Notas Coder Proyecto backend I

Repositorio GIT: coderproyectofinalbackenddpadin

Consigna

Desarrollar el servidor basado en Node.JS y express, que escuche en el puerto 8080 y disponga de dos grupos de rutas: /products y /carts. Dichos endpoints estarán implementados con el router de express, con las siguientes especificaciones:

Para el manejo de productos, el cual tendrá su router en /api/products/ , configurar las siguientes rutas:

* La ruta raíz GET / deberá listar todos los productos de la base. (Incluyendo la limitación ?limit del desafío anterior
* La ruta GET /:pid deberá traer sólo el producto con el id proporcionado

La ruta raíz POST / deberá agregar un nuevo producto con los campos:

id: Number/String (A tu elección, el id NO se manda desde body, se autogenera como lo hemos visto desde los primeros entregables, asegurando que NUNCA se repetirán los ids en el archivo.

title:String,

description:String

code:String

price:Number

status:Boolean

stock:Number

category:String

thumbnails:Array de Strings que contenga las rutas donde están almacenadas las imágenes referentes a dicho producto

Status es true por defecto.

Todos los campos son obligatorios, a excepción de thumbnails

**CARRITO**

Para el carrito, el cual tendrá su router en /api/carts/, configurar dos rutas

* La ruta raíz POST / deberá crear un nuevo carrito con la siguiente estructura:
* Id:Number/String (A tu elección, de igual manera como con los productos, debes asegurar que nunca se dupliquen los ids y que este se autogenere).
* products: Array que contendrá objetos que representen cada producto
* La ruta GET /:cid deberá listar los productos que pertenezcan al carrito con el parámetro cid proporcionados.
* La ruta POST /:cid/product/:pid deberá agregar el producto al arreglo “products” del carrito seleccionado, agregándose como un objeto bajo el siguiente formato:
* product: SÓLO DEBE CONTENER EL ID DEL PRODUCTO (Es crucial que no agregues el producto completo)
* quantity: debe contener el número de ejemplares de dicho producto. El producto, de momento, se agregará de uno en uno.

Además, si un producto ya existente intenta agregarse al producto, incrementar el campo quantity de dicho producto.

Inicializar el Proyecto

1. npm init -y

En package.json se edito:

  "type": "module",

Scripts:

  "start": "node scr/index.js",

   "dev" : "node --watch app.js"

1. npm install moment
2. agregar .gitignore y agregar dentro /mode\_modules
3. npm install express.
4. Crear carpeta src.
5. Crear index.js dentro de la carpeta src
6. En index.js configure express:
   1. Imprt express from “express”
   2. Const app = express()
7. Comienzo a configurar los “endpoints”
   1. Endpoint products
   2. Endpoint carts
8. Metdo POST, uso:

app.use(express.json())

1. Agrego un endpoint tipo POST,PUT y DELETE: para product y carts
2. Ruta para agregar/modificaar un prducto en un carrito:
   1. (GET )http://localhost:8080/carts/20/product/100
3. Generar un I unico para el product.
   1. npm install uuid
4. import {v4 as uuidv4} from 'uuid';
5. let myuuid = uuidv4();
6. se crea en un archive separado (archive.js) la function
   1. **newAsyncProduct** = async (product)
   2. la function crea el archive “productos.json” si es que no existe, si existe lo carga, y luego de agregar el ultimo product, lo Vuelve a grabar.
7. Formato del query para ver productos:  
   a- todos los productos: <http://localhost:8080/products>
8. Con limites: http://localhost:8080/products**?limit=2**
9. Ruta /GET/pid creada, usa la function **listOneAsyncProduct**  = async (pid) del archivos.js
   1. Devuleve un objeto con el product, o objeto vacio si no lo encuentra
   2. Formato del request:
      1. [http://localhost:8080/products/**446662fc-4876-4e94-8473-2ca4014e2d5a**](http://localhost:8080/products/446662fc-4876-4e94-8473-2ca4014e2d5a) **<<= pid**
10. Ruta “PUT” ha sido creada, mediane la function: **updateAsyncProduct**  = async (pid,newData)
    1. Donde pid es el id de product, enviado mediante la ruta, “newData” es tomado del body con los valores a ser actualizados. Id no es recibido y NO se puede actualizer.
11. Ruta “DELETE” creada exitosamente, mediante la function **deleteAsyncProduct = async (pid)**
    1. Donde pid es el id de producto.
12. Ruta /api/carts agregada con exito. Se creo funcion **newAsyncCart** = async (cid,products)
    1. cid se usa para mostrar mensajes, en products se tiene un arreglo de productos:
       * 1. Request tipo POST :http://localhost:8080/api/carts
         2. Parametros:
13. Implementamos rutas con router:
    1. /routes/carts.route.js
    2. /route/products.route.js