

Taller: Creación de una API REST con 3 tablas conectadas a MySQL

Contexto

Imagina que estás desarrollando un sistema para gestionar las reservas de vehículos en un concesionario. Los usuarios pueden registrarse, crear reservas y elegir un vehículo disponible. El objetivo es construir una API REST que permita gestionar **Usuarios**, **Vehículos**, y **Reservas**. La API debe estar conectada a una base de datos MySQL, y realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para cada una de estas entidades.

Requisitos

1. Base de datos MySQL con 3 tablas:

- **Usuarios:** Registra la información de los usuarios que crean reservas.
- **Vehículos:** Almacena los vehículos disponibles para reservar.
- **Reservas:** Almacena las reservas de los vehículos, conectando usuarios con vehículos.

2. Controladores para realizar operaciones CRUD en cada tabla:

- **Usuarios:** Crear, leer, actualizar y eliminar usuarios.
- **Vehículos:** Crear, leer, actualizar y eliminar vehículos.
- **Reservas:** Crear, leer, actualizar y eliminar reservas.

Detalles de las tablas en MySQL

1. Tabla usuarios:

```
CREATE TABLE usuarios (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  nombre VARCHAR(100),  
  email VARCHAR(100),  
  telefono VARCHAR(15)  
);
```

2. Tabla vehiculos:

```
CREATE TABLE vehiculos (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  marca VARCHAR(50),  
  modelo VARCHAR(50),  
  año INT  
);
```

3. Tabla reservas:

```
CREATE TABLE reservas (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  usuario_id INT,  
  vehiculo_id INT,  
  fecha_reserva DATE,  
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id),  
  FOREIGN KEY (vehiculo_id) REFERENCES vehiculos(id)  
);
```

Pasos a seguir

Paso 1: Crear la base de datos en MySQL

- Utiliza los scripts SQL anteriores para crear las tablas en tu base de datos MySQL.

Paso 2: Crear la API en Node.js con Express

1. Instala las dependencias:

- Express
- MySQL2
- Morgan (para logging)

npm install express mysql2 morgan

2. Crea la estructura de la API:

- Crea carpetas para controllers, routes, y database.
- Crea un archivo database.ts para gestionar la conexión a MySQL.
- Crea archivos de rutas para cada entidad (usuarios, vehiculos, reservas).

Paso 3: Crear controladores CRUD

- **Controlador de usuarios:** Crear operaciones CRUD para la tabla usuarios.
- **Controlador de vehículos:** Crear operaciones CRUD para la tabla vehiculos.
- **Controlador de reservas:** Crear operaciones CRUD para la tabla reservas, asegurando que al crear una reserva, se valide que el usuario y el vehículo existen.

Paso 4: Crear rutas

- Crea rutas que gestionen las operaciones de cada tabla. Por ejemplo:

Usuarios:

- GET /usuarios - Obtener todos los usuarios.
- POST /usuarios - Crear un nuevo usuario.
- PUT /usuarios/:id - Actualizar un usuario.
- DELETE /usuarios/:id - Eliminar un usuario.

Vehículos:

- GET /vehiculos - Obtener todos los vehículos.
- POST /vehiculos - Crear un nuevo vehículo.
- PUT /vehiculos/:id - Actualizar un vehículo.
- DELETE /vehiculos/:id - Eliminar un vehículo.

Reservas:

- GET /reservas - Obtener todas las reservas.
- POST /reservas - Crear una nueva reserva.
- PUT /reservas/:id - Actualizar una reserva.
- DELETE /reservas/:id - Eliminar una reserva