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 3. Autenticación y pruebas completas

En esta parte de la práctica añadiremos mecanismos de autenticación basada en tokens a nuestra aplicación, y pasaremos una batería de pruebas para comprobar su funcionamiento.

Instala *jsonwebtoken* en la aplicación y crea una carpeta *auth* donde definiremos un archivo con todas las funciones necesarias para la gestión de la autenticación (generar token, validar token, etc).

Autenticación

La gestión de la autenticación la dividiremos en tres pasos:

Funciones auxiliares de autenticación

Define en la carpeta *auth* un archivo llamado *auth.js* que contenga las funciones necesarias para gestionar la autenticación basada en tokens. Al menos deberá haber:

- Un método para generar un token dado un *login* de usuario válido
- Un método para validar un token que se reciba
- Un método para proteger una ruta, de modo que sólo se pueda acceder a ella si se tiene un token válido. Si no se tiene un token válido, se debe enviar un código 403 (Acceso prohibido), junto con el booleano *ok* a falso y el mensaje *"Acceso no autorizado"* (campo *error*).

Enrutador para usuarios

Este enrutador lo ubicaremos en el fichero *routes/auth.js* y responderá a URIs que comiencen por /auth. Se pide implementar estos servicios:

- **Loguear a un usuario**
  - Endpoint: POST /auth/login
  - Descripción: se obtendrá de la petición el login y password del usuario y, si es correcto, se le devolverá un código 200 con el token de acceso correspondiente (campo *resultado*). En caso contrario se le enviará un código de error 401 (no autorizado) con el mensaje "Login incorrecto" (campo *error*).

Protección de rutas

Añade los mecanismos de autenticación necesarios para que los servicios que impliquen modificar colecciones sólo puedan hacerse estando validado. En concreto nos referimos a los siguientes *endpoints*:

- POST /habitaciones
- PUT /habitaciones/:id
- DELETE /habitaciones/:id
- POST /habitaciones/:id/incidencias
- PUT /habitaciones/:idHab/incidencias/:idInc
- PUT /habitaciones/:id/ultima

- POST /limpiezas/:id

Pruebas

La aplicación se probará con distintos casos de prueba de los diferentes servicios. Se os proporcionará una base (simple) de ejemplos de prueba que podéis usar en vuestro desarrollo, aunque posteriormente se probará con más casos.

Criterios de calificación

La calificación final de la práctica se realizará según estos criterios:

1. Estructura general de la aplicación (carpetas *models, routes* y *auth* con el contenido adecuado, fichero *package.json* con las dependencias correctamente instaladas y fichero *index.js* con el programa principal): **0,5 puntos**. Se incluye en este apartado la gestión de un fichero de configuración de variables de entorno *.env* que, al menos, almacene y utilice correctamente las siguientes variables de entorno:
    a. Puerto de conexión
    b. URL de acceso a la base de datos
    c. Palabra secreta para cifrar los tokens
2. Definición de modelos y esquemas (parte 1 de la práctica): **2 puntos**, repartidos como sigue
    a. Esquema y modelo de habitación (incluyendo subdocumentos de incidencias): 1 punto
    b. Esquema y modelo de limpieza (incluyendo relación con habitación): 0,75 puntos
    c. Esquema y modelo de usuarios: 0,25 puntos
3. Enrutadores: **3 puntos**. Se calificará cada servicio con 0,25 puntos.
4. Autenticación: **1,5 puntos**, repartidos de este modo:
    a. Funciones auxiliares de autenticación correctas: 0,75 puntos
    b. Protección de los servicios necesarios con el middleware de autenticación: 0,75 puntos
5. Servidor principal: **1 punto** (secuenciación correcta de los pasos para poner en marcha el servidor)
6. Pruebas: **2 puntos**. Se probará la aplicación con el conjunto simple de pruebas que se os facilitará en esta etapa, más otras pruebas adicionales que se añadirán. Se descontará -0,25 puntos por cada caso fallido (hasta un máximo de 2 puntos)

Penalizaciones y requisitos a tener en cuenta para superar la práctica

Será condición indispensable para superar la práctica alcanzar **al menos la mitad de puntuación** en los apartados 2, 3 y 6 de los criterios anteriores. Además, en caso de que el profesor considere oportuno realizar una prueba oral a cualquier alumno/a sobre su práctica, será necesario responder correctamente al menos a la mitad de las preguntas que el profesor pueda hacer.

Además, se aplicarán las siguientes **penalizaciones** sobre la nota final:

- Si algún servicio no recibe o devuelve los atributos con el nombre que se indica, o no responde a la URI indicada, se calificará con 0 puntos, sin tener en cuenta si su código es o no correcto.
    - **Ejemplo 1**: si el servicio *POST /habitaciones* espera recibir el atributo **numero** de la habitación y recibe un atributo **num** en su lugar, se calificará con un 0.

  - **Ejemplo 2**: si el servicio *GET /habitaciones* devuelve la respuesta en un campo **result** en lugar del correcto **resultado**, el servicio se calificará con un 0.
  - **Ejemplo 3**: si el servicio de borrado (DELETE) atiende a la URI *habitaciones/borrar/:id* en lugar de */habitaciones/:id*, también se calificará con un 0.
- La no eliminación de la carpeta *node_modules* en el ZIP de entrega se penalizará con 1,5 puntos menos de nota global de la práctica. Esta penalización se verá incrementada en posteriores prácticas.
- Si no se sigue la estructura de proyecto propuesta (nombres de archivos y carpetas) y no se justifica debidamente esta decisión, se penalizará la calificación global de la práctica hasta un 50% de la nota.