MIND HUB.


Backend
Base de Datos

Índice

Bases de datos

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente.

Características:

- Independencia lógica y física de los datos.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Sistema de Gestión de Base de Datos (SGBD)

Son un tipo de software muy específico que sirve de **interfaz** entre: la base de datos, el usuario y las aplicaciones que la utilizan.
Se compone de tres lenguajes: uno de definición de datos, uno de manipulación de datos y otro de consulta.

Tipos de Base de datos

Existen diversos tipos de base de datos, la relacional; la distribuida; NoSQL (no relacionales); orientada a objetos; y, gráficas. La existencia de estas diversas bases de datos se debe a la variedad de forma de trabajo que se requiere de ellas, la dos mas populares en la actualidad son la relacional y no relacional:

Relacionales

Son las más frecuentes por su flexibilidad y facilidad de uso. En este modelo no importa el lugar o la forma en la que estén almacenados los datos. Por el contrario, se accede a la información mediante consultas que permiten acceder de forma ágil y flexible a los datos. Suelen emplear el lenguaje SQL:

- **Relación 1 a 1** (un elemento de una tabla se relaciona con un elemento de la otra, ejemplo: persona/documento)
- **Relación 1 a N** (un elemento de una tabla se relaciona con varios elementos de la otra, ejemplo: persona/autos)
- **Relación N a N** (varios elementos de una tabla se relacionan con varios elementos de otra, ejemplo: alumnos/materias)

Sistema de Gestión de Base de Datos (SGBD)

No Relacionales

Son bases de datos que no usan el lenguaje SQL Entre los lenguajes más usados por las noSQL están:

- CQL (Contextual Query Language)
- GQL (Graph Query Language)
- JSON (JavaScript Object Notation)

MongoDB

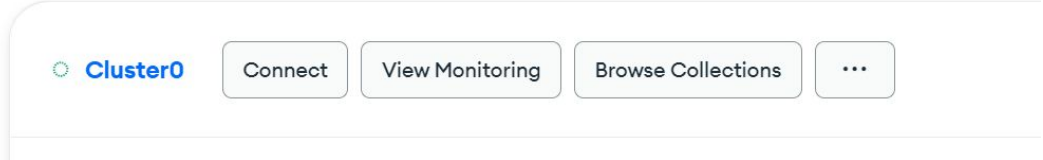
Es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado.

Almacena datos en documentos flexibles similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo.

Registrarse:
https://www.mongodb.com/es/cloud/atlas/register

Configuración de DB en proyecto

Ya registrados en mongo atlas obtendremos los datos de nuestro cluster para poder conectar nuestra base de datos con nuestro backend



En Connect obtendremos el cluster de conexión

Variables de entorno

Crearemos un archivo .env en la raíz de nuestro backend.

La variable global **process.env** es inyectada por el Node en tiempo de ejecución para que la use su aplicación y representa el estado del entorno del sistema en el que se encuentra su aplicación cuando se inicia.

Guardaremos nuestro cluster de conexión en un archivo **.env**, el cual nos permitirá guardar nuestros datos sensibles como contraseñas y datos de conexión en un archivo el cual podemos mantener en nuestro proyecto **sin mostar** en repositorios para ello instalaremos el paquete **dotenv** mediante **npm install dotenv dotenv** es un módulo de dependencia cero que carga variables de entorno desde un .env archivo a process.env.

```
1   MONGO_URI=mongodb+srv://AdrianDeambroggi:[redacted]@cluster0.rgm70.mongodb.net/myTinerary?
```

Agrega tu archivo .env a tu .gitignore para que esta info no sea enviada a tu repositorio

Configuración de DB en proyecto

Inicialmente crearemos una carpeta **config** en nuestro backend.

Crea el archivo de database.js donde estableceremos los parámetros necesarios para la conexión, para ello utilizaremos el paquete **Mongoose** es una librería para Node.js que nos permite escribir consultas para una base de datos de MongoDB, con características como validaciones, construcción de queries, middlewares, conversión de tipos y algunas otras, que enriquecen la funcionalidad de la base de datos. mediante **npm install mongoose**

```
const mongoose = require('mongoose')

mongoose.connect(process.env.MONGO_URI,{

    useUnifiedTopology: true,
    useNewUrlParser: true,

})
.then(()=>console.log("Database conected"))
.catch(err => console.error(err))
```

useUnifiedTopology: Falso por defecto. Establecerlo en true para optar por usar el nuevo motor de administración de conexión del controlador MongoDB. Debe establecer esta opción en true, excepto en el caso improbable de que le impida mantener una conexión estable.

useNewUrlParser: El controlador básico de MongoDB ha perdido su conexión actual Analizador de cadenas. Debido a que este es un cambio importante, se agregó el indicador useNewUrlParser para que los usuarios puedan usar el analizador anterior si encuentran un error en el analizador nuevo.

Agregando DB a Server

Llamamos a nuestra configuración de nuestra db y nuestro archivo .env desde nuestro server para que lo utilice y lo ejecute. Si todo esta OK veremos en nuestra consola de backend el mensaje definido en la configuración de nuestra base de datos **"Database conected"**

```
require('dotenv').config()
require('./config/database')
const express = require('express')
const PORT = 4000

const app = express()

app.listen(PORT,()=> console.log('Server ready on PORT ' + PORT))
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Server ready on PORT 4000
Database connected
```

¡Muchas gracias!

MIND
HUB.