

# Proyecto 1<sup>a</sup> evaluación

---

MONGODB

## Contenido

1.)	Introducción .....	3
2.)	Inserciones .....	3
3.)	Operadores de comparación usados. ....	4
3.1)	\$eq:.....	4
3.2)	\$gt: .....	4
3.3)	\$gte:.....	4
3.4)	\$lt: .....	4
3.5)	\$lte: .....	4
3.6)	\$in:.....	4
3.7)	\$ne:.....	4
3.8)	\$nin:.....	4
4.)	Operadores lógicos usados. ....	5
4.1)	\$and:.....	5
4.2)	\$or: .....	5
4.3)	\$not: .....	5
4.4)	\$nor: .....	5
5.)	Operadores de elemento.....	6
5.1)	\$exist: .....	6
6.)	Operadores de evaluación.....	6
6.1)	\$regex:.....	6
7.)	Operadores de Arrays.....	6
7.1)	\$all: .....	6
7.2)	\$elemMatch:.....	6
7.3)	\$size:.....	6
8.)	Aportación personal. ....	7
8.1)	.? en regex: .....	7
8.2)	Aggregate:.....	7
8.3)	\$group: .....	7
8.4)	\$num: .....	7
8.5)	\$multiply:.....	7
8.6)	\$subtract:.....	7
8.7)	\$avg: .....	7

8.8) \$project: .....	7
8.9) \$size:.....	8
8.10) \$match:.....	8
8.10) \$max: .....	8
8.11) \$min:.....	8
8.12) update: .....	8
8.13) \$set: .....	8
8.14) \$addToSet: .....	8

## 1.) Introducción

En este proyecto he utilizado una base de datos de una empresa que se dedica a vender exclusivamente portátiles.

En las inserciones he usado varios tipos de datos, tales como: Enteros, cadenas alfanuméricas, booleanos, fechas, arrays de documentos, arrays de enteros, arrays de cadenas alfanuméricas, arrays dentro de arrays, etc...

En las propias consultas a la base de datos he usado varios operadores como \$and, \$or, \$nor, \$not, \$in, etc...

## 2.) Inserciones

Hay 2 colecciones en este proyecto:

- La colección portátiles:

```
db.portatiles.insertMany([
  {_id: 1,
    producto: "TUF Gaming F15 FX506HCB-HN200",
    marca: "Asus",
    precioIVA: 1089,
    precio: 825.62,
    socket: "Intel",
    procesador: "i5-11400H",
    ramGB: 16,
    almacenamientoGB: [{SSD: 512, HDD: 512, total: 1024}],
    grafica: "RTX 3050",
    conexiones: ["1xJack", "1xHDMI", "1xRJ45", "1xThunderbolt", "3xUSB", "1xUSBC"],
    tamañosCM: [35.9, 25.6, 2.28],
    SO: false,
    peso: [{kg: 2.30, lbs: 5.07}],
    stock: 4,
  },
],
```

Además, puede traer otro campo llamado "tactil". En caso de que sea true, si trae táctil, en caso de false, no trae táctil.

- La colección pedidos:

```
db.pedidos.insertMany([
  {_id: 1,
    productos: [{"Katana GF66 11UC-045XES", "Blade 14"}, [8, 26]],
    precio: 4173.62,
    localidad: "La Campana",
    fechas: [new Date("2021-10-22"), new Date("2021-10-25")],
    retraso: false,
  },
],
```

Nombre de los productos del pedido y sus respectivos ID

Fecha de envío / Fecha de entrega

### 3.) Operadores de comparación usados.

#### 3.1) \$eq:

Compara documentos donde el valor es igual al indicado:

```
{<campo>: { $eq: <valor> } }
```

#### 3.2) \$gt:

Compara aquellos documentos donde el valor es mayor al indicado:

```
{<campo>: { $gt: <valor> } }
```

#### 3.3) \$gte:

Compara aquellos documentos donde el valor es mayor o igual al indicado:

```
{<campo>: { $gte: <valor> } }
```

#### 3.4) \$lt:

Compara aquellos documentos donde el valor es menor al indicado:

```
{<campo>: { $lt: <valor> } }
```

#### 3.5) \$lte:

Compara aquellos documentos donde el valor es menor o igual al indicado:

```
{<campo>: { $lte: <valor> } }
```

#### 3.6) \$in:

Selecciona los documentos donde el valor de un campo especificado es igual a los valores especificados dentro de una Array:

```
{ <campo>: { $in: [<valor1>, <valor2>, ... <valorN> ] } }
```

#### 3.7) \$ne:

Selecciona los documentos que son la negación del valor indicado dentro del campo especificado.

```
{<campo>: { $ne: <valor> } }
```

#### 3.8) \$nin:

Selecciona los documentos donde el valor de un campo especificado es la negación a los valores especificados dentro de una Array:

```
{ <campo>: { $nin: [<valor1>, <valor2>, ... <valorN> ] } }
```

## 4.) Operadores lógicos usados.

### 4.1) \$and:

Realiza una lógica donde ambos valores indicados tienen que ser cumplidos para que la propia condición AND se cumpla, y selecciona todos los documentos que cumplen todas las expresiones.

```
{ $and: [ { <expresion1> }, { <expresion2> }, ..., { <expresionN> } ] }
```

### 4.2) \$or:

Realiza una lógica donde uno de los dos valores indicados tienen que ser cumplidos para que la propia condición OR se cumpla, y selecciona todos los documentos que cumplen todas las expresiones.

```
{ $or: [ { <expresion1> }, { <expresion2> }, ..., { <expresionN> } ] }
```

### 4.3) \$not:

Realiza una lógica donde uno de los dos valores indicados NO tienen que ser cumplidos para que la propia condición NOT se cumpla, y selecciona todos los documentos que cumplen todas las expresiones.

```
{ $not: [ { <expresion1> }, { <expresion2> }, ..., { <expresionN> } ] }
```

### 4.4) \$nor:

Realiza una lógica donde uno de los dos valores indicados tienen que ser cumplidos para que la propia condición NOR se cumpla, y selecciona todos los documentos que cumplen todas las expresiones.

```
{ $nor: [ { <expresion1> }, { <expresion2> }, ..., { <expresionN> } ] }
```

## 5.) Operadores de elemento.

### 5.1) \$exist:

Busca un campo dentro de un documento: en caso de que sea true y el campo exista devolverá el campo indicado, en caso de que sea false no hace nada.

```
{<campo>: { $exist: <valor booleano> } }
```

## 6.) Operadores de evaluación.

### 6.1) \$regex:

Proporciona opciones de expresiones regulares para campos alfanuméricos para patrones coincidentes.

```
{<campo>: { $regex: /<valor>/<opción> } }
```

## 7.) Operadores de Arrays.

### 7.1) \$all:

Este operador selecciona los documentos donde el valor de un campo es un array que contiene todos los elementos especificados.

```
{ <campo>: { $all: [ <value1> , <value2> ... ] } }
```

### 7.2) \$elemMatch:

Este operador compara documentos que contienen un campo de array con al menos un elemento que coincide con todos los criterios de consulta especificados.

```
{ <campo>: { $elemMatch: [ <comparador1> ,  
<comparador2> ... ] } }
```

### 7.3) \$size:

Este operador devuelve documentos que coinciden con un campo de arrays con el tamaño que le pasemos.

```
{ <campo>: { $size: <valor numérico> } }
```

## 8.) Aportación personal.

### 8.1) .? en regex:

```
db.portatiles.find({
  producto: { $regex: /\.?G733QR-K4008T/i }
})
```

### 8.2) Aggregate:

Procesan varios documentos y devuelven resultados procesados. Se pueden utilizar para:

- Agrupar valores de varios documentos diferentes.
- Realizar operaciones para devolver un resultado único.
- Analizar cambios a la larga.

### 8.3) \$group:

Agrupar los documentos de entrada por la expresión indicada mediante la “\_id” y para cada agrupación distinta, genera un documento.

```
db.portatiles.aggregate([
  { $group:
    { _id: "$marca",
      total:
        { $avg: "$precio" }
    }
  }
])
```

### 8.4) \$sum:

Calcula y devuelve la suma de valores numéricos.

```
{ $sum: <expression> }
```

### 8.5) \$multiply:

Calcula y devuelve la multiplicación de valores numéricos.

```
{ $multiply: <expression> }
```

### 8.6) \$subtract:

Calcula y devuelve la resta de valores numéricos.

```
{ $subtract: [ <expression1>, <expression2> ] }
```

### 8.7) \$avg:

Calcula y devuelve el promedio de valores numéricos.

```
{ $avg: [ <expression1>, <expression2> ] }
```

### 8.8) \$project:

Pasa los documentos con los campos solicitados a la siguiente



etapa del proceso. Los campos especificados pueden ser campos existentes de los documentos de entrada o campos recién calculados.

```
{ $project: { <especificaciones> } }
```

#### 8.9) \$size:

Hace coincidir cualquier array con el número de elementos especificados por el argumento.

```
{ <campo>: { $size: <valor numérico> } }
```

#### 8.10) \$match:

Hace coincidir cualquier array con el número de elementos especificados por el argumento.

```
{ $match: { <query> } }
```

#### 8.10) \$max:

Devuelve el valor máximo.

```
{ $max: <expression> }
```

#### 8.11) \$min:

Devuelve el valor mínimo.

```
{ $min: <expression> }
```

#### 8.12) update:

Sirve para añadir campos y valores a un documento ya existente.

#### 8.13) \$set:

Se combina con el update para añadir campos sin sobrescribir nada a un documento ya existente.

```
db.portatiles.updateOne({_id: 22}, {$set: {"tactil": true}})
```

#### 8.14) \$addToSet:

Se puede usar tanto en el aggregate para agrupar campos como en el update con la misma función que el \$set.

```
db.portatiles.updateOne(
  {
    _id: 22,
    $addToSet: {tactil: true}
  }
)
```

```
db.portatiles.aggregate([
  {$group:
    {
      _id: "$marca",
      nombre:
        {$addToSet: "$producto"}}
  })
```