Instituto Politécnico Formosa

Práctica de Relaciones con Sequelize

Objetivo:

Partiendo del proyecto “Práctica de Diagnóstico TLP2”, se deberán implementar relaciones entre los modelos existentes utilizando Sequelize. Además, será necesario crear nuevos modelos y establecer las relaciones correspondientes entre ellos.

Criterios de Evaluación

1. Presentación del código:
    - Código limpio, ordenado y bien indentado.
    - Uso obligatorio de **try-catch** en todos los controladores para manejo adecuado de errores.
    - 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 de los controladores (por ejemplo: existencia de usuario al crear tarea).
    - Uso correcto de Sequelize con conexión a MySQL y definición adecuada de modelos y relaciones.
    - Respuestas con mensajes claros y códigos HTTP apropiados (200, 201, 400, 404, 500, etc.).
    - Verificación de unicidad en creación y edición (cuando corresponda).
    - 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.

Teniendo en cuenta que en la práctica anterior las ramas **main** y **develop** contienen el mismo contenido, ahora se debe hacer lo siguiente:

- Crear una nueva rama a partir de la rama “**develop”** para guardar un versionado del estado actual, con el nombre “**crud-tasks**”.
- Crear otra rama a partir de “**develop”** llamada “**relaciones”**.
- Realizar el desarrollo del trabajo en la rama “**relaciones”**.
- Al finalizar el trabajo:
    - Hacer un merge limpio de “**relaciones**” hacia “**develop**”, sin conflictos.
    - Luego hacer un merge limpio de “**develop**” hacia “**main**”, para que ambas ramas queden sincronizadas nuevamente
- Durante el desarrollo en la rama **relaciones**, se deben realizar al menos 3 commits con mensajes claros y descriptivos.

Consignas:

Definir las siguientes relaciones con los modelos ya existentes de "User" y "Task":

- Un **usuario** puede tener muchas tareas.
- Cada **tarea** pertenece a un único usuario.

Se debe refactorizar los controladores para que funcionen en base a las nuevas relaciones:

- **POST /api/tasks:** Añadir una nueva tarea relacionada con un usuario.
- **GET /api/tasks:** Obtener todas las tareas con el usuario que las creó.
- **GET /api/tasks/:id:** Obtener una tarea específica por su ID con el usuario que la creó.
- **GET /api/users:** Obtener todos los usuarios con sus tareas.
- **GET /api/users/:id**: Obtener un usuario específico por su ID con sus tareas.

También se deben agregar nuevas validaciones para lo siguiente:

- No se pueden crear tareas sin un usuario.

Partiendo de las entidades desarrolladas crear nuevos modelos manteniendo un snake_case para definir relaciones de:

- Uno a uno.
- Muchos a muchos.

Para los nuevos modelos se debe desarrollar según correspondan las siguientes rutas:

- **POST /api/{nombre}:** Añadir un nuevo registro.
- **GET /api/{nombre}:** Obtener todos los registros con las relaciones anidadas y sus atributos esenciales.

.