Parcial

William Esteban Cruz Forero

1. Si vamos a sumar al modelo relacional la tabla para gestionar los mantenimientos de los equipos, ¿qué debemos hacer en el directorio models para crear Mantenimiento.js? Explicar, en síntesis.

2. En el archivo index.js del directorio models qué relaciones debemos incluir para la nueva tabla Mantenimiento. Enunciar la relación entre tablas.

3. Cuáles archivos se deben crear en los directorios public y views. Enunciar solamente su propósito y su relación

4. En el archivo server.js: a. Qué imports debemos sumar para llamar al modelo Mantenimiento b. Cuáles son los endpoints del CRUD Mantenimientos (API REST) que plantearías (enunciar y describir).

5. ¿Qué debemos sumar al home.js?

Desarrollo:

1. En el directorio models, debemos crear el archivo Mantenimiento.js donde definimos el modelo Sequelize correspondiente a la tabla Mantenimientos.
El modelo incluirá sus atributos, tipos de datos, y relaciones con otras tablas (por ejemplo, con Equipo o Responsable). Esto permite que Sequelize genere el mapeo ORM y podamos manipular la tabla desde el backend.

2. Debemos definir las relaciones entre Mantenimiento y Equipo (y opcionalmente con Responsable si aplica): Esto permite consultar mantenimientos por equipo y viceversa (Equipo.findAll({ include: Mantenimiento })).

3. **En public/:**

*Crear un archivo mantenimientos.js
→ Contendrá el código AJAX (fetch) para consumir los endpoints REST (/api/mantenimientos) y actualizar dinámicamente las tablas en el navegador (CRUD dinámico).

En views/:

*Crear mantenimientos.ejs
→ Es la vista SSR que renderiza la lista de mantenimientos, formularios o botones de acción (crear, editar, eliminar).
El servidor (server.js) la renderiza con datos del modelo Mantenimiento.

Relación:

El archivo .ejs muestra la interfaz (renderizada desde el servidor), y el .js en public/ gestiona las operaciones dinámicas con la API REST.

4. a) const { Ubicacion, Responsable, Equipo, mantenimientos } =
require('./models');

b)

GET /api/mantenimientos Devuelve todos los mantenimientos.

GET /api/mantenimientos/:id Devuelve un mantenimiento por ID.

POST /api/mantenimientos Crea un nuevo mantenimiento (datos en body).

PUT /api/mantenimientos/:id Actualiza un mantenimiento existente.

DELETE /api/mantenimientos/:id Elimina un mantenimiento por ID.

5. En home.js, debemos agregar el código necesario para mostrar un acceso o resumen de mantenimientos en la página principal, y/o un enlace hacia la vista SSR /mantenimientos.