# **📘 Trabajo Práctico: MongoDB con Docker y Mongoose**

## **🎯 Objetivo general**

Que los estudiantes comprendan los conceptos fundamentales de MongoDB y Mongoose, y sean capaces de utilizar Docker para levantar una base de datos MongoDB y conectarse a ella desde una aplicación Node.js utilizando Mongoose.

## **🧠 Parte Teórica (Responder con sus propias palabras)**

**1. ¿Qué es MongoDB? ¿En qué se diferencia de una base de datos relacional como MySQL o PostgreSQL?**

MongoDB es una base de datos NoSQL orientada a documentos, lo que significa que almacena la información en documentos tipo JSON (en realidad BSON, que extiende JSON con más tipos de datos).

**Flexibilidad de esquema:** No requiere esquemas predefinidos, permitiendo almacenar documentos con estructuras variadas dentro de la misma colección.

**Escalabilidad horizontal:** Facilita el escalado mediante particionamiento (sharding) sin la rigidez de las bases de datos relacionales.

**Modelo de datos:** A diferencia de MySQL o PostgreSQL, que utilizan tablas, relaciones y esquemas estrictos, MongoDB maneja datos de forma más natural y ágil para ciertas aplicaciones, como aquellas con datos semiestructurados o en constante evolución.

**2. ¿Qué es una colección en MongoDB? ¿Y qué es un documento? Dibuja un ejemplo básico.**

****Colección:**** Es el equivalente a una tabla en bases de datos relacionales. Es un conjunto de documentos que, por lo general, comparten una estructura similar. Ej;

{

"\_id": "60d5f9a2f8a4e3b6d1e2c3f4",

"nombre": "Juan",

"edad": 30,

"ciudad": "Madrid"

}

**Documento:** Es la unidad básica de almacenamiento en MongoDB, representado en formato BSON/JSON, que puede contener datos anidados, arrays y otros tipos complejos.

Campos

MongoDB

Base de datos

Colecciones

Documentos

**3. ¿Qué es Mongoose? ¿Para qué se utiliza en una app de Node.js?**

Mongoose es una biblioteca ODM (Object Data Modeling) para Node.js que facilita la interacción con MongoDB.

**Gestión de esquemas:** Permite definir esquemas para documentos, asegurando consistencia y validación de los datos.

**Modelado y abstracción:** Proporciona modelos basados en esquemas que simplifican la creación, lectura, actualización y eliminación (CRUD) de documentos.

**Middleware y hooks:** Facilita la implementación de lógica adicional antes o después de las operaciones de la base de datos.

**4. ¿Qué es un esquema en Mongoose? ¿Y qué es un modelo? ¿Cuál es la relación entre ambos?**

****Esquema:**** Es la definición de la estructura de un documento, que incluye los campos, tipos de datos, valores predeterminados y validaciones.

**Modelo:** Es una clase que se crea a partir de un esquema y se utiliza para interactuar con la colección en MongoDB.

El modelo se basa en el esquema; en otras palabras, el esquema define la forma de los datos y el modelo implementa métodos y operaciones sobre esos datos. Cada vez que se crea un documento a través del modelo, este se adhiere a la estructura definida en el esquema.

**5. ¿Qué métodos de Mongoose se pueden usar para:**

**Insertar un nuevo documento:**

* Model.create(doc)
* Crear una instancia con new Model(doc) y luego llamar a .save()

**Buscar documentos:**

* Model.find(criterio) para múltiples documentos
* Model.findOne(criterio) para un solo documento
* Model.findById(id) para buscar por identificador único

**Actualizar un documento:**

* Model.updateOne(criterio, actualización)
* Model.findByIdAndUpdate(id, actualización)
* Model.updateMany(criterio, actualización) para actualizaciones en masa

**Eliminar un documento:**

* Model.deleteOne(criterio)
* Model.findByIdAndDelete(id)
* Model.deleteMany(criterio) para eliminar varios documentos a la vez

## **🧪 Parte Práctica (Completar)**

### **Consigna**

Vas a crear una pequeña aplicación en Node.js que se conecta a una base de datos MongoDB corriendo en Docker. Solo te damos una base para que completes el código y pongas a prueba lo aprendido.

### **Paso 1: Levantar MongoDB con Docker**

1. Crear un archivo docker-compose.yml y completarlo para que:
   * Se use la imagen oficial de MongoDB
   * Exponga el puerto 27017
   * Mantenga los datos persistentes usando volúmenes

version: "3.8"

services:

mongodb:

image: mongo:latest

restart: always

environment:

MONGO\_INITDB\_ROOT\_USERNAME: ${MONGO\_USER}

MONGO\_INITDB\_ROOT\_PASSWORD: ${MONGO\_PASS}

volumes:

- ./mongo:/data/db

ports:

- \*\_\_\_\_\_\_\* completar

1. Levantar el contenedor con el comando correspondiente.
2. Verificar en Docker desktop que este corriendo la bbdd

### **Paso 2: Crear la app en Node.js**

1. Crear un nuevo proyecto e instalar las siguientes dependencias:

* npm init -y
* npm i nodemon --save -dev
* npm i express mongoose body-parser dotenv

1. **Requiere los módulos necesarios**

* Importá los módulos express y mongoose.
* Inicializá una instancia de Express.
* Agregá el middleware para interpretar JSON.

1. **Conectate a MongoDB**

* Usá mongoose.connect() para conectarse a la base de datos.
* Agregá un .then() y .catch() para mostrar mensajes de conexión exitosa o error.

1. **Define el modelo de datos**

* Usá mongoose.Schema para definir el esquema de un **usuario** con los campos:  
  + nombre (tipo String)
  + edad (tipo Number)
  + email (tipo String)
* Usá mongoose.model() para crear el modelo.

1. **Crea la ruta GET /usuarios**

* Usá app.get() para definir una ruta que devuelva todos los usuarios almacenados en la base de datos.
* Usá Usuario.find() para obtener todos los documentos.
* Devolvé los resultados como JSON.

1. **Crea la ruta POST /usuarios**

* Usá app.post() para definir una ruta que reciba un objeto JSON con datos de un nuevo usuario.
* Usá el modelo para crear y guardar ese nuevo usuario.
* Devolvé como respuesta el objeto guardado.

1. **Iniciá el servidor**

* Usá app.listen() para levantar el servidor en el puerto 3000.
* Mostrá un mensaje por consola indicando que el servidor está corriendo.