Práctica 1

Desarrollo de una API REST para la gestión de incidencias y limpieza de las habitaciones de un hotel con Express y Mongoose

Parte 2. Definición de servicios REST

En esta parte de la práctica vamos a definir los servicios REST que accederán a la base de datos para realizar las diferentes operaciones solicitadas.

Instala *express* en la aplicación y crea una carpeta *routes* donde iremos ubicando cada uno de los enrutadores que se explican a continuación.

Enrutadores

En la carpeta *routes* definiremos los enrutadores de los modelos anteriores. En TODOS los servicios, se debe enviar:

- Un código de estado apropiado: 200 si todo ha ido bien, 400 si hay un fallo en la petición y 500 si hay un fallo en el servidor
- Un objeto JSON con estos atributos:
    - **ok**: de tipo booleano indicando si la petición se ha procesado satisfactoriamente o no
    - **error**: sólo estará presente si *ok* es falso. Contendrá el mensaje con el error que se haya producido
    - **resultado**: sólo estará presente si *ok* es verdadero. Contendrá el resultado que se envía como respuesta. Dicho resultado se detalla a continuación para cada servicio.

Enrutador para habitaciones

Este enrutador responderá a URIs que comiencen por /habitaciones. Se pide implementar estos servicios:

- **Obtener un listado de todas las habitaciones:**
    - Endpoint: GET /habitaciones
    - Descripción: devolverá como resultado todo el listado de habitaciones del hotel, con todos sus campos e incluyendo la información de las incidencias. Si no hubiera habitaciones registradas, se devolverá un código 500 con el mensaje "No hay habitaciones registradas en la aplicación".
- **Obtener detalles de una habitación específica:**
    - Endpoint: GET /habitaciones/:id
    - Descripción: devolverá como resultado el detalle de una habitación a partir de su id, incluyendo sus incidencias. Si no se encuentra o el *id* es incorrecto se devolverá un error 400 con el mensaje "No existe el número de habitación".
- **Insertar una habitación**
    - Endpoint: POST /habitaciones
    - Descripción: se añadirá la habitación que se reciba en la petición a la colección habitaciones. En dicha petición se enviarán los campos básicos de la habitación (es decir, todo menos el id y el array de incidencias). Se devolverá como resultado el detalle de la habitación insertada, o un código 400 con el mensaje "Error insertando la habitación", si se ha producido algún error.

- **Actualizar los datos de una habitación**
    - Endpoint: PUT /habitaciones/:id
    - Descripción: permitirá modificar los datos básicos de una habitación (todo salvo el id y el array de incidencias). De nuevo, se enviarán en la petición los campos básicos de la habitación, como en POST. Se devolverá como resultado el detalle de la habitación modificada, o un código 400 con el mensaje "Error actualizando los datos de la habitación" en caso de producirse un error o no encontrarse la habitación indicada.
- **Eliminar una habitación**
    - Endpoint: DELETE /habitaciones/:id
    - Descripción: permitirá eliminar una habitación, junto con sus incidencias, a partir de su id. Como resultado se devolverá la habitación eliminada con todos sus campos, o un error 400 y el mensaje "Error eliminando la habitación" en caso de error o de no encontrarse la habitación
- **Añadir una incidencia en una habitación:**
    - Endpoint: POST /habitaciones/:id/incidencias
    - Descripción: se modificará el array de incidencias de la habitación con el id indicado. Se enviará en la petición la descripción de la incidencia, como problemas de fontanería, electricidad, etc. El campo *inicio* se rellenará de forma automática con la fecha actual, y el campo *fin* se deja sin rellenar. Se devolverá como resultado el detalle de la habitación, o un código 400 y el mensaje "Error añadiendo la incidencia", en caso de error.
- **Actualizar el estado de una incidencia de una habitación:**
    - Endpoint: PUT /habitaciones/:idHab/incidencias/:idInc
    - Descripción: se recorrerá el array de incidencias de la habitación con *idHab*, buscando la incidencia *idInc*. Si se encuentra, se pondrá como fecha de finalización la fecha actual. Se devolverá como resultado el detalle de la habitación, incluyendo las incidencias, o el código 400 con el mensaje “Incidencia no encontrada” si no se pudo actualizar la incidencia.
- **Actualizar última limpieza**
    - Endpoint: PUT /habitaciones/:id/ultima
    - Descripción: se actualizará el campo *ultimaLimpieza* de la habitación con la fecha de la última limpieza realizada para esa habitación (consultar en la colección de “limpiezas”). Se devolverá el detalle de la habitación, o el código 400 con el mensaje “Error actualizando limpieza” si ha habido algún error.

Enrutador para limpiezas

Este enrutador responderá a URIs que comiencen por /limpiezas. Se pide implementar estos servicios:

- **Obtener limpiezas de una habitación**
    - Endpoint: GET /limpiezas/:id
    - Descripción: se obtendrá el listado de limpiezas de la habitación cuyo *id* se especifique, ordenado de fecha más reciente a más antigua. Si no hubiera limpiezas registradas, se devolverá un código 500 con el mensaje "No hay limpiezas registradas para esa habitación "
- **Obtener el estado de limpieza de una habitación:**
    - Endpoint: GET /limpiezas/:id/estado
    - Descripción: Devuelve el estado de limpieza actual de una habitación específica, como "limpia" o "pendiente de limpieza", dependiendo de si la fecha de *ultimaLimpieza*

coincide con la fecha actual o no. Se devolverá el código 400 con el mensaje “Error obteniendo estado de limpieza” si no se encuentra la habitación en cuestión.

- **Actualizar limpieza**
  - Endpoint: POST /limpiezas/:id
  - Descripción: se añadirá la limpieza a la habitación (en la colección de *limpiezas*), con las observaciones que se reciban en la petición (si las hay). La fecha de limpieza se asignará automáticamente a la fecha actual. Se devolverá como resultado el registro de limpieza insertado, o un código 400 con el mensaje "Error actualizando limpieza", si se ha producido algún error.

Servidor principal

Se deberá implementar en el archivo *index.js* la funcionalidad general del servidor principal:

- Carga de librerías/módulos necesarios
- Conexión con la base de datos *hotel*.
- Carga de enrutadores
- Puesta en marcha del servidor

Prueba de los servicios

Se recomienda definir una colección de pruebas (en *ThunderClient, Postman* o una herramienta similar) para verificar el correcto funcionamiento de cada uno de estos servicios.