Conexión a Bases de Datos con Express

Fundamentos y Mejores Prácticas

Julian F. Latorre

November 13, 2024

Agenda

- Introducción y Contextualización
- Fundamentos de Bases de Datos y Express
- Configuración del Entorno
- *Break*
- Implementación Práctica
- Pruebas y Depuración
- Cierre y Preguntas

¿Por qué es importante?

- **Persistencia de Datos**
  - Fundamental para aplicaciones web
  - Estándar de la industria
- **Arquitectura Web Moderna**
  - Frontend: Interfaz de usuario
  - Backend (Express): Procesamiento
  - Base de datos: Almacenamiento

Casos de Uso Comunes

- **Redes Sociales**
  - Posts
  - Comentarios
  - Likes
- **E-Commerce**
  - Productos
  - Órdenes
  - Usuarios
- **Aplicaciones de Tareas**
  - Actividades
  - Estados
  - Seguimiento

Instalación de Dependencias

```
// Dependencias principales
npm install express     // Framework web
npm install mysql2      // Driver de MySQL
npm install dotenv      // Variables de entorno
```

Estructura del Proyecto

```
proyecto/
├── src/
│   ├── config/
│   │   ├── database.js
│   │   └── .env
│   ├── models/
│   │   └── user.model.js
│   ├── routes/
│   │   └── user.routes.js
│   └── app.js
├── package.json
└── README.md
```

Configuración de la Base de Datos

```
const mysql = require('mysql2');
const pool = mysql.createPool({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME,
    waitForConnections: true,
    connectionLimit: 10,
    queueLimit: 0
});

const promisePool = pool.promise();
```

Modelo de Usuario - Parte 1

```
class UserModel {
    static async createTable() {
        const createTableSQL = `
            CREATE TABLE IF NOT EXISTS users (
                id INT PRIMARY KEY AUTO_INCREMENT,
                username VARCHAR(50) NOT NULL UNIQUE,
                email VARCHAR(100) NOT NULL UNIQUE,
                created_at TIMESTAMP DEFAULT
                    CURRENT_TIMESTAMP
            )
        `;
        await pool.query(createTableSQL);
    }
```

Modelo de Usuario - Parte 2

```
    static async create(userData) {
        const { username, email } = userData;
        const [result] = await pool.query(
            'INSERT INTO users (username, email)
                VALUES (?, ?)',
            [username, email]
        );
        return { id: result.insertId, username, email
            };
    }
```

Manejo de Errores

- **Tipos de Errores**
  - Errores de conexión
  - Errores de consulta
  - Errores de validación
- **Mejores Prácticas**
  - Uso de try/catch
  - Logging estructurado
  - Mensajes de error claros

Transacciones

- **Características**
  - Atomicidad
  - Consistencia
  - Aislamiento
  - Durabilidad
- **Cuándo Usarlas**
  - Operaciones múltiples relacionadas
  - Necesidad de rollback
  - Consistencia de datos crítica

Recursos Adicionales

- **Documentación**
  - Express: `https://expressjs.com/`
  - MySQL2: `https://github.com/sidorares/node-mysql2`
- **Temas Avanzados**
  - Autenticación
  - Manejo de sesiones
  - Middleware avanzado
  - Optimización de consultas