MongoDB.

Contenido

[Recursos útiles 1](#_Toc203423611)

[Objetivo de la práctica 2](#_Toc203423612)

[Instalación MongoDB y herramientas asociadas 2](#_Toc203423613)

[Instalación servidor MongoDB como Docker 2](#_Toc203423614)

[MongoDB Compass GUI 2](#_Toc203423615)

[Tutorial MongoDB 4](#_Toc203423616)

[Pasos básicos descritos por el docente. 4](#_Toc203423617)

[Ejercicios sobre bases de datos de ejemplo en MongoAtlas. 8](#_Toc203423618)

[Ejercicios de consultas 8](#_Toc203423619)

[Ejercicios de proyección 8](#_Toc203423620)

[Ejercicios de actualización y borrado 8](#_Toc203423621)

[Ejercicios de agregación 9](#_Toc203423622)

[Trabajando desde Python 9](#_Toc203423623)

[Creación de la base de datos 9](#_Toc203423624)

[Programa Python 9](#_Toc203423625)

[Creación de la colección completa 9](#_Toc203423626)

[Consultas y actualizaciones 9](#_Toc203423627)

[Entregables de la práctica 10](#_Toc203423628)

# Recursos útiles

Recuerde el directorio de recursos compartidos disponible en:

<https://nascorformacion0-my.sharepoint.com/:f:/g/personal/juan_pinuela_docente_nascorformacion_com/EtGSBITnqFpJp6-XpG6REOQBFycP_G9zgcB89jxPymjV2Q?e=IBIEER>

Para esta práctica disponemos de una carpeta con nombre “**mongodb”** donde podemos encontrar todos los ficheros utilizados y creados en este guion.

Recurso recomendado: <https://youtu.be/rcZlFmioTkE?si=VUc8vBl9gZwDt8Fr>

# Objetivo de la práctica

Esta práctica tiene como objetivo introducir al estudiante en los conceptos fundamentales de MongoDB desde Compass GUI con una base de datos local, luego practicar con la base de datos de restaurantes disponible en Atlas y finalmente se propone un ejercicio con Python para simular un e-commerce que integraremos más adelante con otro tipo de base de datos NoSQL (en concreto con Neo4j y con Elasticsearch) a lo largo del curso.

# Instalación MongoDB y herramientas asociadas

## Instalación servidor MongoDB como Docker

Creamos la carpeta “C:\Docker\mongodb” donde copiamos el siguiente docker-compose.yml, que podemos lanzar sin más que ejecutando desde la línea de comandos:

>> docker-compose up -d

version: '3.8'

services:

  mongo:

    image: mongo

    container\_name: mongodb

    restart: always

    ports:

      - "27017:27017"

    environment:

      MONGO\_INITDB\_ROOT\_USERNAME: root

      MONGO\_INITDB\_ROOT\_PASSWORD: example

## MongoDB Compass GUI

Descargamos MongoDB Compass de:

<https://www.mongodb.com/try/download/compass>

Podemos hacer una conexión local primero con el Docker utilizando la siguiente cadena de conexión: **mongodb://root:example@localhost:27017**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Creación de cuenta y carga de datos ejemplo en MongoDB Atlas

Creamos una cuenta y entramos en: <https://www.mongodb.com/es/atlas>. Importante seleccionar opciones como las mostradas en las siguientes imágenes (fundamental copiar usuario y password por un lado y luego cadena de conexión).

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Cargamos la base de datos de restaurantes para practicar:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.**

Y en nuestro MongoDB compass seleccionamos la conexión con la cadena copiada para acceder a la base de datos de restaurantes (la cadena es algo parecido a: mongodb+srv://japinuela73:xxxxxx@cluster0.k9abhla.mongodb.net/).

Tendremos en este momento dos conexiones posibles. Una local y otra con Atlas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

# Tutorial MongoDB

## Pasos básicos descritos por el docente.

1. **Creación de la base de datos “blogs”, colección, “posts”**
2. **Insertamos** 1 documento y luego 3 a continuación:

{

"title": "Post Title 1",

"body": "Body of post.",

"category": "News",

"likes": 1,

"tags": ["news", "events"],

"date": "2025-07-13"

}

[

{

"title": "Post Title 2",

"body": "Body of post.",

"category": "Event",

"likes": 2,

"tags": ["news", "events"],

"date": "2025-07-12"

},

{

"title": "Post Title 3",

"body": "Body of post.",

"category": "Technology",

"likes": 3,

"tags": ["news", "events"],

"date": "2025-07-12"

},

{

"title": "Post Title 4",

"body": "Body of post.",

"category": "Event",

"likes": 4,

"tags": ["news", "events"],

"date": "2025-07-10"

}

]

1. **Find**: {category: "Technology"}, {title:'Post Title 4'}
2. **Proyección**: {title: 1, date: 1} (el campo “\_id” siempre está incluido por defecto).

Además de proyección como opciones de un filtrado tenemos también:

* Ordenación: {"likes": 1}
* *Collation*. Esta opción después de un filtro permite definir definir idioma, sensibilidad y reglas. Por ejemplo:

{

"locale": "es",

"strength": 1

}

| **strength** | **Comparación distingue...** |
| --- | --- |
| 1 | Solo base de caracteres (ignora mayúsculas y tildes). |
| 2 | Distingue tildes, ignora mayúsculas. |
| 3 | Distingue tildes y mayúsculas (como por defecto). |
| 4 | Distingue variantes de idénticas letras. |

1. **Operadores lógicos**

Existen numerosos operadores de consulta que se pueden usar para comparar y referenciar campos de documentos.

Comparación:

$eq: Valores iguales

$ne: Valores distintos

$gt: Valor mayor que otro

$gte: Valor mayor o igual que otro

$lt: Valor menor que otro

$lte: Valor menor o igual que otro

$in: El valor coincide dentro de una matriz

Ejemplo para caja de filtro:

**{likes:{"$gt":25}}**

**{"hobbies": {"$in": ["Correr"]}}**

**{"hobbies": {"$in": ["Correr", "Bares"]}}**

**{“contacto”: {“$match”:**

Lógico

$and: Devuelve documentos donde ambas consultas coinciden

$or: Devuelve documentos donde alguna de las consultas coincide

$nor: Devuelve documentos donde ambas consultas no coinciden

$not: Devuelve documentos donde la consulta no coincide

Ejemplo para caja de filtro:

**{"$and": [{"likes":{"$gte":10}},{"likes":{"$lte":80}}]}**

**{"likes":{"$not":{"$eq":100}}}**

Evaluación

Los siguientes operadores ayudan a evaluar documentos.

$regex: Permite el uso de expresiones regulares al evaluar valores de campo

$text: Realiza una búsqueda de texto

$where: Utiliza una expresión de JavaScript para buscar documentos

Ejemplo:

**{"contacto.movil": { "$regex": "^651" }}**

1. **Operadores de actualización**.

Puedo aplicar un filtro primero con : {category: "Technology”} y luego Update.

**{**

**$set: {**

**"likes": 30 },**

**}**

Sin filtro actualizo todos los documentos pudiendo usar funciones como $inc para incrementar el valor de un campo en todos los documentos a la vez.

{

$set: {

$inc: { likes: 1 }

},

}

Otros operadores de actualización son:

$currentDate: Establece el valor del campo a la fecha actual

$rename: Renombra el campo

$unset: Elimina el campo del documento

Y para el caso de arrays tenemos:

$addToSet: Agrega elementos distintos a una matriz

$pop: Elimina el primer o el último elemento de una matriz

$pull: Elimina todos los elementos de una matriz que coinciden con la consulta

$push: Agrega un elemento a una matriz

1. **Aggregations**. Las operaciones de agregación permiten agrupar, ordenar, realizar cálculos, analizar datos y mucho más. Pueden tener una o más etapas. El orden de estas etapas es importante. Cada etapa actúa sobre los resultados de la etapa anterior.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Otro ejemplo con $project (las referencias a campos deben ir precedidas del símbolo $ para indicar que es un valor de campo, no un literal).

{

"nombre": 1,

"edad": 1,

"categoria": {

"$cond": {

"if": {"$lt":["$edad", 30]},

"then": "joven",

"else": "adulto"

}

}

}

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Y para sumar toda las edades.

{

\_id: null,

sumaEdad: {

$sum: "$edad"

}}

Y para contar

{

\_id: null,

cuenta: {

$sum: 1

}

}}

## Ejercicios sobre bases de datos de ejemplo en MongoAtlas.

### Ejercicios de consultas

Trabajamos sobre la colección restaurants

1. Todos los documentos de la colección restaurants
2. Los que sean de cocina griega (Greek)
3. Los que sean de cocina griega y tengan una nota (grade) de B
4. Los que hayan tenido una valoración (score) de al menos 30 puntos
5. Los que sean de cocina Italina o Mexican y estén en Manhattan (borough)
6. Los peores (con algún score menor de 5) de los de comida italiana

### Ejercicios de proyección

Trabajamos sobre la colección restaurants

1. Tipos de cocina
2. ¿Cuáles son los nombres de los restaurantes que ofrecen comida italiana?
3. ¿Cuál es el nombre, la calle y el barrio de los peores (con algún score menor de 5) de los de comida italiana?

### Ejercicios de actualización y borrado

Vamos a trabajar ahora con la colección **movies** de la BD **sample\_mflix**

1. Actualiza el campo year de la pelicula de título "Civilization" a "1985"
2. Incrementar en 1 el número de comentarios (num\_mflix\_comments) de todas las películas del tipo (type) "movie"
3. Actualiza el campo cast de la película "Civilization" añadiéndole el actor "Perico de los Palotes"
4. Actualiza el campo year de las peliculas con duración (runtime) mayor a 100 para que sea 100 (es decir, no habrá ninguna duración mayor que 100)
5. Actualiza todas las películas cuya duración (runtime) sea de 100, para que no tengan el género (genres) "Action". Es decir, quitarle el género Action del array.
6. Borra las que tengan una puntuación (rating) en imdb menor a 3.4
7. Borra las películas de entre los años 1950 y 1970 y que tengan menos de 80 votos en imdb

### Ejercicios de agregación

Seguimos con la colección movies de la BD sample\_mflix

1. Obtener la cantidad total de películas en la colección
2. Calcular el promedio de duración de todas las películas
3. Contar cuántas películas hay por tipo (type)
4. Dime el título de las 5 películas con mayor duración
5. Calcular la puntuación promedio de imdb, por tipo de pelicula
6. Dime el número de comentarios de películas, según el año (year)

# Trabajando desde Python

## Creación de la base de datos

1. Crea una base de datos llamada **catalogo** y una colección **productos**.
2. Inserta manualmente algunos productos en la colección.
3. Elimina los productos insertados

## Programa Python

### Creación de la colección completa

1. Crea un fichero Python que inserte los productos disponibles en "productos.json" en nuestra base de datos

### Consultas y actualizaciones

1. Imprime en pantalla los productos (toda su información) con stock menor que 10
2. Crea un método **comprar** que reciba como parámetro el nombre del producto y simule el efecto de una compra en el stock con un $inc igual a -1.

Observa la opción “export to languague” en Compass GUI, donde podemos copiar el código Python para hacer un pipeline o una simple query (botón de código fuente justo al lado del botón “Find”).

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# Entregables de la práctica

El alumno deberá entregar las siguientes evidencias al finalizar la práctica:

1. Trabajo con MongoDB Atlas (parte gráfica desde Compass)

* Capturas de pantalla de las consultas realizadas sobre la colección "restaurants" en MongoDB Atlas
* Capturas de las actualizaciones, borrados y agregraciones básicas realizados en la colección movies de la base de datos "sample\\_mflix".

1. Desarrollo en Python, Fichero Python (.py) con el desarrollo del ejercicio correctamente comentado y documentado, indicando:

* Conexión a MongoDB en Docker local.
* Pasos principales realizados.