Práctica 3

Despliegue de aplicaciones web con Node.js en servidores remotos

Parte 1. API REST con Nest.js

Para esta primera parte de la práctica vamos a desarrollar una API REST sencilla con el framework Nest.js, basándonos en parte de la especificación de la Práctica 1 del curso. Dividiremos el trabajo en tres partes más o menos independientes entre sí, de forma que se puedan coordinar en un equipo formado por 2-3 personas, y que cada una haga una de estas partes.

Todo el proyecto deberá compartirse por el equipo en un repositorio GitHub (público), de modo que todos puedan subir cambios a él, y desde ese repositorio se pueda descargar el proyecto completo.

Tarea 1: definición base del proyecto Nest

La persona encargada de esta tarea deberá crear el proyecto Nest y definir los servicios básicos que se indican en este apartado.

Proyecto Nest: pasos iniciales

Para comenzar, crearemos un proyecto Nest.js llamado ***hotel-nest***. Internamente crearemos el paquete de recursos completos (*res* o resources) para el módulo de limpiezas (con el nombre *limpieza*). Esto incluirá:

- El módulo de limpiezas en *src/limpieza/limpieza.module.ts*
- El controlador de limpiezas en *src/limpieza/limpieza.controller.ts*
- El servicio de limpieza en *src/limpieza/limpieza.service.ts*
- La entidad de limpieza en *src/limpieza/entities/limpieza.entity.ts*
- Los DTO de creación y actualización en la subcarpeta *src/limpieza/dto*

Instala también Mongoose y configúralo en el proyecto para conectar a la misma base de datos de prácticas anteriores.

Entidades y esquemas

Definiremos el esquema de limpiezas en la entidad correspondiente *src/limpieza/entities/ limpieza.entity.ts*. Además, necesitaremos otra entidad para el esquema y modelo de habitación, ya que ambas colecciones están relacionadas. Así que también necesitamos crear a mano el fichero ***src/habitacion/entities/habitacion.entity.ts***.

Define ahora dentro de cada entidad su esquema correspondiente, basándote en la especificación de la práctica 1. No incluyas las incidencias como campo del esquema de habitaciones, ya que no las utilizaremos en esta práctica. Incluye finalmente los dos modelos (habitaciones y limpiezas) en el módulo de limpiezas *limpiezas.module.ts*.

Servicios a desarrollar

En el controlador y servicio de limpieza se propone completar los siguientes servicios:

- **GET /limpieza/:id**: obtiene el listado de todas las limpiezas de la habitación con el *id* indicado, ordenadas por fecha descendente. Si no existen limpiezas se devolverá el array vacío
- **POST /limpieza**: inserta una nueva limpieza para una habitación. En el cuerpo de la petición se enviarán todos los datos necesarios *(id* de habitación, fecha de limpieza y, si es necesario, observaciones, todos deben tener el mismo nombre que el campo correspondiente en la base de datos y esquema). Se devolverá el objeto de limpieza insertado si todo ha ido bien, o un *Bad Request* (código 400) en caso de error. Define los mecanismos de validación adecuados en el DTO de creación.
    - PISTA: usa el validador *IsDateString* para la fecha, ya que admite fechas parciales (*yyyy-mm-dd).* En este caso, no se admite campo *message* para el error de validación.
    - PISTA: usa el validador *IsMongoId* para verificar el *id* de la habitación.
- **PUT /limpieza/:id:** modifica los datos de una limpieza dado su *id* (el de la limpieza). En el cuerpo de la petición puede recibir tanto una nueva fecha como unas nuevas observaciones, y se actualizarán sólo los campos pertinentes. Define los mecanismos de validación adecuados para que se genere un código *400 – Bad Request* si la actualización no es correcta. Se devolverá el código *404 – Not Found* si no se encuentra la limpieza a actualizar.
- **GET /limpieza/limpia/:id**: devuelve un JSON con un campo *ok* a *true* o *false* dependiendo de si la habitación con el *id* indicado se ha limpiado en el día de hoy o no. Para ello consultaremos únicamente el campo *ultimaLimpieza* de la habitación.
- **GET /limpieza/limpias***:* devuelve un listado con las habitaciones que se han limpiado en el día de hoy. Si no hay, se devolverá un array vacío.

Tanto para el servicio POST como para el servicio PUT se debe actualizar la fecha de última limpieza de la habitación (documento de la colección de habitaciones) a la limpieza más reciente de la colección de limpiezas.

Elimina del controlador y el servicio aquellos métodos que no vayas a utilizar, como por ejemplo el de borrado. A cambio, necesitarás añadir y modificar métodos para servicios que no son los habituales.

Tarea 2: Autenticación por tokens

La segunda tarea a desarrollar será proteger mediante autenticación por tokens el acceso a los servicios POST y PUT, que implican una modificación de los documentos en la base de datos.

Para los usuarios deberás utilizar la colección de usuarios que ya existe en la base de datos para entregas anteriores. Crea una entidad ***src/usuario/entities/usuario.entity.ts*** y cárgala en el módulo de autenticación para gestionar los accesos. Esta entidad sustituye a la interfaz básica que se tiene en los ejemplos de los apuntes, que se utilizaba para crear un array simple de usuarios. Crea también el DTO de usuario como el de los apuntes, para recibir sus datos en el POST de *login*.

Tarea 3: Conjunto de tests

La tercera tarea consistirá en definir un conjunto de tests lo más completo posible usando la librería Axios. Incluirá, al menos, las siguientes pruebas:

- Obtener limpiezas de una habitación
- Obtener si una habitación está limpia o no
- Obtener el listado de habitaciones que se han limpiado el día de hoy

- Login correcto e incorrecto
- Intentar insertar una limpieza sin login correcto
- Insertar limpieza con login correcto
- Actualizar limpieza sin login correcto
- Actualizar limpieza con login correcto

Para algunas de las pruebas (como por ejemplo obtener si una habitación está limpia o no, o el listado de habitaciones que se han limpiado hoy) puede que haga falta alguna prueba o función auxiliar que inserte alguna limpieza en alguna habitación en la fecha actual.

Para entregar

Como entrega de esta práctica, cualquiera de los miembros del equipo deberá facilitar la URL (pública) del repositorio de GitHub para poder descargar de él el proyecto completo. Se indicará también el nombre de los integrantes del grupo, y qué tarea ha hecho cada uno de ellos.

Criterios de calificación

Cada una de las tareas se calificará como sigue:

TAREA 1:

- Estructura del proyecto y configuración de Mongoose: 1
- Esquemas de limpieza y habitación: 1,5
- GET /limpieza/:id 1,5
- POST /limpieza 1,5
- PUT /limpieza/:id 1,5
- GET /limpieza/limpia/:id 1,5
- GET /limpieza/limpias 1,5

TAREA 2:

- Módulo de usuarios (sin incluir la entidad) 2
- Entidad de usuarios, debidamente asociada al esquema 2
- Módulo de autenticación 2
- Configuración de JWT 2
- Proteger recursos solicitados 2

TAREA 3:

- Obtener limpiezas de habitación 1
- Obtener si una habitación está limpia o no 2
- Obtener listado de habitaciones limpiadas hoy 2
- Login correcto e incorrecto 1
- Intentar insertar una limpieza sin login correcto 1
- Insertar limpieza con login correcto 1
- Actualizar limpieza sin login correcto 1
- Actualizar limpieza con login correcto 1

Cada apartado se calificará con la siguiente escala:

B: 100% B-: 75% R: 50% R-: 25% M: 0%