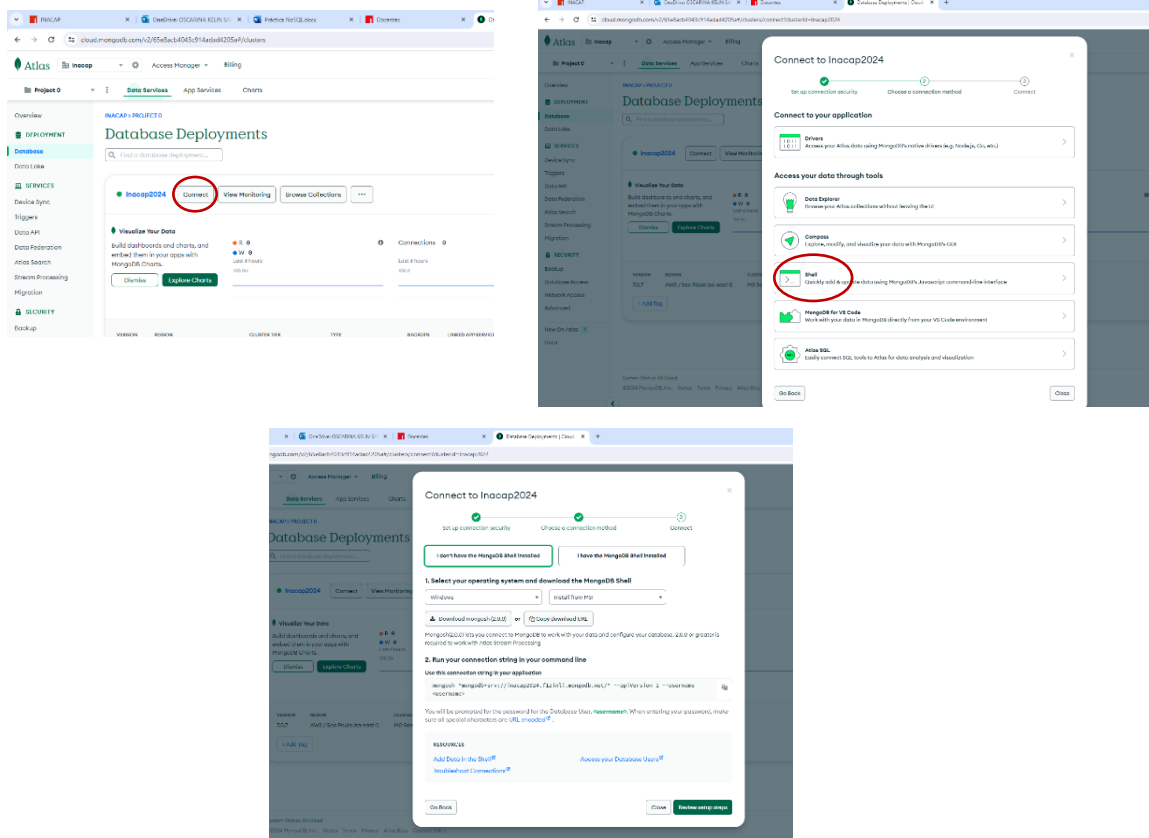
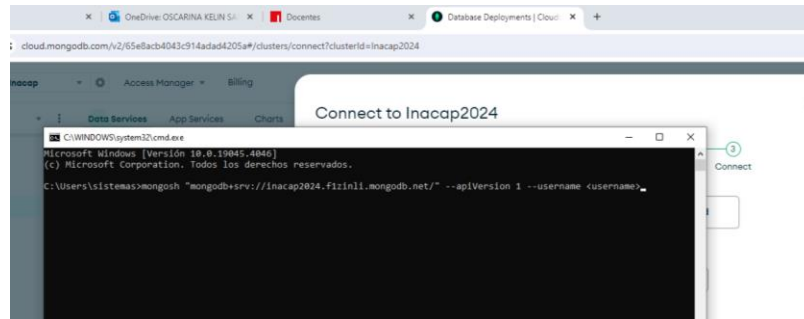


COMENZANDO CON MONGO DB. SHELL

Para iniciar con el Shell ya se debe tener instalado en el computador, de no ser así, primero debemos ingresar a la nuestra sesión en mongodb.com con nuestro usuario y contraseña, luego que estamos en la página de inicio, presionamos el botón connect, veremos las siguientes pantallas:



Al presionar el botón connect se tiene la segunda pantalla donde se encuentra el Compass, el Shell y otras formas de conexión, se da clic al Shell y en la siguiente pantalla se escoge el Sistema operativo que se tiene instalado y se descarga el ejecutable, después de instalar se debe ir a la terminal y copiar la ruta que se muestra en la pantalla, colocando el nombre de usuario.



Al colocar el nombre de usuario, solicita la contraseña, se ejecutara y mostrara el prompt del mongo.

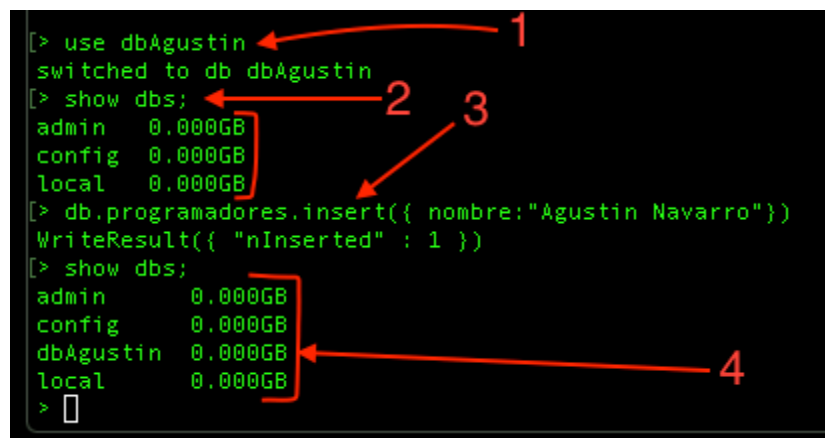
Para hacer uso de una base de datos, o crearla se usa el comando **use**, si se desea listar las bases de datos se utiliza el comando **show dbs**.

```
use [nombre de la base];
```

```
[> use primeraBaseDatos
switched to db primeraBaseDatos
[> use dbMyProyectoAgustin
switched to db dbMyProyectoAgustin
> ]
```

Para insertar y crear una colección de datos en MongoDB solo se debe usar la sentencia insert y agregar el documento (datos) en formato JSON, las colecciones se crean automáticamente en MongoDB una vez que se inserta un elemento o colección.

```
db.[coleccion].insert( [documento en formato JSON] );
```



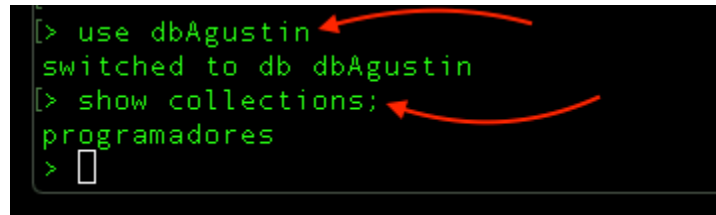
```
[> use dbAgustin
switched to db dbAgustin
[> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
[> db.programadores.insert({ nombre:"Agustin Navarro"})
WriteResult({ "nInserted" : 1 })
[> show dbs;
admin    0.000GB
config   0.000GB
dbAgustin 0.000GB
local    0.000GB
> ]
```

Para ver las colecciones disponibles en la base de datos, recuerda que primero se tiene que seleccionar la base de datos que se va a usar con el comando:

```
use [tabla]
```

Usamos show collection para ver las colecciones de la tabla seleccionada:

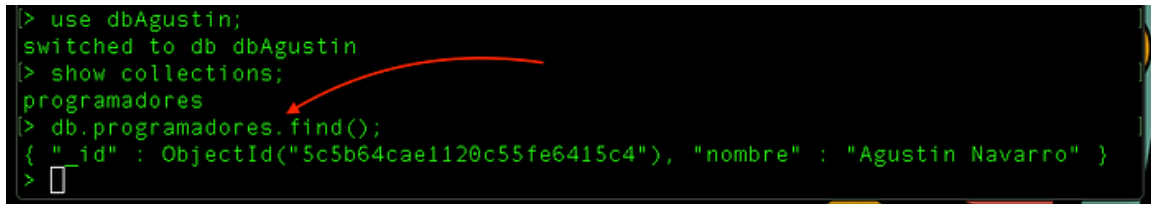
```
show collections;
```



```
[> use dbAgustin
switched to db dbAgustin
[> show collections;
programadores
> ]
```

Con el comando find() se muestra la lista de documentos “registros” de una colección, podemos filtrar o enviar al comando find para especificar los resultados de la consulta.

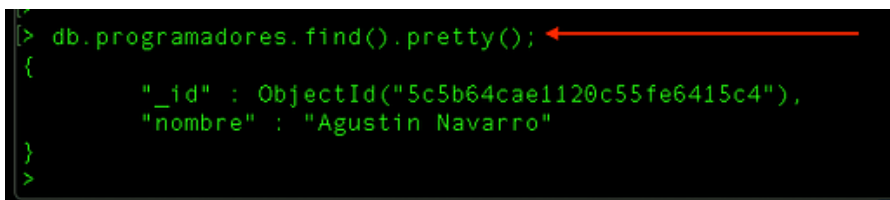
```
db.[coleccion].find();
```



```
[> use dbAgustin;
switched to db dbAgustin
[> show collections;
programadores
[> db.programadores.find();
{ "_id" : ObjectId("5c5b64cae1120c55fe6415c4"), "nombre" : "Agustin Navarro" }
> ]
```

Si se desea que los resultados se vean mucho mejor o darle una salida más visible en la pantalla solo tenemos que agregar a la consulta pretty() que hará que el resultado se vea más ordenado.

```
db.[coleccion].find().pretty();
```



```
[> db.programadores.find().pretty();
{
  "_id" : ObjectId("5c5b64cae1120c55fe6415c4"),
  "nombre" : "Agustin Navarro"
}
>
```

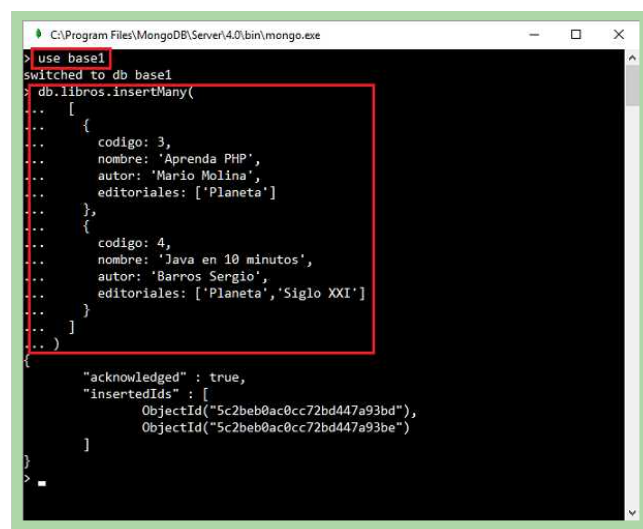
Para insertar un documento o un conjunto de documentos disponemos de los métodos:

- ✓ insertOne: Inserta un documento en una colección.
- ✓ insertMany: Inserta múltiples documentos en una colección.

Al insertar más de un documento en la colección "libros" mediante el método insertMany:

```
db.libros.insertMany(  
  [  
    {  
      codigo: 3,  
      nombre: 'Aprenda PHP',  
      autor: 'Mario Molina',  
      editoriales: ['Planeta']  
    },  
    {  
      codigo: 4,  
      nombre: 'Java en 10 minutos',  
      autor: 'Barros Sergio',  
      editoriales: ['Planeta', 'Siglo XXI']  
    }  
  ]  
)
```

En la consola de MongoDB tenemos como resultado:



```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe  
> use base1  
switched to db base1  
> db.libros.insertMany(  
...  [  
...    {  
...      codigo: 3,  
...      nombre: 'Aprenda PHP',  
...      autor: 'Mario Molina',  
...      editoriales: ['Planeta']  
...    },  
...    {  
...      codigo: 4,  
...      nombre: 'Java en 10 minutos',  
...      autor: 'Barros Sergio',  
...      editoriales: ['Planeta', 'Siglo XXI']  
...    }  
...  ]  
... )  
{  
  "acknowledged" : true,  
  "insertedIds" : [  
    ObjectId("5c2beb0ac0cc72bd447a93bd"),  
    ObjectId("5c2beb0ac0cc72bd447a93be")  
  ]  
}
```

Se debe tener en cuenta que si recién se activó la consola se debe hacer uso de la base de datos "base1" mediante el comando "use":

```
use base1
```

Luego se llama al método "insertMany" y le pasamos un array con todos los documentos a almacenar en la colección "libros".

Tener en cuenta que se utiliza la consola de MongoDB (shell) con el objetivo a aprender los comandos esenciales, luego en la realidad estos datos serán enviados desde nuestras aplicaciones que podrán estar escritas en Python, Ruby, C#, Java etc.

Podemos borrar el contenido de la consola de MongoDB (shell) mediante el comando:

```
cls
```

El mismo resultado lo podemos obtener presionando las teclas: CTRL + L.

Mostremos los documentos almacenados en la colección "libros" mediante el método "find":



```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> db.libros.find()
{ "_id" : ObjectId("5c27f1edcd0f463fbb3699ce"), "codigo" : 1, "nombre" : "El aleph", "autor" : "Borges", "editoriales" : [ "Planeta", "Siglo XXI" ] }
{ "_id" : ObjectId("5c27f4ffcd0f463fbb3699cf"), "codigo" : 2, "nombre" : "Martin Fierro", "autor" : "Jose Hernandez", "editoriales" : [ "Planeta" ] }
{ "_id" : ObjectId("5c2beb0acc72bd447a93bd"), "codigo" : 3, "nombre" : "Aprende a PHP", "autor" : "Mario Molina", "editoriales" : [ "Planeta" ] }
{ "_id" : ObjectId("5c2beb0acc72bd447a93be"), "codigo" : 4, "nombre" : "Java en 10 minutos", "autor" : "Barros Sergio", "editoriales" : [ "Planeta", "Siglo XXI" ] }
>
```

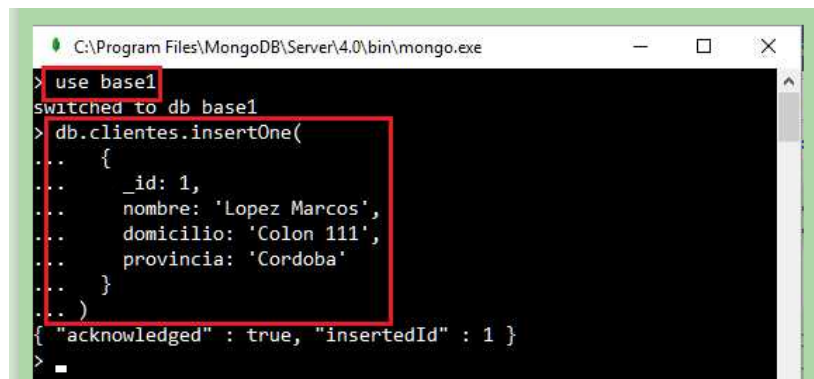
Como podemos observar al ejecutar el método "find" nuestra colección "libros" tiene almacenado 4 documentos.

En MongoDB todo documento requiere un campo clave que se debe llamar _id. Si nosotros como desarrolladores no definimos dicho campo el mismo se crea en forma automática y se carga un valor único.

Podemos definir y cargar un valor en el campo `_id` cuando inicializamos un documento:

```
db.clientes.insertOne(  
  {  
    _id: 1,  
    nombre: 'Lopez Marcos',  
    domicilio: 'Colon 111',  
    provincia: 'Cordoba'  
  }  
)
```

Cuando ejecutamos la inserción desde el shell de MongoDB tenemos como resultado:



```
C:\Program Files\MongoDB\Server\4.0\bin>mongo.exe  
> use base1  
switched to db base1  
> db.clientes.insertOne(  
... {  
...   _id: 1,  
...   nombre: 'Lopez Marcos',  
...   domicilio: 'Colon 111',  
...   provincia: 'Cordoba'  
... }  
... )  
{ "acknowledged" : true, "insertedId" : 1 }  
>
```

Cuando se ejecuta el método `insertOne` nos retorna un objeto JSON informando del resultado de la inserción mediante un objeto con dos campos, el primero `acknowledged` que indica si el documento fue admitido en la colección y el `_id` que en este caso lo define el usuario de la base de datos.

Si se produce un error nos genera un objeto JSON con otra estructura, probemos de intentar de ingresar un segundo documento con el mismo `_id`:

```
db.clientes.insertOne(  
  {  
    _id: 1,  
    nombre: 'Perez Ana',  
    domicilio: 'San Martin 222',  
    provincia: 'Santa Fe'  
  }  
)
```

```

C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> db.clientes.insertOne(
...
...   _id: 1,
...   nombre: 'Perez Ana',
...   domicilio: 'San Martin 222',
...   provincia: 'Santa Fe'
... )
2019-01-02T09:14:49.708-0300 E QUERY [js] WriteError: E11000 duplicate key error collection: base1.clientes index: _id_dup key: { : 1.0 } :
WriteError({
  "index" : 0,
  "code" : 11000,
  "errmsg" : "E11000 duplicate key error collection: base1.clientes index: _id_dup key: { : 1.0 }",
  "op" : {
    "_id" : 1,
    "nombre" : "Perez Ana",
    "domicilio" : "San Martin 222",
    "provincia" : "Santa Fe"
  }
})
WriteError@src/mongo/shell/bulk_api.js:461:48
Bulk/mergeBatchResults@src/mongo/shell/bulk_api.js:841:49
Bulk/executeBatch@src/mongo/shell/bulk_api.js:906:13
Bulk/this.execute@src/mongo/shell/bulk_api.js:1150:21
DBCollection.prototype.insertOne@src/mongo/shell/crud_api.js:252:9
@(shell):1:1
>

```

Nos retorna un objeto JSON que entre otros campos define uno llamado errmsg con el mensaje de error.

Si nuestra aplicación administra el campo '_id' hay que tener en cuenta que nunca puede repetirse y en el caso que intentemos ingresar un documento con clave repetida luego dicho documento no se inserta en la colección.

Mostremos todos los documentos almacenados en la colección libros:

```

C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> db.clientes.find()
{ "_id" : 1, "nombre" : "Lopez Marcos", "domicilio" : "Colon 111", "provincia" : "Cordoba" }
>

```

Hay uno solo ya que el segundo intento no se cargó.

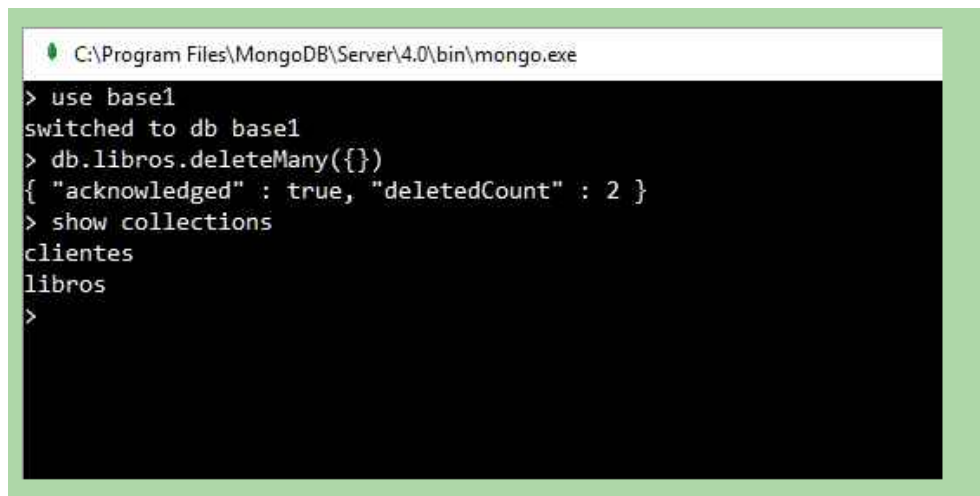
Hemos visto como se crea una base de datos, una colección y se insertan documentos en la misma.

Si queremos eliminar todos los documentos de una colección debemos utilizar el método "deleteMany" aplicado a una colección existente:

```
use base1
db.libros.deleteMany({})
show collections
```

Debemos pasar un objeto vacío que se indica con las llaves abiertas y cerradas {}. Luego veremos que podemos borrar solo los documentos que cumplen cierta condición.

Es importante notar que luego de llamar al método deleteMany la colección "libros" sigue existiendo:

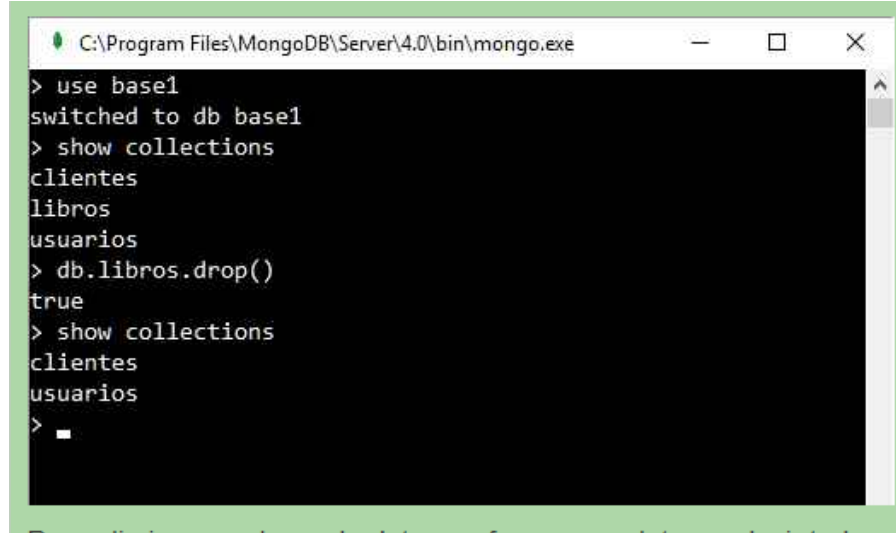


```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> use base1
switched to db base1
> db.libros.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 2 }
> show collections
clientes
libros
>
```

Para eliminar los documentos de una colección y la colección propiamente dicha debemos emplear el método "drop":

```
use base1
db.libros.drop()
show collections
```


Luego de llamar al método drop de la colección "libros" la misma deja de existir:

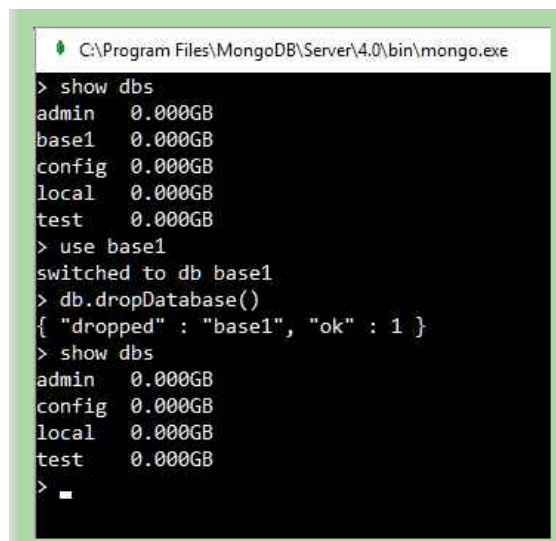


```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> use base1
switched to db base1
> show collections
clientes
libros
usuarios
> db.libros.drop()
true
> show collections
clientes
usuarios
> _
```

Para eliminar una base de datos en forma completa, es decir todas sus colecciones y documentos debemos emplear el método dropDatabase del objeto "db":

```
show dbs
use base1
db.dropDatabase()
show dbs
```

El método dropDatabase elimina la base de datos activa:



```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
> show dbs
admin  0.000GB
base1  0.000GB
config 0.000GB
local  0.000GB
test   0.000GB
> use base1
switched to db base1
> db.dropDatabase()
{ "dropped" : "base1", "ok" : 1 }
> show dbs
admin  0.000GB
config 0.000GB
local  0.000GB
test   0.000GB
> _
```

Problemas propuestos

1. Crear una base de datos llamada "SEDE".
2. Agregar las siguientes colecciones: "alumnos", "profesores", "ramos", inserta al menos 3 documentos con una estructura a su elección.
3. Mostrar todas las bases de datos actuales.
4. Crear una base de datos "Biblioteca".
5. Crear las colecciones "Libros" y "Clientes" en la base de datos "Biblioteca" y agregar al menos 2 documentos con una estructura a su elección.
6. Activar la base de datos "local" y luego imprimir las colecciones existentes.
7. Activar la base de datos "Biblioteca" y luego imprimir las colecciones existentes.
8. Mostrar nuevamente las colecciones existentes en la base de datos "SEDE".
9. Insertar 2 documentos en la colección alumnos.
10. Mostrar todos los documentos de la colección "ramos".
11. Eliminar la colección "ramos"
12. Eliminar la base de datos "Biblioteca" y mostrar las bases de datos existentes.