**Clase 16:** Integración de Node.js/Express.js con Bases de Datos

Objetivos de la clase

* Aprender a integrar Node.js/Express.js con bases de datos utilizando MongoDB como ejemplo.
* Comprender los conceptos básicos de MongoDB y su conexión con Node.js.
* Realizar operaciones básicas de CRUD (Crear, Leer, Actualizar, Eliminar) en una base de datos MongoDB desde una aplicación Node.js/Express.js.

**Preguntas para orientar la lectura**

* *¿Qué diferencias clave existen entre una base de datos NoSQL como MongoDB y una base de datos relacional tradicional?*
* *¿Cómo se realiza la conexión entre una aplicación Node.js/Express.js y una base de datos MongoDB?*
* *¿Cuáles son los métodos básicos de CRUD y cómo se utilizan para realizar operaciones de Crear, Leer, Actualizar y Eliminar en MongoDB?*
* *¿Cómo puedes manejar errores y eventos durante la conexión y operación con MongoDB en una aplicación Node.js/Express.js?*

**Introducción a MongoDB**

*Definiciones*

MongoDB es una base de datos NoSQL orientada a documentos que almacena datos en formato JSON-like, lo que permite una estructura flexible y escalable.

En lugar de usar tablas y filas como en las bases de datos relacionales, MongoDB utiliza colecciones y documentos.

*Instalación y configuración*

**Instalación:** podemos descargar MongoDB desde <https://www.mongodb.com/try/download/community>

Para instalar MongoDB, debemos seguir el procedimiento según nuestro sistema operativo, les acercamos más información: <https://docs.mongodb.com/manual/installation>

**Configuración**

Una vez instalado, inicia el servidor MongoDB con el comando **mongod**.

En otra terminal, podemos usar el cliente MongoDB **mongo** para interactuar con la base de datos.

**Conceptos básicos**

* Colecciones: Conjunto de documentos en MongoDB, similar a las tablas en bases de datos relacionales.
* Documentos: Unidades de datos en formato BSON (Binary JSON). Cada documento puede tener una estructura diferente.
* Campos: Atributos dentro de un documento, similares a las columnas en una tabla.

{

"\_id": "unique\_document\_id",

"nombre": "Juan Perez",

"edad": 30,

"email": "juan.perez@example.com"

}

**Conexión de Node.js/Express.js con MongoDB**

*Uso de la biblioteca oficial*

Para conectar MongoDB con una aplicación Node.js/Express.js, utilizaremos la biblioteca **mongodb** o, para una capa de abstracción más sencilla, **mongoose** (<https://mongoosejs.com/>)

*Instalación*

Instalamos **mongoose** (recomendado) usando npm:

Desde la terminal:

npm install mongoose

*Configuración de la conexión*

// Importar Mongoose

const mongoose = require('mongoose');

// URL de conexión a MongoDB (asegúrate de reemplazar los valores por los correctos)

const url = 'mongodb://localhost:27017/mi\_base\_de\_datos';

// Configuración de la conexión

mongoose.connect(url, {

useNewUrlParser: true,

useUnifiedTopology: true

});

// Manejo de eventos de conexión

mongoose.connection.on('connected', () => {

console.log('Conectado a MongoDB');

});

mongoose.connection.on('error', (err) => {

console.error('Error de conexión a MongoDB:', err);

});

mongoose.connection.on('disconnected', () => {

console.log('Desconectado de MongoDB');

});

**Operaciones CRUD con MongoDB**

*Operaciones básicas*

Para realizar operaciones CRUD, definiremos un modelo de datos utilizando Mongoose y ejecutaremos diversas operaciones.

*Definición de un modelo*

// Importar Mongoose

const mongoose = require('mongoose');

// Definir un esquema para el modelo

const usuarioSchema = new mongoose.Schema({

nombre: String,

edad: Number,

email: String

});

// Crear el modelo

const Usuario = mongoose.model('Usuario', usuarioSchema);

*Operaciones CRUD*

Crear (Insertar)

// Crear un nuevo documento

const nuevoUsuario = new Usuario({

nombre: 'Ana Gómez',

edad: 28,

email: 'ana.gomez@example.com'

});

// Guardar el documento en la base de datos

nuevoUsuario.save((err, usuario) => {

if (err) return console.error(err);

console.log('Usuario creado:', usuario);

});

Leer (Buscar)

// Encontrar un usuario por su email

Usuario.findOne({ email: 'ana.gomez@example.com' }, (err, usuario) => {

if (err) return console.error(err);

console.log('Usuario encontrado:', usuario);

});

Actualizar

// Actualizar la edad de un usuario

Usuario.updateOne(

{ email: 'ana.gomez@example.com' },

{ $set: { edad: 29 } },

(err, result) => {

if (err) return console.error(err);

console.log('Usuario actualizado:', result);

}

);

Eliminar

// Eliminar un usuario por su email

Usuario.deleteOne({ email: 'ana.gomez@example.com' }, (err) => {

if (err) return console.error(err);

console.log('Usuario eliminado');

});