note todas las funcionalidades que ofrece el modo gráfico de MongoDB y proceda a crear un documento en la colección productos. Para ello use la opción ADD DATA y luego escoja Insertar Documento:



Tome en consideración que debe seguir con el formato JSON para el manejo de los documentos al momento de la creación de cada uno, y que el modo gráfico le muestra los ID de los documentos de manera automática y directa. La siguiente imagen muestra un documento insertado en el modo gráfico:

Insert to Collection prueba.productos

VIEW

```
1 ▾ /**
2  * Paste one or more documents here
3  */
4 ▾ {
5 ▾   {
6     "_id": {
7       "$oid": "60e26ffe0c8f470b357d52a7"
8     },
9     "nombre": "teclado",
10    "marca": "Logitech"
11  }
12 }
```

Puede apreciar que se agregó el documento correspondiente:

prueba.productos

	DOCUMENTS	6	TOTAL SIZE	756B	AVG. SIZE	126B	INDEXES	1	TOTAL SIZE	36.0KB	AVG. SIZE	36.0KB
--	-----------	---	------------	------	-----------	------	---------	---	------------	--------	-----------	--------

Documents Aggregations Schema Explain Plan Indexes Validation

{ field: 'value' }

Displaying documents 1 - 6 of 6

`_id: ObjectId("60e23efbaac412083d89275f")`
`nombre: "Celular"`
`precio: 459.95`
`marca: "LG"`
`color: "negro"`
`garantia: false`
`> detalles: Object`
`observación: "reconstruido"`

`_id: ObjectId("60e260c7b40753c32ceb3c06")`
`nombre: "Bocinas"`
`precio: 39.49`

`_id: ObjectId("60e26ffe0c8f470b357d52a7")`
`nombre: "teclado"`
`marca: "Logitech"`

Realice la inserción de un documento en cualquiera de sus colecciones utilizando el modo gráfico y muestra las capturas correspondientes.

F. RECURSOS:

Computador con acceso a internet, Software SQL Server, acceso a plataforma ecampus.utp.ac.pa/moodle, curso de Base de Datos 1.

G. RESULTADOS:

1. Liste los documentos de la colección productos (muestre la captura de pantalla respectiva).

`db.productos.find()`

```
> db.productos.find()
{ "_id" : ObjectId("62ca957a1bc172aeea155d86"), "nombre" : "Reloj digital" }
{ "_id" : ObjectId("62ca96a71bc172aeea155d87"), "nombre" : "Audifonos", "precio" : 49.95, "marca" : "Aiwa", "color" : "negro", "garantia" : true, "detalles" : { "pais" : "Japón", "ciudad" : "Nagasaki", "fecha_producción" : ISODate("2020-12-23T05:00:00Z") } }
{ "_id" : ObjectId("62ca96a71bc172aeea155d88"), "nombre" : "Monitor", "precio" : 149.85, "marca" : "Dell", "color" : "negro", "garantia" : true }
{ "_id" : ObjectId("62ca96a71bc172aeea155d89"), "nombre" : "Celular", "precio" : 459.95, "marca" : "LG", "color" : "negro", "conservación" : "reconstruido", "garantia" : false, "detalles" : { "pais" : "China" } }
{ "_id" : ObjectId("62cb9842869d032344aa8274"), "nombre" : "tablet", "precio" : 119.95, "marca" : "Samsung", "color" : "gris", "garantia" : true, "detalles" : { "pais" : "Korea del Sur", "ciudad" : "Seúl", "fecha_producción" : ISODate("2021-05-10T05:00:00Z") } }
>
```

2. Inserte una cantidad de documentos en las colecciones de su base de datos y use diferentes esquemas. La cantidad señalada para cada colección aparece en paréntesis:

- clientes (4)

`db.Clientes.insert(`

`[`

`{"nombre":"Johel","Cargo":"Presidente Ejecutivo"},`

`{"nombre":"Kevin","Cargo":"Director de Tecnología"},`

`{"nombre":"Luis Ernesto","Cargo":"Director de Voluntariado"},`

`{"nombre":"Rolando","Cargo":"Director de contenidos"}])`

```
> db.Clientes.insert(
... [
... {"nombre":"Johel","Cargo":"Presidente Ejecutivo"},
... {"nombre":"Kevin","Cargo":"Director de Tecnología"},
... {"nombre":"Luis Ernesto","Cargo":"Director de Voluntariado"},
... {"nombre":"Rolando","Cargo":"Director de contenidos"}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 4,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>
```

db.Clientes.find()

```
> db.Clientes.find()
{ "_id" : ObjectId("62cb99c1869d032344aa8275"), "nombre" : "Johel", "Cargo" : "Presidente Ejecutivo" }
{ "_id" : ObjectId("62cb99c1869d032344aa8276"), "nombre" : "Kevin", "Cargo" : "Director de Tecnología" }
{ "_id" : ObjectId("62cb99c1869d032344aa8277"), "nombre" : "Luis Ernesto", "Cargo" : "Director de Voluntariado" }
{ "_id" : ObjectId("62cb99c1869d032344aa8278"), "nombre" : "Rolando", "Cargo" : "Director de contenidos" }
>
```

- departamentos (3)

db.Departamentos.insert(

```
[
{"nombre":"UX","Lead":"Valerie Hernández"},
{"nombre":"FrontEnd","Lead":"Kendall Kant"},
{"nombre":"Datos","Lead":"Gabriel Verbel"}])
```

```
> db.Departamentos.insert(
... [
... {"nombre":"UX","Lead":"Valerie Hernández"},
... {"nombre":"FrontEnd","Lead":"Kendall Kant"},
... {"nombre":"Datos","Lead":"Gabriel Verbel"}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 3,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>
```

db.Departamentos.find()

```
> db.Departamentos.find()
{ "_id" : ObjectId("62cb9add869d032344aa8279"), "nombre" : "UX", "Lead" : "Valerie Hernández" }
{ "_id" : ObjectId("62cb9add869d032344aa827a"), "nombre" : "FrontEnd", "Lead" : "Kendall Kant" }
{ "_id" : ObjectId("62cb9add869d032344aa827b"), "nombre" : "Datos", "Lead" : "Gabriel Verbel" }
>
```

- vendedores (2)

db.Vendedores.insert(

```
[
{"nombre":"May Janee Maldonado","Antiguedad":"2 años"},
{"nombre":"Karly Ivimas","Antiguedad":"1 año y medio"}])
```

```

> db.Vendedores.insert(
... [
... {"nombre":"May Janee Maldonado","Antiguedad":"2 años"},
... {"nombre":"Karly Ivimas","Antiguedad":"1 año y medio"}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>

```

db.Vendedores.find()

```

> db.Vendedores.find()
{ "_id" : ObjectId("62cb9be4869d032344aa827c"), "nombre" : "May Janee Maldonado", "Antiguedad" : "2 años" }
{ "_id" : ObjectId("62cb9be4869d032344aa827d"), "nombre" : "Karly Ivimas", "Antiguedad" : "1 año y medio" }
>

```

3. Muestre cómo queda el documento respectivo luego de esta modificación usando el método find y pretty.

db.Clientes.find().pretty()

```

> db.Clientes.find().pretty()
{
  "_id" : ObjectId("62cb99c1869d032344aa8275"),
  "nombre" : "Johel",
  "Cargo" : "Presidente Ejecutivo"
}
{
  "_id" : ObjectId("62cb99c1869d032344aa8276"),
  "nombre" : "Kevin",
  "Cargo" : "Director de Tecnología"
}
{
  "_id" : ObjectId("62cb99c1869d032344aa8277"),
  "nombre" : "Luis Ernesto",
  "Cargo" : "Director de Voluntariado"
}
{
  "_id" : ObjectId("62cb99c1869d032344aa8278"),
  "nombre" : "Rolando",
  "Cargo" : "Director de contenidos"
}

```

4. Actualice uno de los documentos de cualquier colección en su base de datos.

db.Clientes.update({"nombre":"Johel"},{\$set:{"Cargo":"Presidente del Consejo Directivo"}})

```
> db.Clientes.update({"nombre":"Johel"},{$set:{"Cargo":"Presidente del Consejo Directivo"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

db.Clientes.find()

```
> db.Clientes.find()
{ "_id" : ObjectId("62cb99c1869d032344aa8275"), "nombre" : "Johel", "Cargo" : "Presidente del Consejo Directivo" }
{ "_id" : ObjectId("62cb99c1869d032344aa8276"), "nombre" : "Kevin", "Cargo" : "Director de Tecnología" }
{ "_id" : ObjectId("62cb99c1869d032344aa8277"), "nombre" : "Luis Ernesto", "Cargo" : "Director de Voluntariado" }
{ "_id" : ObjectId("62cb99c1869d032344aa8278"), "nombre" : "Rolando", "Cargo" : "Director de contenidos" }
>
```

db.Clientes.find({"nombre":"Johel"}).pretty()

```
> db.Clientes.find({"nombre":"Johel"}).pretty()
{
  "_id" : ObjectId("62cb99c1869d032344aa8275"),
  "nombre" : "Johel",
  "Cargo" : "Presidente del Consejo Directivo"
}
```

5. Realice la inserción de un documento en cualquiera de sus colecciones utilizando el modo gráfico y muestra las capturas correspondientes.

The screenshot shows the MongoDB Compass interface for the 'prueba.Clientes' collection. The 'Documents' tab is selected, and the collection contains 4 documents. The first document is highlighted, showing its fields: `_id`, `nombre`, and `Cargo`.

prueba.Clientes

4 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET ...

ADD DATA VIEW {}

Displaying documents 1 - 4 of 4 REFRESH

```
_id: ObjectId('62cb99c1869d032344aa8275')
nombre: "Johel"
Cargo: "Presidente del Consejo Directivo"
```

```
_id: ObjectId('62cb99c1869d032344aa8276')
nombre: "Kevin"
Cargo: "Director de Tecnología"
```

```
_id: ObjectId('62cb99c1869d032344aa8277')
nombre: "Luis Ernesto"
Cargo: "Director de Voluntariado"
```

```
_id: ObjectId('62cb99c1869d032344aa8278')
nombre: "Rolando"
Cargo: "Director de contenidos"
```

```
/**
 * Paste one or more documents here
 */
{
  "_id": {
    "$oid": "62cba3b473c5aeca61fd238e"
  },
  "nombre": "Diego",
  "Cargo": "Secretario de la Junta Directiva"
}
```

Insert to Collection prueba.Clientes

VIEW

```
1  ▾ /**
2  ▾  * Paste one or more documents here
3  ▾  */
4  ▾  {
5  ▾    "_id": {
6  ▾      "$oid": "62cba49d73c5aeca61fd2390"
7  ▾    },
8  ▾    "nombre": "Diego",
9  ▾    "Cargo": "Secretario de la Junta Directiva"
10 ▾  }
```

Cancel

Insert

```
_id: ObjectId('62cba49d73c5aeca61fd2390')
nombre: "Diego"
Cargo: "Secretario de la Junta Directiva"
```

- Los problemas tienen una ponderación de 20 puntos y se evaluará que aparezca las evidencias respectivas de resultados.