Práctica Tema 2

Desarrollo de una API REST para productos asturianos con Express y Mongoose

En esta práctica vamos a ampliar lo hecho en la práctica del tema anterior, para construir una API REST más compleja y completa sobre la aplicación de productos asturianos, pero esta vez utilizando una base de datos MongoDB y la API que ofrece la librería de Mongoose.

1. Estructura de la aplicación

Crearemos una aplicación llamada **ProdAsturianos_V2**, e instalaremos dentro los módulos de *express*, *mongoose* y *body-parser* (recordad que el módulo body-parser solo será necesario instalarlo para versiones de Express inferiores a 4.16). La aplicación estará estructurada en las siguientes carpetas y archivos (además del archivo `package.json` que generaremos con `npm init`):

- Archivo `index.js` en la raíz del proyecto, donde irá el servidor principal.
- Carpeta `models` donde almacenaremos los esquemas y modelos de la aplicación. Concretamente tendrá un archivo: `producto.js`, que completaremos después.
- Carpeta `routes` con el enrutador para la colección de productos. Contendrá el archivo `productos.js`.

2. Definiendo esquemas y modelos

En la carpeta `models` almacenaremos la definición de esquemas y modelos de nuestra base de datos. En concreto será uno, con los campos que se indican a continuación (se deben respetar los nombres de los campos, que se marcan en negrita, incluyendo mayúsculas o minúsculas si procede):

Modelo *producto*

En el archivo `models/producto.js` definiremos el esquema para la colección de productos. Cada producto tendrá los siguientes campos:

- El **nombre** del producto (obligatorio, de tipo texto, con una longitud mínima de 3 caracteres).
- El **precio** del producto (numérico obligatorio, mayor que 0).
- La **descripcion** del producto (texto largo obligatorio, sin tamaño mínimo).
- La **imagen** del producto (texto con la ruta relativa, no obligatoria).
- Unos **comentarios** realizados por los usuarios, dando su opinión sobre el producto. Este campo será un array de subdocumentos del tipo de esquema que comentaremos a continuación.

Para los comentarios, definiremos un esquema local a este modelo, que contendrá estos campos:

- El **nombre del usuario** que realiza el comentario. Será de tipo texto obligatorio.

- El **comentario** sobre el producto. Será de tipo texto obligatorio, con tamaño mínimo de 5.

Recuerda exportar el modelo.

3. Los enrutadores

En la carpeta `routes` definiremos el enrutador del modelo anterior. 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 *productos*

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

- `GET /productos` : devolverá como `resultado` todo el listado de productos, con todos sus campos, incluyendo la información de los comentarios. Si no hubiera productos, se devolverá un código 500 con el mensaje `"No se encontraron productos"` .

- `GET /productos/:id` : devolverá como `resultado` la ficha para una producto a partir de su *id*. Si no se encuentra, se devolverá un error 400 con el mensaje `"Producto no encontrado"` .

- `POST /productos` : añadirá el producto que se reciba en la petición a la colección. En dicha petición se enviarán los campos básicos del producto (es decir, todo menos el *id* y el array de comentarios). Se devolverá como `resultado` el producto insertado, o un código 400 con el mensaje `"Error insertando el producto"` , si ha habido algún error.

- `PUT /productos/:id` : permitirá modificar los datos básicos del producto (todo salvo el *id* y el array de comentarios). De nuevo, se enviarán en la petición los campos básicos del producto, como en POST. Se devolverá como `resultado` el producto modificado, o un código 400 con el mensaje `"Error modificando el producto"` .

- `POST /productos/comentarios/:idProducto` : permitirá modificar el array de comentarios del producto con el *id* indicado, añadiendo el nuevo comentario que se envíe en la petición. Se enviarán en la petición los campos del nuevo comentario, y se devolverá como resultado el producto entero modificado, o un código 400 y el mensaje `"Error modificando los comentarios del producto"` , en caso de error.

- `DELETE /productos/:id` : permitirá eliminar un producto, junto con sus comentarios, a partir de su *id*. Como `resultado` se devolverá el producto eliminado con todos sus campos, o un error 400 y el mensaje `"Error eliminando el producto"` .

- `DELETE /productos/comentarios/:idProducto/:idComentario`: permitirá eliminar el comentario con el *id* de comentario indicado, para el producto con el *id* de producto indicado. Devolverá como `resultado` el producto en su estado actual, o un error 400 y el mensaje `"Error eliminado el comentario del producto"`.

4. El servidor principal

El servidor principal deberá:

- Cargar las librerías necesarias (al menos, *express*, *mongoose* y *body-parser*)
- Cargar los enrutadores de productos
- Conectar a una base de datos "productos" de MongoDB
- Inicializar Express, y aplicar el middleware de *body-parser* para procesar datos en formato JSON, y asociar los enrutadores respectivamente a las rutas de `/productos`.
- Poner en marcha el servidor Express por el puerto 8080.

5. Pruebas con Postman

Se pide, además, que elaboréis una colección de pruebas Postman llamada **ProdAsturianos_V2**, para probar cada uno de los servicios indicados anteriormente. Expórtala a un archivo con el mismo nombre.

6. Entrega y calificación

Deberéis entregar un archivo ZIP o similar, con vuestro nombre y el prefijo "PracT2". Por ejemplo, si os llamáis José Pérez, el archivo de entrega deberá ser `PracT2_Jose_Perez.zip`. Dentro, deberá contener:

- El proyecto **ProdAsturianos_V2** de Node, sin que contenga la carpeta `node_modules`.
- El archivo de Postman exportado, con las pruebas de la colección.

6.1. Calificación de la práctica

Los criterios para calificar esta práctica son los siguientes:

- Estructura correcta del proyecto, con las carpetas y nombres de archivos indicados en el enunciado, archivo `package.json` correctamente definido con las dependencias incorporadas: **0,75 puntos**
- Modelo de datos: **2,5 puntos**:
    - Esquema y modelo básico para los productos (sin los comentarios): **1,5 punto**
    - Esquema local para los comentarios: **1 puntos**
- Enrutadores: **3,5 puntos**, repartidos así:
    - Servicios implementados: **0,5 puntos** cada servicio (7 en total)
- Funcionalidad del archivo principal `index.js`: **1 puntos**
- Colección Postman, con los servicios correctamente añadidos para probarse: **1,75 puntos** (*0,25 puntos* para cada petición)

- Claridad y limpieza del código, y uso de un comentario inicial en cada fichero fuente explicando qué hace: **0,5 puntos**.

Penalizaciones a tener en cuenta

- Si algún servicio no recibe o devuelve los atributos con el nombre indicado, o no responde a la URI indicada, se calificará con **0 puntos**, independientemente de lo bien o mal que esté su código

> **Ejemplo**: si en el servicio `POST /productos` esperamos recibir un campo `nombreProducto` en lugar del original `nombre` que figura en el esquema, el servicio se calificará con un 0.

> **Ejemplo 2**: si en el servicio `GET /productos` enviamos en el objeto de respuesta un atributo `result`, en lugar de uno `resultado`, el servicio se calificará con un 0.

> **Ejemplo 3**: si en el servicio de borrado (DELETE) se responde a la URI `/productos/borrar/id` en lugar de a `/productos/id`, el servicio se calificará con un 0.

- La no eliminación de la carpeta `node_modules` en el archivo ZIP de entrega se penalizará con 1,5 puntos menos de nota global de la práctica.

- Si se sigue una estructura de proyecto, código y/o nombres de archivo distinta a la propuesta, y no se justifica debidamente, o dicha justificación no es satisfactoria, se penalizará la calificación global de la práctica con hasta el 50% de la nota de la misma.

- En el apartado de calificación de los **enrutadores**, se deberán obtener **al menos 1,75 puntos** del total de 3,5 para considerar aprobada la práctica (además de llegar a la nota mínima exigible).