Análisis y Desarrollo de Software
Ficha: 2791446
Competencia: construcción de software (220501096)
Evidencia: GA6-220501096-AA1-EV03.
Link: github/dmikan

Creación de los objetos de la Base de Datos (NoSQL)

Danny Julián Perilla Mikán

Enero, 2025

1 Introducción

Las bases de datos no relacionales, también conocidas como NoSQL, son sistemas diseñados para manejar grandes volúmenes de datos de manera flexible y eficiente. A diferencia de las bases de datos relacionales, que almacenan la información en tablas organizadas por filas y columnas, las bases de datos NoSQL utilizan estructuras más diversas como documentos, claves-valor, grafos o columnas. Esta flexibilidad permite trabajar con datos no estructurados o semi-estructurados, ideal para aplicaciones modernas como redes sociales, sistemas de recomendación y análisis en tiempo real. Una ventaja clave es su escalabilidad horizontal, que facilita manejar grandes cantidades de datos distribuidos en varios servidores.

2 Conexión remota (opcional)

Para este ejercicio, se utilizó MongoDB Atlas como plataforma para gestionar una base de datos en la nube. Se creó un clúster y se conectó de forma remota al entorno local utilizando Visual Studio Code con la extensión "MongoDB for VS Code". El proceso consistió en instalar la extensión desde la sección "Extensiones" de VS Code, abrir el "Command Palette" (Comando Rápido) y seleccionar la opción "MongoDB: Connect". Luego, se pegó el string de conexión proporcionado por MongoDB Atlas para establecer el vínculo con la base de datos. Este método es una de las alternativas disponibles, junto con herramientas como MongoDB Compass, la terminal o directamente sobre nuestras aplicaciones web.

3 Caso de estudio

Se desean registrar los datos de varios tipos de carros en una base de datos documental, en este caso se usa MongoDB, que utiliza el formato JSON para almacenar datos. A continuación se muestran algunas de las operaciones de inserción, actualización y eliminación.

- **db.cars.insertOne(...)**: Inserta un solo documento en la colección `cars` con los datos especificados (placa, número de identificación del vehículo, modelo, marca y kilometraje).
- **db.cars.insertMany([...])**: Inserta múltiples documentos en la colección `cars`. Cada documento contiene la información de un coche (placa, número de identificación, modelo, marca y kilometraje).
- **db.cars.updateOne(...)**: Actualiza el primer documento que coincide con el filtro dado (en este caso, el `license_plate ABC123`) estableciendo el valor del campo `model` a 2025.

- **db.cars.deleteOne(...)**: Elimina el primer documento que coincide con el filtro dado (en este caso, el `license_plate` `IND123`).
- **db.cars.find()**: Realiza una consulta para recuperar todos los documentos de la colección `cars`.

El siguiente script se puede correr en Visual Studio Code con la extensión para MongoDB o directamente se puede digitar cada comando en una terminal o shell.

```
// Connect to the database called 'car_dealer'
use('car_dealer')

// Insert a single document into the 'cars' collection
db.cars.insertOne({
    license_plate: "ABC123",
    vehicle_identification_number: "1HGCM82633A123456",
    model: 2023,
    brand: "Toyota",
    kilometer_reading: 30000
})

// Insert multiple documents into the 'cars' collection using insertMany
db.cars.insertMany([
    {
        license_plate: "XYZ789",
        vehicle_identification_number: "2FTRX18W1XCA12345",
        model: 2021,
        brand: "Ford",
        kilometer_reading: 45000
    },
    {
        license_plate: "LMN456",
        vehicle_identification_number: "3N1BC13E38L123456",
        model: 2020,
        brand: "Nissan",
        kilometer_reading: 35000
    },
    {
        license_plate: "JKL321",
        vehicle_identification_number: "5UXWX7C56E0D12345",
        model: 2022,
        brand: "BMW",
        kilometer_reading: 20000
    },
    {
        license_plate: "GHI654",
        vehicle_identification_number: "WAUZZZ8V1DA123456",
        model: 2019,
        brand: "Audi",
        kilometer_reading: 60000
    }
]);

// Insert another single document into the collection
db.cars.insertOne({
    license_plate: "IND123",
    vehicle_identification_number: "JHMCM56557C403024",
    model: 2019,
```

```
    brand: "Hyundai",
    kilometer_reading: 80000
});

// updateOne: Only the first document that matches the filter condition will be updated, according to the internal order of the documents in the database. If you need to work with all documents that have the same license_plate, use updateMany.

db.cars.updateOne(
    { license_plate: "ABC123" },
    { $set: { model: 2025 } }
);

// deleteOne: Only the first document that matches the filter will be deleted. If you need to work with all documents that have the same license_plate, use deleteMany.

db.cars.deleteOne(
    { license_plate: "IND123" }
);

// Query the records in the database
db.cars.find({})
```

Una vez se han generado los registros se pueden consultar desde cualquier lugar que esté conectado a la base de datos remota usando las credenciales de acceso. Lo hacemos ahora desde una terminal para probar varios entornos.

```
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\djper>mongosh "mongodb+srv://cluster0.wzjym.mongodb.net/" --apiVersion 1 --username djperillam
Enter password: **********
Current Mongosh Log ID: 6790423bc7a77e1e49f2b08e
Connecting to:          mongodb+srv://<credentials>@cluster0.wzjym.mongodb.net/?appName=mongosh+2.0.0
Using MongoDB:          8.0.4 (API Version 1)
Using Mongosh:          2.0.0
mongosh 2.3.8 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-xezb4h-shard-0 [primary] test> show dbs
car_dealer    40.00 KiB
turismo_web   72.00 KiB
admin        384.00 KiB
local          9.82 GiB
Atlas atlas-xezb4h-shard-0 [primary] test> use car_dealer
switched to db car_dealer
```

```
Atlas atlas-xezb4h-shard-0 [primary] car_dealer> show collections
cars
Atlas atlas-xezb4h-shard-0 [primary] car_dealer> db.cars.find({})
[
  {
    _id: ObjectId("67903ea4b193fcead4cc1b4f"),
    license_plate: 'ABC123',
    vehicle_identification_number: '1HGCM82633A123456',
    model: 2025,
    brand: 'Toyota',
    kilometer_reading: 30000
  },
  {
    _id: ObjectId("67903ea4b193fcead4cc1b50"),
    license_plate: 'XYZ789',
    vehicle_identification_number: '2FTRX18W1XCA12345',
    model: 2021,
    brand: 'Ford',
    kilometer_reading: 45000
  },
  {
    _id: ObjectId("67903ea4b193fcead4cc1b51"),
    license_plate: 'LMN456',
    vehicle_identification_number: '3N1BC13E38L123456',
    model: 2020,
    brand: 'Nissan',
    kilometer_reading: 35000
  },
  {
    _id: ObjectId("67903ea4b193fcead4cc1b52"),
    license_plate: 'JKL321',
    vehicle_identification_number: '5UXWX7C56E0D12345',
    model: 2022,
    brand: 'BMW',
    kilometer_reading: 20000
  },
  {
    _id: ObjectId("67903ea4b193fcead4cc1b53"),
    license_plate: 'GHI654',
    vehicle_identification_number: 'WAUZZZ8V1DA123456',
    model: 2019,
    brand: 'Audi',
    kilometer_reading: 60000
  }
```

```
]
Atlas atlas-xezb4h-shard-0 [primary] car_dealer>
```

4 Conclusiones

Las bases de datos no relacionales no requieren un esquema fijo de metadatos y campos unificados, lo que les permite a cada registro tener su propio conjunto de campos clave-valor. Esta flexibilidad es una ventaja significativa cuando se trabaja con conjuntos de datos que no necesariamente comparten la misma estructura de información. Aunque en el ejemplo del concesionario todos los registros siguen un esquema común, esta uniformidad no es obligatoria en bases de datos no relacionales, lo que permite una mayor adaptabilidad a diversos tipos de datos.