Práctica de Back-end

Crear servicio web REST

En esta práctica crearemos el **back-end** de la aplicación de **e-commerce** que hemos estado desarrollando en las demás prácticas. Crearemos con **Node.js** nuestra **REST API** que permitirá crear, modificar, editar, consultar y borrar los productos, para facilitar las cosas guardaremos la información en archivos JSON y realizaremos una autenticación muy básica. Para validar nuestro servicio web, lo haremos únicamente con solicitudes de HTTP para realizar operaciones de altas, bajas, cambios y consultas usando **PostMan**.

Descripción de la práctica

Crear una REST API para nuestra aplicación de **e-commerce**. Lo cual implicará realizar lo siguiente.

1. Consultar productos
2. Consultar producto en especifico
3. Consultar productos según una lista de IDs
4. Registro de nuevos productos (con validaciones)
5. Editar información de un producto
6. Borrar algún producto
7. Filtro de productos (obtener todos o algunos según los parámetros)
8. Manejo de errores de respuesta de HTTP
9. Validación de headers usando middlewares

Endpoint	Método	headers	body	Comentarios
/products/	GET			Si se pasa el parámetro “query” aplicar el filtro correspondiente.
/products/:id	GET			Regresar el producto correspondiente. (mandar status 200). Si el ID no coincide mandar status 404.
/products/cart	POST	content-type: application/json	Arreglo de objetos con los atributos: productUuid, amount	Regresar un arreglo con los productos correspondientes. (mandar status 200) Si body no es un arreglo mandar status 400. Si algún ID no coincide mandar status 404.
/admin/products/	POST	x-auth content-type: application/json	uuid, imageUrl, title, description, unit, category, pricePerUnit, stock	Crear el producto correspondiente, en caso de éxito mandar status 201, de lo contrario mandar 400.

/admin/products/:id	PUT	x-auth content-type: application/json	imageUrl, title, description, unit, category, pricePerUnit, stock	Actualizar el producto correspondiente, en caso de éxito mandar status 200, de lo contrario mandar 400. Si el ID no coincide mandar status 404.
/admin/products/:id	DELETE	x-auth		Borrar el producto correspondiente, en caso de éxito mandar status 200. Si el ID no coincide mandar status 404.

Antes de iniciar

Crea una carpeta que se llame practica3 en la consola pon **npm init --yes** para crear el package.json. **Instala como dependencia express**. Crea un archivo **products.json** donde guardaremos la información de los productos inicialmente que tenga un arreglo vacío **[ ]**.

Crea un archivo **server.js y data_handler.js** e **importa express y fs** respectivamente.

En una variable **products** carga y parsea el archivo '*products.json*'

Carga de forma global el middleware que permite parsear el json (**express.json()**)

Crea una ruta raíz que solo regresa "*e-commerce app práctica 3*"

Levanta el servicio en el puerto 3000.

¡Prueba que te funciona!

Estructura de Archivos Sugerida

```
|-- Practica3/
  |-- app/
    |-- controllers
      |-- data_handler.js
      |-- product.js
      |-- router.js
      |-- shopping_cart.js
      |-- utils.js
    |-- data/
      |-- products.json
    |-- routes/
      |-- products.js
      |-- admin_products.js
    |-- views/
      |-- home.html
      |-- shopping_cart.html
```

GET /products

Lo primero que realizaremos será regresar la lista de productos, para lo cual los apoyaremos de los archivos realizado en la Práctica 2. En caso de haber implementado la parte opcional de filtros aquí podremos agregar el acceso para regresar la lista filtrada.

1. Regresar todos los productos: debemos regresar los valores que ya estén cargados en nuestro arreglo de productos (**products**).
2. Regresar lista filtrada de productos: verificar si la petición cuenta con parámetros de búsqueda, en cuyo caso mandaremos filtrar la lista y regresar solo los valores deseado (puedes consultar la Práctica 2 para ver los parámetros de búsqueda).

POST /products/cart

Para poder cargar los productos correspondientes en nuestro carrito de compras, debemos permitir la búsqueda de productos directamente por su ID.

1. Tendremos que validar que el body recibido sea de tipo arreglo, de lo contrario debemos regresar un estatus de error (400).
2. Iteraremos por el arreglo recibido e iremos agregando los elementos correspondientes a una lista temporal de productos, en caso de que no se encuentre algún producto, debemos regresar un estatus de error (404) y un mensaje explicando que no se encontró dicho producto.
3. Al terminar si todos los productos fueron encontrados, regresamos la lista temporal con un estatus de éxito (200). Recuerda agregar a la respuesta el header correspondiente para JSON.

GET /products/:id

Debemos permitir la búsqueda de productos directamente por su ID.

1. En caso de que el ID sea inválido, debemos regresar un estatus de error (404) con un mensaje explicando que no se encontró el producto.
2. En caso de que todo este correcto, regresa un estatus de éxito (200) y el producto correspondiente. Recuerda agregar a la respuesta el header correspondiente para JSON.

Middleware de autenticación

1. Crear un middleware llamado **validateAdmin** que verifica que exista el header ***x-auth*** con el valor ***"admin"***. Mandar error 403 en caso de que no exista "Acceso no autorizado, no se cuenta con privilegios de administrador".

POST /admin/products

Para la parte de administrador, lo primero que realizaremos será trabajar con el registro de un producto. Antes de continuar no olvides **poner el middleware de autenticar**. Para hacer el registro debemos hacer varias cosas:

1. Validar que en el body se encuentren los atributos que nos interesan en caso contrario regresar **bad request (400)** e indicar en el mensaje de respuesta cuales atributos faltan. Apóyate del try/catch y las validaciones de la Práctica 2.
2. Asegúrate que el ID del producto, se autogenere usando UUIDs.
3. Guardar el nuevo producto en el arreglo y regresar el status 201 y como mensaje el nombre del producto creado.
4. No te olvides de actualizar el archivo de productos.

PUT /admin/products/:id

Permitiremos al administrador modificar los atributos de algún producto. **No olvides poner el middleware de autenticar**

1. Verifica que el ID corresponda a un producto existente, de lo contrario regresa un estatus de error (404).
2. Validar que en el body se encuentren los atributos que nos interesan similar al POST.
3. Guardar los cambios al producto en el arreglo y regresar el status 200, con un mensaje con el nombre del producto que fue actualizado.
4. No te olvides de actualizar el archivo de productos.

DELETE /admin/products/:id

Permitiremos al administrador borrar un producto específico. **No olvides poner el middleware de autenticar**

1. Verifica que el ID corresponda a un producto existente, de lo contrario regresa un estatus de error (404).
2. Guardar los cambios en el arreglo y regresar el status 200, con un mensaje con el nombre del producto que fue borrado.
3. No te olvides de actualizar el archivo de productos.

GET /, /home, /shopping_cart

Investiga como mostrar los archivos HTML de la Práctica 1, de manera que:

1. Al ingresar a la ruta raíz (**/**) o a la ruta de home (**/home**) se muestre **home.html**.
2. Al ingresar a la ruta del carrito de compras (**/shopping_cart**) se muestre **shopping_cart.html**.

Evaluación

Requerimiento (incluye validaciones, regresar código de error y que funcione correctamente).	Valor	Realizado (Si/No)
1. GET /products - Filtro por parámetros de búsqueda.	15	
2. POST /products/cart	15	
3. GET /products/:id	10	
4. Middleware de autenticar	10	
5. POST /admin/products	10	
6. PUT /admin/products/:id	15	
7. DELETE /admin/products:id	10	
8. GET /, /home, /shopping_cart	10	
9. Código estructurado y modularizado - Archivo de router global - Archivo de router para productos - Archivo de router para endpoints de administrador de productos	10	
10. Manejo de archivos	10	

Máxima calificación de la práctica: 100 puntos.

Entregables:

- Entregar todos los archivos JS y HTML, así como una colección (JSON) de PostMan o un archivo HTTP de REST Client, validando el funcionamiento de cada Endpoint.
- Debes además adjuntar un reporte con las evidencias y la rúbrica contestada de acuerdo a lo que se alcanzó a entregar.
- En el reporte añadir conclusiones, cosas por mejorar, temas que costaron trabajo, etc…