Instituto Politécnico Formosa

Práctica CRUD de Tareas y Usuarios con Backend usando Sequelize

Objetivo

Desarrollar una aplicación backend que permita realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para gestionar tareas y usuarios, utilizando Node.js, Express, MySQL y Sequelize ORM.

Criterios de Evaluación

1. Presentación del código

- Código limpio, ordenado y bien indentado.
- Uso obligatorio de try-catch en todos los controladores.
- Organización correcta del proyecto en carpetas: src/config, src/models, src/routes, src/controllers.
- Uso exclusivo de módulos ESModules (import / export).
- Código funcional, modularizado y sin errores de ejecución.

2. Validaciones y lógica

- Validaciones correctamente implementadas dentro del controlador.
- Uso de Sequelize con conexión a MySQL y uso de modelos definidos.
- Respuestas con mensajes claros y códigos HTTP apropiados (200, 201, 400, 404, 500 etc.).
- Verificación de unicidad en creación y edición.
- Verificación de existencia previa antes de editar o eliminar un recurso.
- Mostrar mensajes de éxito o error al realizar operaciones CRUD.

Entrega mediante Git y GitHub – Uso de ramas y commits

Requisitos obligatorios para el control de versiones

- El proyecto debe ser versionado con Git desde el inicio.
- Se deben crear y utilizar dos ramas:
  - **main**: rama principal para la entrega final.
  - **develop**: rama de desarrollo donde se realiza el examen.
- Todo el desarrollo debe hacerse en la rama develop.
- Al finalizar, se debe hacer un merge limpio de develop hacia main, sin conflictos.
- Se deben realizar al menos 5 commits con mensajes claros y descriptivos.

Requisitos del repositorio remoto

- El nombre del repositorio en GitHub debe seguir la siguiente convención:
  - crud-tasks-apellido-nombre. Ejemplo: crud-tasks-perez-juan
- El repositorio debe contener:
  - Todos los archivos del proyecto (app.js, package.json, src/ completo).
  - El archivo .env.example con los nombres de las variables utilizadas.
- El archivo .env y la carpeta node_modules no deben subirse.

- El enlace al repositorio debe entregarse en Google Classroom antes del horario límite establecido.

Consignas

Base de Datos:

- Crear una base de datos MySQL llamada tasks_users_db.
- Usar Sequelize para definir los modelos y crear las tablas automáticamente.

Modelo User:

- id (INTEGER, PRIMARY KEY, AUTO_INCREMENT)
- name (STRING(100), NOT NULL)
- email (STRING(100), UNIQUE, NOT NULL)
- password (STRING(100), NOT NULL)

Modelo Task:

- **id** (INTEGER, PRIMARY KEY, AUTO_INCREMENT)
- **title** (STRING(100),UNIQUE, NOT NULL)
- **description** (STRING(100), NOT NULL)
- **isComplete** (BOOLEAN, DEFAULT FALSE)

API Endpoints:

Rutas para Usuarios:

- **POST /api/users:** Crear un nuevo usuario.
- **GET /api/users:** Obtener todos los usuarios.
- **GET /api/users/:id:** Obtener un usuario específico por su ID.
- **PUT /api/users/:id:** Actualizar un usuario específico por su ID.
- **DELETE /api/users/:id:** Eliminar un usuario específico por su ID.

Rutas para Tareas:

- **POST /api/tasks:** Añadir una nueva tarea.
- **GET /api/tasks:** Obtener todas las tareas.
- **GET /api/tasks/:id:** Obtener una tarea específica por su ID.
- **PUT /api/tasks/:id:** Actualizar una tarea específica por su ID.
- **DELETE /api/tasks/:id:** Eliminar una tarea específica por su ID.

Validación de Datos:

- Validar los datos recibidos antes de añadir o editar una nueva tarea:
    - **title**: Debe ser un cadena única en la base de datos no vacía y de un máximo de 100 caracteres.
    - **description**: Debe ser una cadena no vacía y de un máximo de 100 caracteres.
    - **isComplete**: Debe ser un valor booleano.

- Validar los datos recibidos antes de añadir o editar un nuevo usuario:
    - **name**: Debe ser una cadena no vacía y de un máximo de 100 caracteres.
    - **email**: Debe ser un cadena única en la base de datos no vacía y de un máximo de 100 caracteres.
    - **password**: Debe ser una cadena no vacía y de un máximo de 100 caracteres.