Texto

El contenido generado por IA puede ser incorrecto.

índice

Contenido

[Introducción y Objetivo 3](#_Toc207631530)

[Requerimientos previos 4](#_Toc207631531)

[Desarrollo 4](#_Toc207631532)

[Abrir el workspace 4](#_Toc207631533)

[implementación del index 6](#_Toc207631534)

[Implementacion de Rutas.js 7](#_Toc207631535)

[Implementación del módulo productsRouter.js 8](#_Toc207631536)

[Implementación del modulo UsersRouter.js 9](#_Toc207631537)

[Implementación del módulo CategoriesRouter.js 10](#_Toc207631538)

# 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.

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) 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 encargado de manejar los endpoints relacionados con productos. Se utiliza la librería **faker** para generar una lista de productos ficticios al inicio de la ejecución, lo que permite simular datos sin necesidad de una base de datos real.Rutas implementadas:

**Rutas implementadas**

* **GET /**  
  Retorna todos los productos generados.
* **GET /filter**  
  Devuelve el texto "Soy una ruta de filtro". Funciona únicamente como placeholder o ruta de ejemplo.
* **GET /:id**  
  Busca un producto específico por su id y lo devuelve en formato JSON.
* **GET /category/:categoryId**  
  Filtra los productos por categoryId y devuelve un arreglo con los resultados que coincidan.
* **GET /brand/:brandId**  
  Filtra los productos por brandId.

Texto

El contenido generado por IA puede ser incorrecto.

# Implementación del modulo UsersRouter.js

Este archivo define un router de Express para manejar los endpoints relacionados con usuarios. A diferencia de versiones anteriores, los datos ya no se generan dinámicamente por cada petición, sino que se crea un conjunto inicial de **10 usuarios** al momento de cargar el módulo.

**Endpoints definidos**

### Rutas implementadas

* GET / **Devuelve la lista completa de usuarios en formato JSON.**
* GET /filter **Responde con el texto "Soy una ruta de filtro". Es un ejemplo o placeholder.**
* GET /:id **Busca un usuario por su id y lo devuelve en formato JSON**

**Dos rutas dinámicas que reciben parámetros en la URL para categorías y marcas, devolviendo esos parámetros en formato JSON.**

Texto

El contenido generado por IA puede ser incorrecto.

# Implementación del módulo CategoriesRouter.js

Este archivo define un router de Express que maneja los endpoints relacionados con categorías. Se utiliza faker para generar **10 categorías ficticias**.

**¿Qué rutas implementa?**

**Rutas implementadas**

* **GET /  
  Devuelve todas las categorías generadas en el arreglo inicial.**
* **GET /filter  
  Responde con el texto "Soy una ruta de filtro".**
* **GET /:id  
  Busca una categoría por su identificador numérico y devuelve sus datos.**

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Implementación de BrandsRouter.jsEste

Este archivo define un router de Express para gestionar los endpoints relacionados con las marcas. Al igual que los demás módulos, utiliza faker para generar datos ficticios.

**Rutas implementadas:**

**Rutas implementadas**

* **GET /  
  Devuelve todas las marcas generadas.**
* **GET /filter  
  Responde con el texto "Soy una ruta de filtro".**
* **GET /:id  
  Busca y devuelve una marca específica por su id.**

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.