Primeros Pasos

Después de la instalación, en una consola hago correr el servidor

```
C:> mongod
```

Y en otra consola, corro mongo

```
C:> mongo
```

Ctrl+L para limpiar la pantalla

Para mostrar las bases de datos

```
C:> show dbs
```

Para crear una db `(database)`

```
C:> use tienda
```

Creo una colección nueva llamada productos, e inserto un nuevo documento

```
C:> db.productos.insertOne({producto:"libro1", precio: 9.99})
```

Para ver todos los documentos de esta colección

```
C:> db.productos.find()
```

Para verlo en un formato más amigable

```
C:> db.productos.find().pretty()
```

Puedo poner los campos que quiera

```
    C:> db.productos.insertOne({producto:"libro1", precio: 9.99, descripción:"buen
libro" })
```

Puedo insertar un documento dentro de otro

```
     C:> db.productos.insertOne({producto:"libro1", precio: 9.99,
descripción:"buen libro", detalles:{tapa: "dura", páginas: 200} })
```

Creo la base de datos usuarios. En realidad no se crea hasta que no le introduzco datos

```
    C:> use usuarios
```

Creo la colección usuarioDatos e inserto un documento

```
    C:> db.usuarioDatos.insertOne({nombre:"Juan", edad: 20})
```

Al insertarlo, ObjectId tiene la información de la hora en que se introdujeron los datos

- Tipos de datos: puede albergar números, strings, booleanos, otros documentos...
- No pueden haber 2 id's iguales

CRUD

- C de Create
- R de Read
- U de Update
- D de Delete
- Hay 3 casos de uso habituales
    - Aplicación, una web, una app: MongoDB driver
    - Analítica o Businnes Inteligence: conector BI/ shell
    - Administración: shell
- Todos ellos conectando al MongoDB Server

Create

```
insertOne(data, options)
insertMany(data, options)
```

Read

```
find(filter, options)
findOne(filter, options)
```

Update

```
updateOne(filter, data, options)
updateMany(filter, data, options)
replaceOne(filter, data, options)
```

Delete

```
deleteOne(filter, options)
deleteMany(filter, options)
```

El filtro puede ser el nombre o cualquier campo

Para actualizar, por ejemplo, esto da ERROR

```
   C:> db.usuarioDatos.updateOne( {nombre: "Juan"}, {campo_añadido: "nuevo campo"}
)
```

- PORQUE? porque para actualizar, se usan ATOMIC OPERATORS
- En este caso $set, debo enmarcar entre llaves el objeto

  ```
    C:> db.usuarioDatos.updateMany( { },  { $set {campo_añadido: "nuevo
  campo"} } )
  ```

- En lugar de updateOne, puedo usar updateMany({ }), con el objeto vacío, para que le agregue el nuevo campo a todos los documentos

Para borrarlos todos puedo usar deleteMany con el filtro 'campo_añadido' También podría haber usado deleteMany({ }) con un objeto vacío para borrarlos todos

```
C:> db.usuarioDatos.deleteMany({campo_añadido:"nuevo campo"})
```

insertMany

Para usar insertMany hay que colocar los objetos en un arreglo llamado matriz, separado por comas

```
C:> db.usuarios.insertMany( [ { objeto1: objeto }, { objeto2: objeto } ] )
```

find

find nos muestra los documentos que cumplan esa condición/filtro

```
C:> db.usuario.find(nombre:"Juan")
```

- findOne nos muestra el primero que se insertó que encuentre con esa condición/filtro a travñes del id
- Si quisiera buscar mayores de 25 usaría el atomic operator $gt de "greater than"

  ```
  C:> db.usuario.findOne( { edad: { $gt: 25 } } )
  ```

updateOne, update, replace

Para actualizar usaremos el id del objeto y el atomic operator $set

```
C:> db.usuarios.updateOne( {_id:ObjectId("675476345364") }, { $set:{ alta:
false} } )
```

- Con el update a secas funcionaría igual
- Con replace debería introducir todos los datos que quiero actualizar, ya que lo que introduzca es lo que será el documento

find y el objeto cursor

find no devuelve todos los datos, solo los primeros 20 por defecto

- Se puede forzar para que los muestre todos

```
C:>db.nombre.find().toArray()
```

- También se puede usar el forEach

```
C:> db.nombre.forEach(nombre) =>{ printjson(nombre) }
```

- .pretty funciona solo con find, no con findOne ni ningún otro

Proyección en MongoDB

Cómo hacer para que devuelva solo un campo

- Para esto se usa la proyección.
- Se pone el find con un objeto vacío y solo se usa 1 o 0.
- Si quiero que lo imprima uso 1

```
C:> db.nombres.find( {}, {nombre: 1} )
```

- Si quiero exclui el id, le pongo un 0 `(se incluye por defecto)`

```
C:> db.nombres.find( {}, {nombre: 1, _id: 0} )
```

- En la proyección se excluye todo menos el campo seleccionado con un 1 y el id que viene por defecto

Documentos y matrices incrustados

Un documento puede tener otros documentos incrustados, y a su vez dentro de estos otros.

- Se puede llegar hasta 100 niveles, que es mucho y casi nunca se utiliza
- Hay un máximo de 16 MB por documento
- Además de documentos también se pueden incrustar matrices`arreglos`
- Con updateMany, si quiero actualizarlos todos, empiezo con un objeto vacío

```
C:> db.nombres.updateMany( {}, { $set: {estado:{descripcion:"activo",
actualizado: true } } } )
```

Matrices

```
    C:> db.nombres.updateOne( {nombre: "Juan"}, {$set: {aficiones: ["malabarismo",
"windsurf", "pingpong"] } } )
```

Accediendo a los datos estructurados

Para acceder uso la notación de punto

```
    C:> db.nombres.findOne( { nombre: "Juan" } ).aficiones
```

Si quiero que me devuelva los documentos donde aparezca el campo de un objeto interno de otro objeto

```
    C:> db.nombres.find( {objeto.objeto_interno: "campo"} ).pretty()
```

Puede meterse dentro de la matriz donde aficiones es cantar

```
    C:>db.nombres.find( {aficiones: "cantar"} )
```