Texto

El contenido generado por IA puede ser incorrecto.

índice

**Introducción y Objetivo**

A continuación, desarrollare un reporte de un proyecto universitario sobre la modulación de endpoints utilizando node.js, js, express y demás tecnologías que se especificaran durante el reporte.

El objetivo el proyecto es crear endpoints modulados como obtener todos los datos, obtener por id, products, obtener por category obtener por Brand para las siguientes entidades:

* USERS:
  + Id
  + Name
  + Username
  + Password
* CATEGORIES
  + Id
  + categoryName
  + description
  + active
* BRANDS
  + Id
  + brandName
  + description
  + active
* PRODUCTS
  + Id
  + image
  + productName
  + description
  + price
  + stock
  + categoryId
  + brandId

**Requerimientos previos**

Para el desarrollo de este proyecto ocupamos del editor Visual studio code, la tecnología node.js para las instalaciones de las dependencias necesarias, y el conocimiento de JavaScript y el manejo de comandos en la terminal.

**Desarrollo**

**Abrir el workspace**

Antes de comenzar con la actualización de nuestro proyecto, es necesario asegurarnos de estar trabajando en el entorno de desarrollo que configuramos en la práctica anterior. Para ello, abre tu editor de código (VS Code, PyCharm u otro de tu preferencia) y carga el proyecto en el que hemos estado trabajando.

Para iniciar el proyecto tenemos que hacer el uso del comando npm install express y se creara el entorno de trabajo Texto

El contenido generado por IA puede ser incorrecto.

Después instalaremos la tecnología fakker con el siguiente comando npm install fakerTexto

El contenido generado por IA puede ser incorrecto.

Una vez creado el entorno crearemos un archivo de nombre .gitignore y con el uso de la página gitignore.io se hará lo siguienteInterfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Una vez creado el documento copiaras el output y lo pegaras en el proyecto

Una vez esto hecho en el index se harán las siguientes importaciones

Texto

El contenido generado por IA puede ser incorrecto.

* **express**: Importa el framework Express, que sirve para crear y gestionar el servidor web, definir rutas y manejar peticiones HTTP de manera sencilla y eficiente.
* **faker**: Importa la librería Faker, utilizada para generar datos falsos o aleatorios, útil para pruebas y desarrollo sin necesidad de datos reales.
* **routerApi**: Importa un módulo local (./routes/rutas) que probablemente contiene la definición y modularizacion de rutas adicionales para tu aplicación, ayudando a mantener el código organizado y escalable.

**implementación del index**

En el archivo index.js se implementa la configuración principal del servidor Express para tu proyecto. Se importan las dependencias necesarias (express, faker y las rutas modulares), se crea una instancia de la aplicación y se define el puerto de escucha.

Se configuran varias rutas básicas:

* Una ruta raíz (/) que responde con un mensaje de bienvenida.
* Una ruta adicional (/nuevaruta) que responde con otro mensaje.
* Dos rutas dinámicas que reciben parámetros en la URL para categorías y marcas, devolviendo esos parámetros en formato JSON.

Finalmente, se importa y utiliza un módulo de rutas externas (routerApi) para organizar mejor las rutas del proyecto. El servidor se inicia y queda escuchando en el puerto especificado, mostrando mensajes en consola para confirmar que está funcionando.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Implementacion de Rutas.js**

El archivo rutas.js funciona como un módulo centralizador de rutas para tu aplicación Express. Su propósito es importar los routers de diferentes recursos (productos, usuarios, categorías y marcas) y asociarlos a rutas base específicas en la aplicación principal.

¿Qué hace exactamente?

* Importa los routers de productos, usuarios, categorías y marcas desde sus respectivos archivos.
* Define la función routerApi(app), que recibe la instancia de la aplicación Express.
* Dentro de esta función, utiliza [app.use()](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") para montar cada router en una ruta base:
  + /products para productos
  + users para usuarios
  + /categories para categorías
  + /brands para marcas
* Exporta la función para que pueda ser utilizada en el archivo principal (index.js), permitiendo así una organización modular y escalable de las rutas

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**Implementación del módulo productsRouter.js**

Este archivo define un router de Express para gestionar endpoints relacionados con productos. Utiliza la librería [faker](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) para generar datos falsos de productos, lo que es útil para pruebas y desarrollo sin necesidad de una base de datos real.

Rutas implementadas:

1. GET /  
   Devuelve un arreglo de productos falsos. El número de productos se puede controlar con el parámetro de consulta [size](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "); si no se especifica, devuelve 10 productos por defecto. Cada producto incluye campos como id, imagen, nombre, descripción, precio, stock, id de categoría y de marca, todos generados aleatoriamente.
2. GET /filter  
   Devuelve un mensaje de texto simple: "Soy una ruta de filtro". Es una ruta de ejemplo o placeholder.
3. Texto

   El contenido generado por IA puede ser incorrecto.GET /:id  
   Devuelve un producto falso con el id especificado en la URL. Los demás campos (imagen, nombre, descripción, precio, stock, id de categoría y de marca) también se generan aleatoriamente.

**Implementación del modulo UsersRouter.js**

Este archivo define un router de Express para manejar rutas relacionadas con usuarios. Utiliza la librería [faker](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) para generar datos de usuario falsos y expone tres endpoints principales.

**Endpoints definidos**

1. **GET /**
   * **Funcionalidad:** Devuelve una lista de usuarios generados aleatoriamente.
   * **Parámetro de consulta:** [size](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") (opcional) para definir cuántos usuarios retornar (por defecto 10).
   * **Respuesta:** Array de objetos usuario con campos: [id](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [name](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "), [username](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "), [password](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").
2. **GET /filter**
   * **Funcionalidad:** Ruta de ejemplo que responde con el texto "Soy una ruta de filtro".
   * **Respuesta:** Texto plano.
3. **GET /:id**
   * **Funcionalidad:** Devuelve un usuario fijo con el id recibido por parámetro.
   * **Respuesta:** Objeto usuario con campos: id, name, username, password (valores fijos).Texto

     El contenido generado por IA puede ser incorrecto.

**Implementación del módulo CategoriesRouter.js**

Este archivo permite simular una API REST de categorías, devolviendo datos ficticios útiles para desarrollo frontend o pruebas, sin necesidad de una base de datos real.

El archivo CategoriesRouter.js define y exporta un router de Express para manejar las rutas relacionadas con categorías. Utiliza la librería faker para generar datos falsos de categorías, lo que es útil para pruebas y desarrollo.

**¿Qué rutas implementa?**

1. **GET /**  
   Devuelve un arreglo de categorías falsas. Puedes especificar cuántas categorías quieres con el parámetro de consulta [size](vscode-file://vscode-app/c:/Users/cesar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "); si no lo envías, devuelve 10 por defecto. Cada categoría tiene un id, nombre, descripción y un estado activo (booleano), todos generados aleatoriamente.
2. **GET /filter**  
   Devuelve un mensaje de texto simple: "Soy una ruta de filtro". Es solo un ejemplo o placeholder.
3. **GET /:id**  
   Devuelve una categoría falsa con el id especificado en la URL. Los demás datos (nombre, descripción y estado) se generan aleatoriamente.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Implementación de BrandsRouter.jsEste

Este archivo implementa un router de Express para gestionar las rutas relacionadas con marcas (brands) en tu API. Utiliza la librería faker para generar datos falsos, lo que permite simular una base de datos de marcas durante el desarrollo o pruebas.

**Rutas implementadas:**

1. **GET /**
   * Devuelve un arreglo de marcas falsas.
   * El número de marcas se puede controlar con el parámetro de consulta size (por defecto, 10).
   * Cada marca incluye:
     + id: identificador único generado aleatoriamente.
     + brandName: nombre de la marca (usando un nombre de producto falso).
     + description: descripción generada aleatoriamente.
     + active: estado booleano aleatorio.
2. **GET /filter**
   * Devuelve el texto "Soy una ruta de filtro".
   * Es una ruta de ejemplo o placeholder.
3. **GET /:id**
   * Devuelve una marca falsa con el id especificado en la URL.
   * Los demás campos (brandName, description, active) se generan aleatoriamente.

**Ejecución del proyecto**

Para la ejecución del proyecto abriremos una terminal en visual studio con el comando ctrl + shift + ñ o también se puede ejecutar desde el símbolo del sistema(CMD) lo único que tienes que hacer es ubicar la ubicación del proyecto en tu dispositivo y copiarla después escribirás esto en el símbolo del sistema : cd (ubicación del proyecto) una vez ubicado en la terminal usaremos el siguiente comando: **npm run dev**

Texto

El contenido generado por IA puede ser incorrecto.

Y una vez el proyecto corra sin errores estará listo para su uso

**Resultados**

Resultados 1 - Vista principal de la documentación automática

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Resultados 2 - Vista Endpoint products, búsqueda por id

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Resultados 3 - Vista Endpoint users, búsqueda por id

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Resultados 4 - Vista Endpoint brand, búsqueda por id

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Resultados 5 - Vista Endpoint categories, búsqueda por id

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Resultados 6 - Vista Endpoint búsqueda category por id

Pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Resultados 7 - Vista Endpoint búsquedabrand por id

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Conclusiones**

En este documento se abordó el aprendizaje de las tecnologías necesarias para crear un servidor con Express. El código fue desarrollado manualmente desde cero, incluyendo la instalación de dependencias mediante *npm*. Asimismo, se trabajó con la definición de rutas, el uso del lenguaje de programación JavaScript y la integración de diversas herramientas como *.gitignore*, *faker* y otras librerías. Finalmente, se elaboró un reporte que permitió llevar un registro detallado del proyecto en un documento.