

Taller Práctico: Conexión a Bases de Datos con Express

Julian F. Latorre
Bootcamp Desarrollo Web Full Stack

13 de noviembre de 2024

Información General

Este taller está diseñado para poner en práctica los conceptos aprendidos en la clase de Conexión a Bases de Datos con Express. Cada ejercicio incluye código base y guías paso a paso para su implementación.

Tiempo estimado: 2-3 horas **Requisitos previos:** Node.js y MySQL instalados

Ejercicio 1: Sistema de Biblioteca

Configuración Inicial

Implementaremos un sistema de gestión de biblioteca que permitirá:

- Gestionar libros
- Gestionar préstamos
- Gestionar usuarios

Paso 1: Configura el proyecto

```
mkdir biblioteca-express
cd biblioteca-express
npm init -y
npm install express mysql2 dotenv
```

Paso 2: Estructura el proyecto

```
biblioteca-express/
├── src/
│   ├── config/
│   │   ├── database.js
│   │   └── .env
│   ├── models/
│   │   ├── book.model.js
│   │   └── loan.model.js
│   ├── routes/
│   │   ├── book.routes.js
│   │   └── loan.routes.js
│   └── app.js
└── package.json
```

🔗 Implementación de la Base de Datos

Paso 3: Crea las tablas necesarias

```
-- Código base para la tabla de libros
CREATE TABLE books (
  id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  author VARCHAR(100) NOT NULL,
  isbn VARCHAR(13) UNIQUE,
  available BOOLEAN DEFAULT true,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Completa la tabla de préstamos
CREATE TABLE loans (
  -- TODO: Implementa los campos necesarios
);
```

📦 Implementación del Modelo

Paso 4: Implementa el modelo de libros

```
// models/book.model.js
const { pool } = require('../config/database');

class BookModel {
  // TODO: Implementa los métodos CRUD
  static async create(bookData) {
    // Implementa la creación de libros
  }

  static async findAll() {
    // Implementa la búsqueda de todos los libros
  }

  // ... más métodos
}
```

Implementación de Rutas

Paso 5: Implementa las rutas

```
// routes/book.routes.js
const express = require('express');
const router = express.Router();
const BookModel = require('../models/book.model');

// TODO: Implementa las rutas CRUD
router.post('/', async (req, res) => {
  // Implementa la creación
});

// ... más rutas
```

Ejercicio 2: Sistema de Préstamos

Implementación de Transacciones

Implementa un sistema de préstamos que:

- Verifique la disponibilidad del libro
- Actualice el estado del libro
- Registre el préstamo
- Use transacciones para garantizar la integridad

```
async function loanBook(bookId, userId) {
  const connection = await pool.getConnection();
  try {
    // TODO: Implementa la transacción
    await connection.beginTransaction();

    // 1. Verifica disponibilidad
    // 2. Actualiza estado
    // 3. Registra préstamo

    await connection.commit();
  } catch (error) {
    await connection.rollback();
    throw error;
  } finally {
    connection.release();
  }
}
```

Ejercicio 3: Reportes y Consultas

Implementación de Consultas Complejas

Implementa las siguientes consultas:

1. Libros más prestados
2. Usuarios con préstamos vencidos
3. Disponibilidad de libros por categoría

```
// TODO: Implementa las consultas
async function getTopBooks() {
  // Implementa la consulta
}
```

Recursos Adicionales

- Documentación de Express: <https://expressjs.com/>
- Documentación de MySQL2: <https://github.com/sidorares/node-mysql2>
- Tutorial de transacciones MySQL: <https://dev.mysql.com/doc/refman/8.0/en/commit.html>