Trabajo Práctico Obligatorio: Integración de Aplicaciones - Sistema de Gestión Escolar

En este trabajo práctico, se solicita desarrollar una aplicación web que permita realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en una base de datos MySQL utilizando Node.js, el módulo mysql2 y el framework Express. El objetivo es crear un sistema de gestión escolar para administrar información relacionada con estudiantes, profesores y cursos, y establecer las relaciones entre ellos.

La aplicación debe cumplir con los siguientes requisitos:

1. Configuración de la base de datos:
    a) Utilizar una base de datos MySQL local.
    b) Crear las siguientes tablas:
        - Tabla "estudiantes" con los campos: id (entero, clave primaria), nombre (cadena de texto), edad (entero) y grado (cadena de texto).
        - Tabla "profesores" con los campos: id (entero, clave primaria), nombre (cadena de texto), especialidad (cadena de texto) y email (cadena de texto).
        - Tabla "cursos" con los campos: id (entero, clave primaria), nombre (cadena de texto) y descripcion (cadena de texto).
        - Tabla "estudiantes_cursos" para establecer la relación entre estudiantes y cursos. Esta tabla debe tener los campos: estudiante_id (entero, clave foránea a la tabla "estudiantes") y curso_id (entero, clave foránea a la tabla "cursos").
2. API RESTful:
    a) Crear una API RESTful que permita realizar las operaciones CRUD sobre las tablas "estudiantes", "profesores" y "cursos".
    b) Utilizar los métodos HTTP adecuados para cada operación (GET, POST, PUT, DELETE).
    c) Los endpoints de la API deben seguir las convenciones RESTful y ser los siguientes:
        - Obtener todos los estudiantes: /estudiantes (GET)
        - Obtener un estudiante por su ID: /estudiantes/:id (GET)
        - Crear un nuevo estudiante: /estudiantes (POST)
        - Actualizar un estudiante existente: /estudiantes/:id (PUT)
        - Eliminar un estudiante: /estudiantes/:id (DELETE)
        - Obtener todos los profesores: /profesores (GET)
        - Obtener un profesor por su ID: /profesores/:id (GET)

- Crear un nuevo profesor: /profesores (POST)
- Actualizar un profesor existente: /profesores/:id (PUT)
- Eliminar un profesor: /profesores/:id (DELETE)
- Obtener todos los cursos: /cursos (GET)
- Obtener un curso por su ID: /cursos/:id (GET)
- Crear un nuevo curso: /cursos (POST)
- Actualizar un curso existente: /cursos/:id (PUT)
- Eliminar un curso: /cursos/:id (DELETE)
- Obtener todos los cursos de un estudiante: /estudiantes/:id/cursos
- Obtener todos los estudiantes de un curso: `/cursos/:id/estudiantes` (GET)
- Agregar un estudiante a un curso: `/cursos/:id/estudiantes` (POST)
- Eliminar un estudiante de un curso: `/cursos/:id/estudiantes/:estudianteId` (DELETE)

3. Validación de datos:

a) Implementar validaciones para asegurar que los campos obligatorios de cada entidad sean ingresados correctamente.

b) Validar que los campos de edad y grado sean del tipo adecuado y cumplan con restricciones específicas.

c) Manejar correctamente los errores de validación y mostrar mensajes de error claros al usuario.

Obs: Investigue la siguiente librería: https://express-validator.github.io/docs
Implemente las validaciones con la dependencia anterior.

4. Interfaz de usuario:

a) Realizar las pruebas de cada endpoint utilizando Insonmia o Postman.

5. Documentación:

a) Incluir documentación clara y concisa sobre cómo ejecutar la aplicación y realizar las pruebas.

b) Recuerda utilizar buenas prácticas de programación, como la modularización del código, el uso de controladores y enrutadores, y la gestión adecuada de errores.

Fecha de entrega: 05/06/2023