Taller Práctico: Desarrollo Backend con Express y MySQL

Instructor Backend

25 de noviembre de 2024

Índice

1. Introducción

1.1. Objetivo del Taller

El presente instructivo tiene como objetivo guiar paso a paso el desarrollo de una aplicación backend utilizando Express.js y MySQL, proporcionando una base sólida para comprender la construcción de APIs REST.

1.2. Competencias a Desarrollar

Al finalizar este taller, los participantes podrán:

- Configurar un entorno de desarrollo Node.js
- Crear servidores con Express.js
- Conectar aplicaciones con bases de datos MySQL
- Implementar operaciones CRUD
- Manejar rutas y middleware

2. Configuración del Entorno

2.1. Instalación de Dependencias

Ejecutar los siguientes comandos en terminal:

```
# Inicializar proyecto
npm init -y

# Instalar dependencias principales
npm install express mysql2 body-parser dotenv
npm install --save-dev nodemon
```

2.2. Estructura de Proyecto

Crear la siguiente estructura de directorios:

```
backend-workshop/
 src/
    config/
       database.js
    controllers/
       usuariosController.js
    routes/
       usuariosRoutes.js
    app.js
 .env
 package.json
 README.md
```

3. Configuración de Base de Datos

3.1. Script de Conexión

En `src/config/database.js`:

```
require('dotenv').config();
const mysql = require('mysql2/promise');

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
});

module.exports = pool;
```

3.2. Modelo de Datos

Script SQL para crear tabla de usuarios:

```
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. Desarrollo de Controladores

En src/controllers/usuariosController.js:

```
const db = require('../config/database');

exports.listarUsuarios = async (req, res) => {
    try {
        const [usuarios] = await db.query('SELECT * FROM usuarios');
        res.json(usuarios);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
};

exports.crearUsuario = async (req, res) => {
    const { nombre, email } = req.body;
    try {
        const [resultado] = await db.query(
            'INSERT INTO usuarios (nombre, email) VALUES (?, ?)',
            [nombre, email]
        );
        res.status(201).json({
```

```
            id: resultado.insertId,
            mensaje: 'Usuario creado exitosamente'
        });
    } catch (error) {
        res.status(400).json({ error: error.message });
    }
};
```

5. Configuración de Rutas

En src/routes/usuariosRoutes.js:

```
const express = require('express');
const router = express.Router();
const usuariosController = require('../controllers/usuariosController
    ');

router.get('/', usuariosController.listarUsuarios);
router.post('/', usuariosController.crearUsuario);

module.exports = router;
```

6. Aplicación Principal

En src/app.js:

```
const express = require('express');
const bodyParser = require('body-parser');
const usuariosRoutes = require('./routes/usuariosRoutes');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());
app.use('/usuarios', usuariosRoutes);

app.listen(PORT, () => {
    console.log(`Servidor corriendo en puerto ${PORT}`);
});
```

7. Archivos de Configuración

7.1. Variables de Entorno

Contenido de .env:

```
DB_HOST=localhost
DB_USER=tu_usuario
DB_PASSWORD=tu_contraseña
DB_NAME=backend_workshop
PORT=3000
```

8. Pruebas y Validación

8.1. Comandos para Ejecución

Añadir al `package.json`:

```
{
  "scripts": {
    "start": "node src/app.js",
    "dev": "nodemon src/app.js"
  }
}
```

9. Conclusiones

- Hemos construido una API REST básica
- Implementamos conexión con base de datos
- Separamos responsabilidades usando MVC

10. Recomendaciones Finales

1. Implementar validaciones de datos
2. Añadir manejo de errores
3. Usar autenticación en endpoints
4. Documentar la API