

# Gestión de Datos para Robótica

## T2b - MongoDB

Álvaro Vázquez Álvarez  
Departamento de Electrónica e Computación

✉ [alvaro.vazquez@usc.es](mailto:alvaro.vazquez@usc.es)

📍 Pabellón III - Despacho 4

Curso 2023-2024

# Tabla de contenidos

---

- Introducción
- Elementos de MongoDB
- Herramientas
- Conexión a MongoDB
- Manejo de BD
  - Importar Bases de Datos
  - Selección
  - Inserción
  - Consultas
  - Actualizaciones
- Colecciones
- Bibliografía

# Introducción

---

MongoDB es la BD NoSQL más conocida.

Modelo documental → los documentos se basan en JSON.

2021 - 2022 → MongoDB v5.0.

Destaca porque:

- Soporta esquemas dinámicos: diferentes documentos de una misma colección pueden tener atributos diferentes.
- No soporta *joins*, ya que no escala bien.
- No soporta transacciones → lo que en un SGBD puede suponer múltiples operaciones, con MongoDB se puede hacer en una sola operación al insertar/actualizar todo un documento de manera atómica

# Elementos

Dentro de una instancia podemos tener 0 o más bases de datos

- Contenedor de alto nivel

Cada base de datos tendrá 0 o más colecciones.

- Similar a una tabla.
- Tipos: normal, limitadas

Las colecciones contiene 0 o más **documentos**

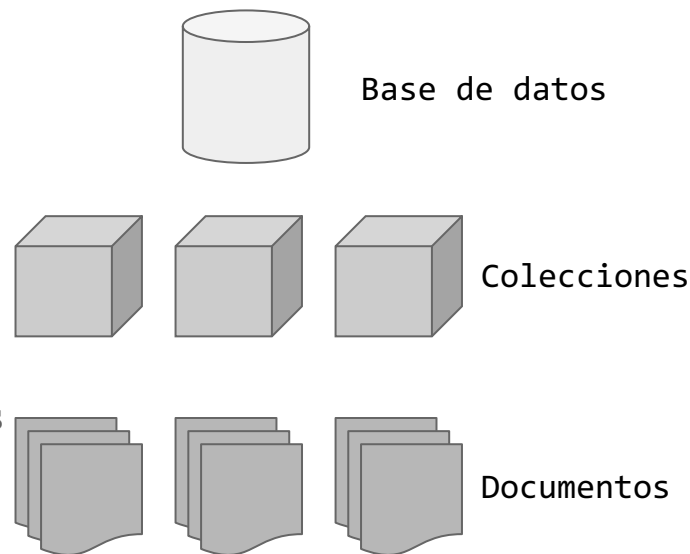
Cada documento contiene 0 o más atributos, compuestos de **parejas clave/valor**.

- No siguen ninguna esquema
- Dos documentos de una misma colección pueden contener todos los atributos diferentes entre sí.

Soporta índices para acelerar la búsqueda de datos.

Al realizar cualquier consulta, se devuelve un cursor

- Permite contar, ordenar, limitar o saltar documentos



# Herramientas y utilidades

---

Demonio → mongod

Shell → mongo

Importar / Exportar → mongoimport / mongoexport

Backup → mongodump / mongorestore

BSON a JSON → bsondump

Rendimiento → mongostat

Drivers → <http://docs.mongodb.org/ecosystem/drivers/>

GUI → RoboMongo: <http://robomongo.org/>

# Conexión a MongoDB

## Conexión a MongoDB

```
mongo [--host <host>] [--port <2766|port>]
```

```
~$ mongo
```

```
mongo
```

```
MongoDB shell version v3.6.8
```

```
connecting to: mongodb://127.0.0.1:27017
```

```
Implicit session: session { "id" : UUID("18fb83c3-a978-4e03-8791-22ae3dfb6620") }
```

```
MongoDB server version: 5.0.6
```

```
>
```

# Manejo de BD

---

## Importar Bases de Datos

```
mongoimport [--host <host>] [--port <2766|port>] [--db <db_name>]
             [--collection <collection_name>] --file <file_name>
```

```
~$ mongoimport --host localhost --type json --db aux --collection
students --file students.json
```

```
2022-04-26T16:35:26.005+0000      connected to: localhost
2022-04-26T16:35:26.034+0000      imported 200 documents
```

# Manejo de BD

---

## Consultar Bases de Datos

```
show dbs
```

```
~$ show dbs
```

```
admin    0.000GB
config   0.000GB
local    0.000GB
```

## Seleccionar Bases de Datos

```
use <database>
```

```
use ejemplo
switched to db ejemplo
>
```



# Manejo de BD

## Inserciones en Bases de Datos

- Crear una colección

```
db.createCollection(<collectionName>)
```

```
db.createCollection("ejemplo")
(ok : 1)
```

- Insertar documentos (I)

```
db.<collection_name>.insert(<document|variable>)
```

```
db.<collection_name>.save(<document|variable>)
```



No admite duplicados. Misma clave→error



Admite duplicados. Misma clave→actualiza

# Manejo de BD

## Inserciones en Bases de Datos

- Insertar documentos (II)

```
usuario1 = {
  nombre: "Eduardo",
  edad: 22,
  altura: 1.85,
  casado: true,
  fechaAlta: new Date()
}
```



Asignación a una variable usando formato tabular

```
usuario2 = { nombre: "María", apellido1: "Pérez",
  apellido2: "Rodríguez", edad: 32,
  altura: 1.74, fechaAlta: new Date() }
```



Asignación a una variable usando  
formato compacto

```
usuario3 = { nombre: "Concha", apellido1: "López", apellido2: "Arias", edad: 22, altura: 1.66 }
```

```
> db.ejemplo.insert(usuario1)
```



Insertar un documento usando una variable.

```
> db.ejemplo.insert([usuario2, usuario3])
```



Insertar usando múltiples variables

```
> db.ejemplo.insert({nombre:"Luis", apellido1:"López", apellido2:"Miras"})
```



Insertar documento

# Manejo de BD

## Consultas en Bases de Datos

- Búsquedas básicas

Campos a mostrar  
--SELECT en SQL--

{ } → muestra todos los campos  
{c1:1,c2:1,...} → indica los campos a mostrar

```
db.<collection_name>.find({query},{...}).[modifiers()]
db.<collection_name>.findOne({query},{...}).[modifiers()]
```

Devuelve todos los resultados que coinciden con {query}

Devuelve el primer resultado que coincide con {query}

Condición que deben satisfacer los resultados  
--WHERE en SQL--

Opciones de manipulación/visualización de los resultados

{ } → sin condiciones. Muestra todos los resultados.  
{c1:v1, c2:v2} → muestra los resultados que cumplan las condiciones especificadas en c1:v1 Y c2:v2

.pretty(): muestra el resultado formateado.  
.sort(): muestra el resultado ordenado según criterio  
.limit(): limita los resultados obtenidos.  
.skip(): ignora los N primeros documentos.  
.count(): obtiene la cuenta de documentos devueltos.  
.toArray(): convierte los resultados a un array.

# Manejo de BD

## Consultas en Bases de Datos

### • Búsquedas básicas (I)

```
> db.ejemplo.find({})
```

Si no se indica nada se muestran todos los resultados y todos los campos

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23,
  "altura" : 1.85, "casado" : true, "fechaAlta" : ISODate("2022-04-25T11:44:00.453Z") }
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" :
  "Pérez", "apellido2" : "Rodríguez", "edad" : 32, "altura" : 1.74, "fechaAlta" :
  ISODate("2022-04-25T11:44:08.062Z") }
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha", "apellido1" :
  "López", "apellido2" : "Arias", "edad" : 22, "altura" : 1.66 }
```

Todos los documentos tienen un `_id` (único).  
+ Usuario lo crea manualmente  
+ MongoDB lo crea automáticamente.

```
> db.ejemplo.find({nombre:"Eduardo"})
```

Obtener solo los que el nombre sea Eduardo

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23, "altura" : 1.85,
  "casado" : true, "fechaAlta" : ISODate("2022-04-25T11:44:00.453Z") }
```

# Manejo de BD

## Consultas en Bases de Datos

- Búsquedas básicas (II)

```
> db.ejemplo.find({nombre:"Eduardo"},{nombre:1,edad:1})
```

← Mostrar los campos nombre y edad.

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23 }
```



Siempre se muestra el \_id aunque no se indique explícitamente.

Obtener nombre y edad de los usuarios  
que no cumplen la condición



```
> db.ejemplo.find({edad:{$ne:23}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("62668373b2a119d14cdadcfd"), "nombre" : "María", "edad" : 32 }
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores lógicos (I)


Se hace un OR por cada uno de los elementos del array.  
Si uno se cumple → TRUE y ya NO sigue (OR cortocircuito)



```
db.ejemplo.find({$or:[or_element1, or_element2, or_elementN]}, {projections})
db.ejemplo.find({$and:[and_element1, and_element2, and_elementN]}, {projections})
```



Se hace un AND por cada uno de los elementos del array.  
Si uno NO se cumple → FALSE ya NO sigue (AND cortocircuito)



```
{ $and: [...] }
{ $or: [...] }
{ $not: {} }
{ $nor: [...] }
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores lógicos (II)

```
> db.ejemplo.find({$or:[{nombre:"Eduardo"}, {nombre:"Concha"}]}, {nombre:1, edad:1})

{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23 }
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha", "edad" : 22 }
```

```
> db.ejemplo.find({$and:[{nombre:"Eduardo"}, {edad:22}]}, {nombre:1, edad:1})
—
```



NO hay ningún usuario que se llame Eduardo Y tenga 22 años.

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación genéricos (I)

```
db.<collection_name>.find({c1:{...:v1}}, {projections})
```

Operadores de comparación

{\$gt:<v1>} → mayor que  
 {\$gte:<v1>} → mayor o igual que  
 {\$lt:<v1>} → menor que  
 {\$lte:<v1>} → menor o igual que  
 {\$eq:<v1>} → igual que  
 {\$ne:<v1>} → distinto que  
 {\$in:[v1,v2,...]} → pertenezca al menos a un valor



COMPARACIÓN DE  
 STRINGS NO SE REALIZA  
 POR SU ASCII,  
 SE REALIZA COMO EN UN  
 DICCIONARIO  
 (orden lexicográfico)



# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación genéricos (I)

Edad mayor (\$gt) que 23



```
> db.ejemplo.find({edad:{$gt:23}}, {nombre:1, edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" :  
"María", "edad" : 32 } }
```

Edad menor (\$lt) que 30


y (\$and)

Altura mayor (\$gt) que 1.70



```
> db.ejemplo.find({$and:[{edad:{$lt:30}}, {altura:{$gt:1.70}}]}, {nombre:1, edad:1, altura:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23, "altura" : 1.85 }
```



```
{ $gt:value } (>)  
{ $gte:value } (>=)  
{ $lt:value } (<),  
{ $lte:value } (<=)  
{ $eq:value } (=),  
{ $ne:value } (<>)  
{ $in:[v1,v2,...] } (∈)
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación genéricos (II)

Edad pertenezca a algún valor  
del vector (\$in)




```
> db.ejemplo.find({edad:{$in:[22, 23]}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo",  
  "edad" : 23 }  
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha",  
  "edad" : 22 }
```

```
> db.ejemplo.find({edad:{$ne:22}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23 }  
{ "_id" : ObjectId("62668373b2a119d14cdadcfd"), "nombre" : "María", "edad" : 32 }
```



```
{ $gt:value } (>)  
{ $gte:value } (>=)  
{ $lt:value } (<),  
{ $lte:value } (<=)  
{ $eq:value } (=),  
{ $ne:value } (<>)  
{ $in:[v1,v2,...] } (€)
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación con strings (I)

```
db.<collection_name>.find({c1:{$...}}, {projections})
```

Operadores de comparación

{\$exists:true|false} → determina si un determinado campo existe o no  
 {\$type:"data\_type"} → comprueba si un dato es de un determinado tipo  
 {\$regex:"reg\_exp"} → busca la coincidencia de un patrón dentro de un texto

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación con strings (I)

Devolver los documentos en el que existan  
({\$exist:true}) los campos apellido1 Y(,) apellido2.



```
> db.ejemplo.find({apellido1:{$exists:true},{apellido2:{$exists:true}},
{apellido1:1,apellido2:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "apellido1" : "Pérez", "apellido2" : "Rodríguez" }
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "apellido1" : "López", "apellido2" : "Arias" }
```

```
> db.ejemplo.find({edad:{$gt:20},{edad:{$lt:23}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23}
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha", "edad" : 22 }
```

```
> db.ejemplo.find({edad:{$lt:23},{edad:{$gt:20}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" : 23}
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha", "edad" : 22 }
```

AND implícito. INCONSISTENTE!

¿?

{ \$exists: true | false }  
{ \$type: "data\_type" }  
{ \$regex: "reg\_exp" }



# Manejo de BD

## Consultas en Bases de Datos

- Operadores de comparación con strings (II)

Obtener los nombres que tengan cualquier carácter y terminan en una a (minúscula). Por defecto es case-sensitive



```
> db.ejemplo.find({nombre:{$regex:".*a$"}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" : "Pérez", "apellido2" : "Rodríguez" }
{ "_id" : ObjectId("626683c8b2a119d14cdadcfe"), "nombre" : "Concha", "edad" : 22 }
```


Obtener los nombres que empiecen por m y terminen en a



Opciones: Habilitar ignore-case

```
> db.ejemplo.find({nombre:{$regex:"^m.*a$"},$options:"i"}},{nombre:1,edad:1})
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" : "Pérez", "apellido2" : "Rodríguez" }
```



```
{ $exists:true|false }
{ $type:"data_type" }
{ $regex:"reg_exp" }
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de consulta sobre arrays (I)

```


usuario4 = {
  nombre: "Juan",
  apellido1: "Pérez",
  apellido2: "Pérez",
  edad: 55,
  altura: 1.74,
  email: ["jpp@empresa.com", "jpp@gmail.com"],
  rrss: [
    {
      nombre: "instagram",
      usuario: "@juan_pp",
      seguidores: 8250
    },
    {
      nombre: "tiktok",
      usuario: "@jpp_tiktok",
      seguidores: 25
    }
  ]
}
  
```

← Array de Strings

← Documento 1

← Array de Documentos

← Documento 2



Inserción de  
documentos con  
vectores


# Manejo de BD

## Consultas en Bases de Datos

- Operadores de consulta sobre arrays (II)

```
db.<collection_name>.find({c1:{$...}}, {projections})
```

Operadores de comparación



```
{ $all: [v1, v2, ...] }  
{ $in: [v1, v2, ...] }  
{ $nin: [v1, v2, ...] }  
{ $size: <number> }  
{ $slice: [skip, limit] }
```

<value> → comprueba si un campo tiene el valor especificado.

{ \$all: [v1, v2, ...] } → comprueba si un campo tiene todos los valores especificados

{ \$in: [v1, v2, ...] } → comprueba si un campo tiene al menos uno de los valores especificados

{ \$nin: [v1, v2, ...] } → comprueba si un campo no tiene ninguno de los valores especificados

{ \$size: <number> } → comprueba si el tamaño del vector es del tamaño especificado

{ \$slice: [skip, limit] } → permite acotar el número de resultados devueltos

skip → número de elementos a ignorar (comenzando por el principio)

limit → número máximo de resultados a mostrar.

# Manejo de BD

## Consultas en Bases de Datos


- Operadores de consulta sobre arrays (III)

Busca una única coincidencia en el vector



```
> db.ejemplo.find({email:"jpp@gmail.com"}, {nombre:1, email:1}).pretty()

{
  "_id" : ObjectId("62679dc3bd92fbd18b5b6ca7"),
  "nombre" : "Juan",
  "email" : [
    "jpp@empresa.com",
    "jpp@gmail.com"
  ]
}
```



```
{ $all: [v1, v2, ...] }
{ $in: [v1, v2, ...] }
{ $nin: [v1, v2, ...] }
{ $size: <number> }
{ $slice: [skip, limit] }
```



# Manejo de BD

## Consultas en Bases de Datos


- Operadores de consulta sobre arrays (IV)

```
> db.ejemplo.find({email:{$all:["jpp@gmail.com", "jpp@empresa.com"]}},
{nombre:1, email:1}).pretty()
```

```
{
  "_id" : ObjectId("62679dc3bd92fbd18b5b6ca7"),
  "nombre" : "Juan",
  "email" : [
    "jpp@empresa.com",
    "jpp@gmail.com"
  ]
}
```



Todos los elementos  
tienen que estar en el  
vector



```
{$all:[v1,v2,...]}
{$in:[v1,v2,...]}
{$nin:[v1,v2,...]}
{$size:<number>}
{$slice:[skip,limit]}
```

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de consulta sobre arrays (V)


```
> db.ejemplo.find({email:{$size:3}}, {nombre:1, email:1}).pretty()
—
```



Devuelve los documentos cuyo tamaño del array email sea 3.

Mostrar un máximo de 3 sin la primera entrada en el array

```
> db.ejemplo.find({email:{$in:["avl@empresa.com", "jpp@empresa.com"]}},
{email:{$slice:[1,3]}, {nombre:1}}
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca7"), "nombre" : "Juan", "email" : [ "jpp@gmail.com" ] }
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [ "avl@gmail.com" ] }
```



```
{ $all: [v1, v2, ...] }
{ $in: [v1, v2, ...] }
{ $nin: [v1, v2, ...] }
{ $size: <number> }
{ $slice: [skip, limit] }
```

Al menos uno de ellos (\$in)

SE PUEDE COMBINAR LAS OPERACIONES DE CONSULTAS SOBRE ARRAYS CON CUALQUIER OPERADOR ANTERIOR

# Manejo de BD

## Consultas en Bases de Datos

- Operadores de consulta sobre subdocumentos (I)

Consulta estándar (como si fuese otro campo más)



```
> db.ejemplo.find({rrss:{nombre:"tiktok", usuario:"@avallen"}}, {nombre:1})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana" }
```

Acceso “campo.campo\_subdocumento”



```
> db.ejemplo.find({"rrss.nombre":"instagram"}, {nombre:1})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca7"), "nombre" : "Juan" }
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana" }
```



DOT NOTATION



SE PUEDE COMBINAR LAS OPERACIONES DE CONSULTAS SOBRE SUBDOCUMENTOS  
CON CUALQUIER OPERADOR ANTERIOR

# Manejo de BD

## Consultas en Bases de Datos

- Modificadores sobre consultas (I)

```
> db.ejemplo.find({query},{...}).[modifiers()]
```

↑  
Opciones de manipulación/visualización de los  
↓  
resultados

.pretty() → muestra el resultado formateado.  
 .sort(c1:1|-1,...) → muestra el resultado ordenado según criterio  
 .limit(<number>) → limita los resultados obtenidos.  
 .skip(<number>) → ignora los N primeros documentos.  
 .count() → obtiene la cuenta de documentos devueltos.  
 .toArray() → convierte los resultados a un array.

# Manejo de BD

## Consultas en Bases de Datos

- Modificadores sobre consultas (I)

```
> db.ejemplo.find({nombre:{$regex:".*a$"}},{nombre:1,apellido1:1,apellido2:1}).sort({apellido1:1,apellido2:-1})
```



Ordenar los resultados por  
apellido1 ascendentemente (apellido1:1) y luego por  
apellido2 descendientemente (apellido2:-1)

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca6"), "nombre" : "Concha", "apellido1" : "López", "apellido2" : "Arias" }
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" : "Pérez", "apellido2" : "Rodríguez" }
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "apellido1" : "Vallén", "apellido2" : "López" }
```

```
> db.ejemplo.find({nombre:{$regex:".*a$"}}).count()
```



Cuenta el número de  
resultados obtenidos

3

```
.pretty()
.sort()
.limit()
.skip()
.count()
.toArray()
```



# Manejo de BD

## Consultas en Bases de Datos

- Modificadores sobre consultas (II)

```
> db.ejemplo.find({nombre: {$regex: ".*a$"}}, {nombre: 1}).skip(3)
```



Ignora los 3  
primeros  
resultados

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca6"), "nombre" : "Concha" }
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca7"), "nombre" : "Juan" }
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana" }
```

```
> db.ejemplo.find({nombre: {$regex: ".*a$"}}, {nombre: 1}).skip(3).limit(1)
```

```
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca6"), "nombre" : "Concha" }
```



Ignora los 3 primeros  
resultados (skip) y, de los  
restantes, muestra solo uno  
(limit)

```
.pretty()
.sort()
.limit()
.skip()
.count()
.toArray()
```



# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (I)

Consulta que se usará para seleccionar los documentos a modificar

Opciones de comportamiento

`{ $multi:true|false }` → múltiples actualizaciones  
`{ $upsert{c1:v1,...} }` → actualiza y, si no existe lo inserta

```
> db.<collection_name>.update({<query>},{},{})
```

Modificadores de inserción

`{...}` → actualiza los datos (requiere inserción del documento entero)  
`{ $set{c1:v1,c2:v2,...} }` → actualiza sólo los campos indicados de un documento  
`{ $unset{c1:1,c2:1,...} }` → elimina uno (o varios) campos de los documentos.  
`{ $inc:{c1:<cantidad>} }` → incrementa el valor de un campo en una cantidad.

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (I)

```
> db.<collection_name>.update(<query>,{...})
```


```
db.ejemplo.find({ nombre: "Eduardo" })
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" :
23, "altura" : 1.85, "casado" : true, "fechaAlta" : ISODate("2022-04-
25T11:44:00.453Z") }
```

```
> db.ejemplo.update({nombre: "Eduardo"},{nombre: "Eduardo", edad : 24,
altura : 1.85, casado : true, fechaAlta : new Date()})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.ejemplo.find({ nombre: "Eduardo" })
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca4"), "nombre" : "Eduardo", "edad" :
24, "altura" : 1.85, "casado" : true, "fechaAlta" : ISODate("2022-04-
25T11:44:00.453Z") }
```



{...}



OBLIGA A  
INSERTAR EL  
DOCUMENTO ENTERO



# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (II)

```
> db.<collection_name>.update(<query>,{...},{upsert:true|false})
```

```
> db.ejemplo.update({nombre:"Miguel"},{nombre: "Miguel", edad : 42, altura :  
1.80, casado : false},{upsert:true})  
WriteResult({"nMatched" : 0,"nUpserted" : 1,"nModified" : 0,"_id" :  
ObjectId("62669449a85a751516c735e0")})
```

```
db.ejemplo.find({nombre:"Miguel"}).pretty()  
{  
  "_id" : ObjectId("62669449a85a751516c735e0"),  
  "nombre" : "Miguel",  
  "edad" : 42,  
  "altura" : 1.8,  
  "casado" : false  
}
```

{...}

{upsert:...}



# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (III)

```
> db.<collection_name>.update(<query>,{ $set:{f1:v1,f2:v2,...}})
```

```
db.ejemplo.find({ nombre: "María" })
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" : "Pérez", "apellido2" : "Rodríguez", "edad" : 32, "altura" : 1.74, "casado" : false, "fechaAlta" : ISODate("2022-04-25T11:44:08.062Z") }
```

```
> db.ejemplo.update({nombre: "María"},{ $set{edad:33, casado:true}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
db.ejemplo.find({ nombre: "María" })
{ "_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "apellido1" : "Pérez", "apellido2" : "Rodríguez", "edad" : 33, "altura" : 1.74, "casado": "true", "fechaAlta" : ISODate("2022-04-25T11:44:08.062Z") }
```



```
{ $set{...} }
```

MODIFICA SOLO  
LOS CAMPOS  
INDICADOS

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (IV)

```
> db.<collection_name>.update(<{<query>>},{<$set:{c1:v1,c2:v2,...}>},{multi:true})
```

```
db.ejemplo.find({ casado: true } ,{nombre:1,casado:1,numhijos:1})
{"_id" : ObjectId("6262a0ccb2a119d14cdadcfc"), "nombre" : "Eduardo", "casado":
"true"}
{"_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "casado":"true"}
```

```
> db.ejemplo.update({casado:true},{<$set{numhijos: 0}>},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
```

```
db.ejemplo.find({casado: true} ,{nombre:1,casado:1,numhijos:1})
{"_id" : ObjectId("6262a0ccb2a119d14cdadcfc"), "nombre" : "Eduardo", "casado":
"true", "numhijos":0}
{"_id" : ObjectId("6266bac3bd92fbd18b5b6ca5"), "nombre" : "María", "casado":
"true", "numhijos":0}
```



{<\$set{...}>}

{<\$multi:...>}

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (V)

```
> db.<collection_name>.update(<{<query>>},{$unset:{c1:1,c2:1,...}})
```

```
db.ejemplo.find({nombre:"Miguel"})
{"_id" : ObjectId("62669449a85a751516c735e0"), "nombre":"Miguel","edad":42,
"altura" : 1.8, "casado" : false}
```

```
> db.ejemplo.update({nombre:"Miguel"},{$unset:{casado:1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
db.ejemplo.find({nombre:"Miguel"})
{"_id" : ObjectId("62669449a85a751516c735e0"), "nombre":"Miguel","edad" : 42,
"altura" : 1.8, "casado" : false}
```



```
{$unset{...}}
```

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones básicas (V)

```
> db.<collection_name>.update({<query>},{ $inc:{v1:<cant>}})
```

```
db.ejemplo.find({nombre:"Miguel"})
{"_id" : ObjectId("62669449a85a751516c735e0"), "nombre":"Miguel","edad":42,
"altura" : 1.8, "casado" : false}
```

```
> db.ejemplo.update({nombre:"Miguel"},{$inc:{edad:12}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
db.ejemplo.find({nombre:"Miguel"})
{"_id" : ObjectId("62669449a85a751516c735e0"), "nombre":"Miguel","edad" : 54,
"altura" : 1.8}
```



{ \$inc {... } }

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (I)

```
> db.<collection_name>.update ({<query>},{})
```

DOT NOTATION

Modificadores de inserción

`{$set:{c1.<pos>:v1, c2.$:v2}}` → actualiza el elemento en una determinada posición del array  
`{$addToSet{...}}` → añade un elemento al final del array. Si ya existe NO se añade.  
`{$pop:{c1:1|-1}}` → elimina el primer (-1) o último elemento (1) del array.  
`{$pull:{c1:v1}}` → elimina el elemento del array con un determinado valor.  
`{$pullAll:{c1:[v1,v2,v3,...]}}` → elimina varios elementos del array.

# Manejo de BD


## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (II)

Posición dentro del array  
(índices empiezan en 0)



```
> db.<collection_name>.update(<query>,{ $set:{c1.pos:v1,c2.pos:v2,...}})
```



```
{ $set:{c1.pos:v1} }  
{ $set:{c1.$:v1} }
```

```
db.ejemplo.find({nombre:"Ana"},{nombre:1,email:1})  
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [  
  "avl@empresa.com", "avl@gmail.com" ] }  
  
> db.ejemplo.update({nombre:"Ana"},{ $set:{ "email.1":"ana.vallen.lopez@gmail.com" } })  
  
db.ejemplo.find({nombre:"Ana"})  
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [  
  "avl@empresa.com", "ana.vallen.lopez@gmail.com" ] }
```

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (II)


```
> db.<collection_name>.update(<query>,{ $set: {c1.pos:v1,c2.pos:v2,...}})
```

Posición dentro del array  
(índices empiezan en 0)



```
db.ejemplo.update({nombre:"Ana"},{$set:{email.5:"info@anavallen.es"}})
```

```
db.ejemplo.find({nombre:"Ana"})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"avl@empresa.com", "ana.vallen.lopez@gmail.com", null, null, null,
"info@anavallen.es" ] }
```



```
{ $set: {c1.pos:v1} }
{ $set: {c1.$:v1} }
```



SI POSICIÓN  
INSERCIÓN ES  
MAYOR QUE EL Nº  
ELEMENTOS ARRAY,  
SE RELLENAN CON  
NULL HASTA LLEGAR  
A LA POSICIÓN DE  
INSERCIÓN



# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (II)

Si se conoce el valor del elemento pero no la posición




```
> db.<collection_name>.update(<{query}>,{ $set: {c1.$:v1,c2.$:v2,...}})
```

```
> db.ejemplo.find({nombre:"Ana"}},{nombre:1,email:1})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"avl@empresa.com", "avl@gmail.com" ] }
```

```
> db.ejemplo.update({nombre:"Ana",email:"avl@empresa.com"},
{$set: {"email.$": "updated@empresa.com"}})
```

```
> db.ejemplo.find({nombre:"Ana"})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com" ] }
```



```
{ $set: {c1.pos:v1} }
{ $set: {c1.$:v1} }
```

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (II)

```
> db.<collection_name>.update (<query> , {$addToSet: {...}})
```



```
{$addToSet: {...}}
```

Opciones de comportamiento

c1:v1 → añade el elemento en la última posición del vector  
c1:{\$each:[v1,v2,...]} → inserta múltiples elementos en el vector

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (II)

```
> db.<collection_name>.update(<query>,{ $addToSet:{...}})
```



```
{ $addToSet:{...}}
```

c1:v1

```
> db.ejemplo.find({nombre:"Ana"},{nombre:1,email:1})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com" ] }

> db.ejemplo.update({nombre:"Ana"},{ $addToSet:{ "email":"new.email@empresa.com"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

> db.ejemplo.find({nombre:"Ana"})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com" ] }
```

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (III)



```
> db.<collection_name>.update(<query>,{ $addToSet: { c1: { $each: [v1,...] } } })
```

```
{ $addToSet: { ... } }
```

```
c1: { $each: [v1,...] }
```

```
> db.ejemplo.find({ nombre: "Ana" }, { nombre: 1, email: 1 })
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com" ] }
```

```
> db.ejemplo.update({ nombre: "Ana" }, { $addToSet: { "email": { $each: ["new.email@empresa.com",
"latest@email.com"] } } })
```

```
> db.ejemplo.find({ nombre: "Ana" })
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com",
"latest@email.com" ] }
```



No se inserta de nuevo  
porque ya existe

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (IV)



`{ $pop: { c1: 1 | -1 } }`

```
> db.<collection_name>.update (<query> , { $pop: { c1: v1 } })
```

```
> db.ejemplo.find({nombre:"Ana"} , {nombre:1, email:1})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com",
"latest@email.com" ] }
```

```
> db.ejemplo.update ({nombre:"Ana"} , { $pop: { email: 1 } })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.ejemplo.find({nombre:"Ana"})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com",
"latest@email.com" ] }
```

# Manejo de BD

## Actualizaciones en Bases de Datos


- Operaciones avanzadas: arrays (V)

```
> db.<collection_name>.update(<{query}>,{ $pull: {c1:v1}})
```

```
> db.ejemplo.find({nombre:"Ana"}},{nombre:1,email:1})
"_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com" ]

> db.ejemplo.update({nombre:"Ana"},{ $pull: {email:"updated@empresa.com"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.ejemplo.find({nombre:"Ana"})
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"updated@empresa.com", "ana.vallen.lopez@gmail.com", "new.email@empresa.com" ] }
```



```
{ $pull: {c1:v1} }
```

# Manejo de BD

## Actualizaciones en Bases de Datos

- Operaciones avanzadas: arrays (VI)



```
{ $pullAll: { c1: [ ... ] } }
```

```
> db.<collection_name>.update ( {<query> }, { $pullAll: { c1: [v1,v2,...] } } )
```

```
> db.ejemplo.find( { nombre: "Ana" }, { nombre: 1, email: 1 } )
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"ana.vallen.lopez@gmail.com", "new.email@empresa.com" ] }

> db.ejemplo.update ( { nombre: "Ana" }, { $pullAll: { email: [ "ana.vallen.lopez@gmail.com",
"new.email@empresa.com", "new.email@empresa.com" ] } } )
WriteResult( { "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 } )

> db.ejemplo.find( { nombre: "Ana" } )
{ "_id" : ObjectId("62679dc3bd92fbd18b5b6ca8"), "nombre" : "Ana", "email" : [
"ana.vallen.lopez@gmail.com", "new.email@empresa.com", "new.email@empresa.com" ] }
```

# Colecciones

## Uniones entre colecciones

- MongoDB **no** es un SGBDR → ineficaz gestionando relaciones entre colecciones (tablas)
- MongoDB es compatible con JOINS entre colecciones (a partir de la versión 3.2).
  - Se recomienda limitar su uso.

Permite hacer consultas  
sobre múltiples documentos y  
colecciones



Realiza un  
JOIN entre  
colecciones



```
> db.<collection_name>.aggregate({
  $lookup: {
    from:<collection_to_join>,
    localField:<collection_name_field>,
    foreignField:<collection_to_join_field>,
    as:<output_name>
  }
})
```



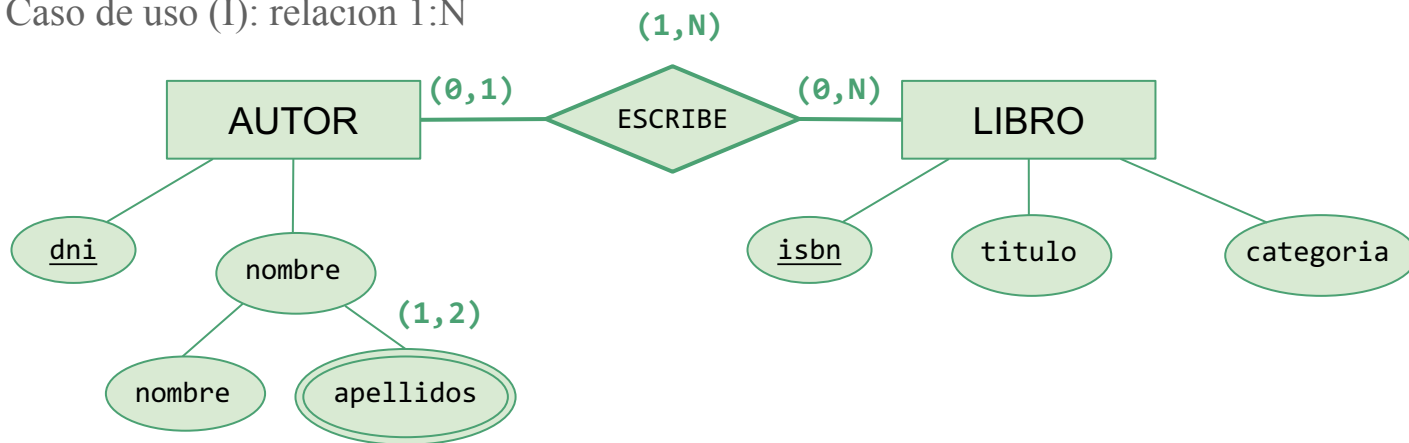
Parámetros para  
hacer el JOIN



# Colecciones

## Uniones entre colecciones

- Caso de uso (I): relacion 1:N



# Colecciones

## Uniones entre colecciones

- Caso de uso (I)

```
db.autores.insert([
... {
...   _id: '54963125B',
...   nombre: { pila: 'Brad', apellidos: 'Dayley' },
...   edad: 54
... }
]);
```

```
db.libros.insert([
... {
...   _id: '978-1839210648',
...   titulo: 'NoSQL with MongoDB in 24H',
...   categoria: 'Tecnología y Ordenadores',
...   autor: '54963125B'
... },
... {
...   _id: '978-0596158101',
...   titulo: 'Node.js, MongoDB, and AngularJS',
...   categoria: 'Tecnología y Ordenadores',
...   autor: '54963125B'
... }
]);
```

LIBROS QUE HA  
ESCRITO CADA AUTOR

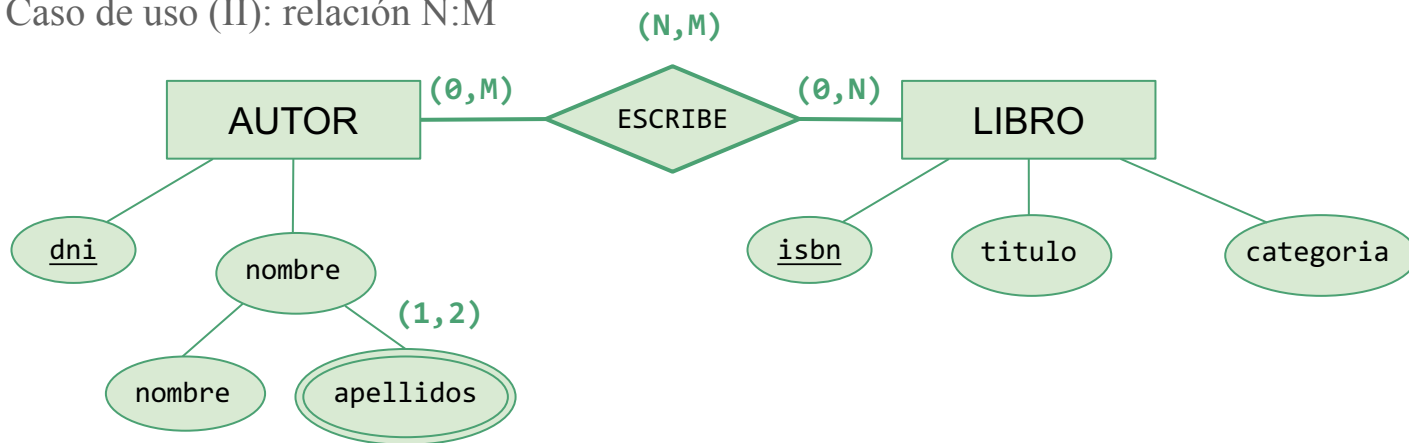
```
> db.autores.aggregate({
  $lookup: {
    from: "libros",
    localField: "_id",
    foreignField: "autor",
    as: "libros_escritos"
  }
})
```

```
{ "_id" : "54963125B",
  "nombre" : { "pila" : "Brad", "apellidos" : "Dayley" },
  "edad" : 54,
  "libros_escritos" : [ { "_id" : "978-1839210648", "titulo" : "NoSQL
with MongoDB in 24H", "categoria" : "Tecnología y Ordenadores",
"autor" : "54963125B" }, { "_id" : "978-0596158101", "titulo" :
"Node.js, MongoDB, and AngularJS", "categoria" : "Tecnología y
Ordenadores", "autor" : "54963125B" } ] }
```

# Colecciones

## Uniones entre colecciones

- Caso de uso (II): relación N:M



# Colecciones

## Uniones entre colecciones

- Caso de uso (I)

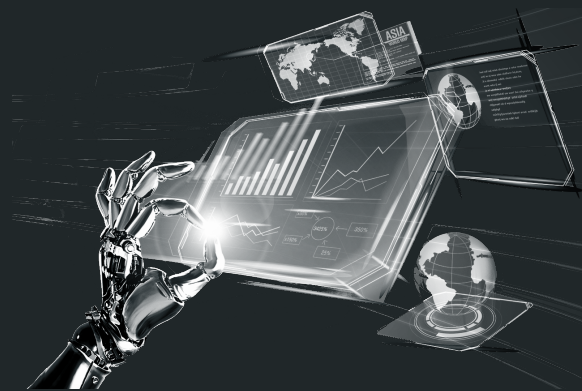
```
db.autores.insert([
... {
...   _id: '54963125B',
...   nombre: { pila: 'Juned', apellidos: 'Ahsan' },
...   edad: 61
... }, {
...   _id: '24915478Y',
...   nombre: { pila: 'Liviu', apellidos: 'Nedov' },
... } ]]);
```

```
db.libros.insert([
... {
...   _id: '978-1839210648',
...   titulo: 'NoSQL with MongoDB in 24H',
...   categoria: 'Tecnología y Ordenadores',
...   autor: '54963125B'
... },
... {
...   _id: '978-0596158101',
...   titulo: 'Mongo DB Fundamentals',
...   categoria: 'Tecnología y Ordenadores',
...   autor: [ '54963125B', '24915478Y' ]
... } ]]);
```

LIBROS QUE HA  
ESCRITO CADA AUTOR

```
> db.autores.aggregate({
  $lookup: {
    from: "libros",
    localField: "_id",
    foreignField: "autor",
    as: "libros_escritos"
  }
})
```

```
{ "_id" : "54963125B", "nombre" : { "pila" : "Juned", "apellidos" :
  "Ahsan" }, "edad" : 61,
  "libros_escritos" : [ { "_id" : "978-1839210648", "titulo" : "NoSQL
  with MongoDB in 24H", "categoria" : "Tecnología y Ordenadores",
  "autor" : "54963125B" }, { "_id" : "978-0596158101", "titulo" :
  "Mongo DB Fundamentals", "categoria" : "Tecnología y Ordenadores",
  "autor" : [ "54963125B", "24915478Y" ] } ] }
```



# Gestión de Datos para Robótica

## T2b - MongoDB

Álvaro Vázquez Álvarez  
Departamento de Electrónica e Computación

✉ [alvaro.vazquez@usc.es](mailto:alvaro.vazquez@usc.es)

📍 Pabellón III - Despacho 4

Curso 2023-2024