Documentación Ampliada: Bases de Datos y Mongoose en Node.js con Ejemplos de Superhéroes

Índice

5. **Comparación con otras Tecnologías**
    - 5.1 Mongoose vs Sequelize
    - 5.2 Mongoose vs MongoDB Nativo

---

1. Introducción a Bases de Datos

Una **base de datos** es un sistema que permite almacenar y organizar grandes volúmenes de información de manera eficiente. Las bases de datos se clasifican en **relacionales (SQL)** y **no relacionales (NoSQL)**, con diferencias importantes en estructura y uso.

1.1 Bases de Datos Relacionales (SQL)

Las bases de datos relacionales organizan la información en tablas conectadas mediante relaciones. SQL (Structured Query Language) permite la manipulación y consulta de estos datos.

Ventajas

- **Integridad referencial**: Las relaciones entre tablas aseguran que los datos estén correctamente estructurados.
- **Soporte para transacciones**: Las operaciones se pueden realizar de manera transaccional, garantizando que todos los cambios sean coherentes.

Desventajas

- **Rigidez**: La estructura de las tablas es rígida y difícil de modificar.
- **Escalabilidad limitada**: Son menos eficientes para escalar horizontalmente.

Ejemplos: MySQL, PostgreSQL.

1.2 Bases de Datos No Relacionales (NoSQL)

Las bases de datos NoSQL permiten almacenar datos en documentos flexibles sin estructura fija, facilitando el almacenamiento de datos semiestructurados o cambiantes.

Ventajas

- **Escalabilidad horizontal**: Permiten distribuir los datos de manera más eficiente en entornos grandes.
- **Flexibilidad de esquema**: No necesitan una estructura fija, facilitando el cambio de esquema.

Desventajas

- **Menor consistencia de datos**: No siempre garantizan la integridad de los datos.
- **Limitado soporte para transacciones complejas**: Algunas bases de datos NoSQL no soportan

transacciones ACID.

Ejemplos: MongoDB, Redis.

---

2. MongoDB Compass

2.1 ¿Qué es MongoDB Compass?

MongoDB Compass es una **herramienta gráfica** que permite gestionar y visualizar datos en MongoDB sin necesidad de escribir comandos. Ideal para desarrolladores que prefieren una **interfaz visual** para explorar y modificar datos.

Para descargar MongoDB Compass dirigete a la siguiente direccion:

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

y descarga la adecuada para tu sistema operativo

2.2 Conectarse a una Base de Datos con MongoDB Compass

Para conectarse a MongoDB usando Compass, se utiliza una **cadena de conexión** que incluye las credenciales y la ubicación de la base de datos.

Formato de Cadena de Conexión:

```
mongodb+srv://usuario:password@cluster0.mongodb.net/nombreBD?retryWrites=true&w=majority
```

Pasos en MongoDB Compass:

1. Abre MongoDB Compass.
2. Ingresa la cadena de conexión en el campo “Connection String”.
3. Haz clic en “Connect” para conectarte a la base de datos.

2.3 Crear y Manejar Colecciones en MongoDB Compass

1. Selecciona la base de datos en el panel de navegación de Compass.
2. Haz clic en “Create Collection” y nómbrala, por ejemplo, `superheroes`.

2.4 Agregar Documentos en una Colección con MongoDB Compass

1. Selecciona la colección `superheroes`.

2. Haz clic en “Insert Document” para añadir un documento en formato JSON:

```
{
    "nombreSuperHeroe": "Spiderman",
```

```
    "nombreReal": "Peter Parker",
    "edad": 25,
    "planetaOrigen": "Tierra",
    "debilidad": "Radioactiva",
    "poderes": ["Trepar paredes", "Sentido arácnido", "Super fuerza", "Agilidad"],
    "aliados": ["Ironman"],
    "enemigos": ["Duende Verde"]
}
```

Este bloque de código JSON crea un superhéroe llamado **Spiderman** con varias propiedades como `nombreReal`, `edad`, `planetaOrigen`, `poderes`, `aliados` y `enemigos`. Al guardar, Compass lo almacena en la base de datos.

2.5 Consultar Datos en MongoDB Compass

1. Dirígete a "Filter" en la colección `superheroes`.
2. Ingresa el filtro en JSON, por ejemplo, `{ "planetaOrigen": "Tierra" }`.
3. Haz clic en "Apply" para ver los resultados.

Este filtro muestra todos los superhéroes cuyo `planetaOrigen` es "Tierra".

---

3. MongoDB: Base de Datos NoSQL

MongoDB es una base de datos orientada a documentos, ideal para aplicaciones que requieren flexibilidad de esquema y escalabilidad.

3.1 Ventajas y Desventajas de MongoDB

- **Ventajas**:
  - Escalabilidad horizontal.
  - Flexibilidad para almacenar datos semi-estructurados.
- **Desventajas**:
  - Menor soporte para transacciones complejas.

3.2 Ejemplo de Base de Datos de Superhéroes

MongoDB permite almacenar datos de superhéroes en documentos JSON, organizando cada héroe en un solo documento para facilitar su consulta y modificación.

3.3 Métodos CRUD en MongoDB

Crear un Superhéroe

```
db.collection('superheroes').insertOne({
```

```
  nombreSuperHeroe: 'Spiderman',
  nombreReal: 'Peter Parker',
  edad: 25,
  planetaOrigen: 'Tierra',
  debilidad: 'Radioactiva',
  poderes: ['Trepar paredes', 'Sentido arácnido', 'Super fuerza', 'Agilidad'],
  aliados: ['Ironman'],
  enemigos: ['Duende Verde']
});
```

Este comando inserta un nuevo documento en la colección `superheroes` con los detalles de Spiderman, incluyendo su nombre, edad, poderes y aliados.

Leer Superhéroes

```
db.collection('superheroes').find({ planetaOrigen: 'Tierra' });
```

Este comando busca y devuelve todos los superhéroes cuyo planeta de origen es “Tierra”.

Actualizar Superhéroes

```
db.collection('superheroes').updateOne(
  { nombreSuperHeroe: 'Spiderman' },
  { $set: { edad: 26 } }
);
```

Aquí se actualiza la edad de Spiderman a 26 en la colección `superheroes`.

Eliminar un Superhéroe

```
db.collection('superheroes').deleteOne({ nombreSuperHeroe: 'Spiderman' });
```

Este comando elimina el documento de Spiderman de la colección `superheroes`.

---

4. Mongoose como ODM (Object Data Modeling)

4.1 ¿Por qué utilizar un ODM?

Un **ODM** (Object Data Modeling) como Mongoose abstrae la interacción con MongoDB, facilitando el manejo de datos en aplicaciones complejas al implementar validaciones, middlewares y consistencia de esquemas. Esto es útil en aplicaciones Node.js para manejar datos de manera consistente.

Ventajas de Mongoose:

- **Validación de datos**: Verifica que los datos almacenados cumplan con las reglas definidas.
- **Abstracción de complejidad**: Simplifica las operaciones CRUD con métodos de alto nivel.
- **Middlewares**: Automatiza acciones antes o después de operaciones, mejorando la gestión de

procesos.

4.2 Definición de

Esquema y Modelo de Superhéroes en Mongoose

Ejemplo de Esquema de Superhéroes:

```
const superheroSchema = new mongoose.Schema({
  nombreSuperHeroe: { type: String, required: true },
  nombreReal: { type: String, required: true },
  edad: { type: Number, min: 0 },
  planetaOrigen: { type: String, default: 'Desconocido' },
  debilidad: String,
  poderes: [String],
  aliados: [String],
  enemigos: [String],
  createdAt: { type: Date, default: Date.now }
});
```

Este esquema define la estructura de un superhéroe, incluyendo validaciones en `nombreSuperHeroe` y `nombreReal` para que siempre tengan valor, y un valor predeterminado `Desconocido` para `planetaOrigen`.

4.3 Creación de Modelos en Mongoose

Un modelo representa la colección en MongoDB y permite interactuar con ella.

Ejemplo de Creación de Modelo:

```
const SuperHero = mongoose.model('SuperHero', superheroSchema);
```

Este modelo se basa en el esquema `superheroSchema` y permite crear, leer, actualizar y eliminar documentos de la colección `SuperHero`.

4.4 Ejemplos CRUD en Mongoose

Crear un Superhéroe

```
SuperHero.create({
  nombreSuperHeroe: 'Ironman',
  nombreReal: 'Tony Stark',
  edad: 45,
  planetaOrigen: 'Tierra',
  debilidad: 'Dependiente de la tecnología',
  poderes: ['Armadura blindada', 'Volar', 'Láseres'],
  aliados: ['Spiderman'],
  enemigos: ['Mandarín']
});
```

Este bloque crea un nuevo superhéroe, Ironman, con sus propiedades y relaciones en la colección `SuperHero`.

Buscar Superhéroes

```
SuperHero.find({ planetaOrigen: 'Tierra' });
```

Busca todos los superhéroes cuyo `planetaOrigen` es “Tierra”.

Actualizar Superhéroes

```
SuperHero.updateOne(
  { nombreSuperHeroe: 'Spiderman' },
  { $set: { edad: 26 } }
);
```

Actualiza la edad de Spiderman en la base de datos a 26.

Eliminar Superhéroes

```
SuperHero.deleteOne({ nombreSuperHeroe: 'Ironman' });
```

Elimina el documento de Ironman de la colección.

---

5. Comparación con otras Tecnologías

5.1 Mongoose vs Sequelize

Mongoose es un ODM para bases de datos NoSQL como MongoDB, mientras que **Sequelize** es un ORM (Object Relational Mapping) para SQL.

Característica	Mongoose	Sequelize
Base de Datos	MongoDB (NoSQL)	MySQL, PostgreSQL (SQL)
Modelo de Datos	Flexible, JSON	Relacional, tablas
Validaciones	Incluidas en el esquema	Validaciones incluidas
Transacciones	Limitadas	Soporte completo

5.2 Mongoose vs MongoDB Nativo

Mongoose ofrece una capa de abstracción sobre MongoDB, facilitando el manejo de datos y agregando validaciones mediante esquemas.