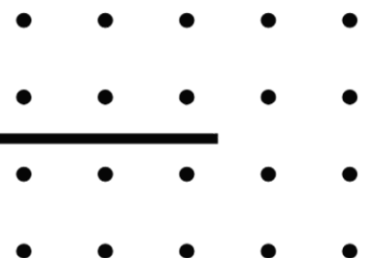

• • • • •

César Cázares

P1_ENDPOINTS MODULADOS



índice

Contenido

Introducción y Objetivo.....	3
Requerimientos previos.....	4
Desarrollo.....	4
Abrir el workspace.....	4
implementación del index.....	6
Implementacion de Rutas.js	7
Implementación del módulo productsRouter.js.....	8
Implementación del modulo UsersRouter.js.....	9
Implementación del módulo CategoriesRouter.js	10

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

```
PS C:\Users\cesar\Downloads\MYSTORE_302 - Copy\MYSTORE_302 - Copy> npm i express

up to date, audited 181 packages in 2s

46 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Después instalaremos la tecnología faker con el siguiente comando npm install faker

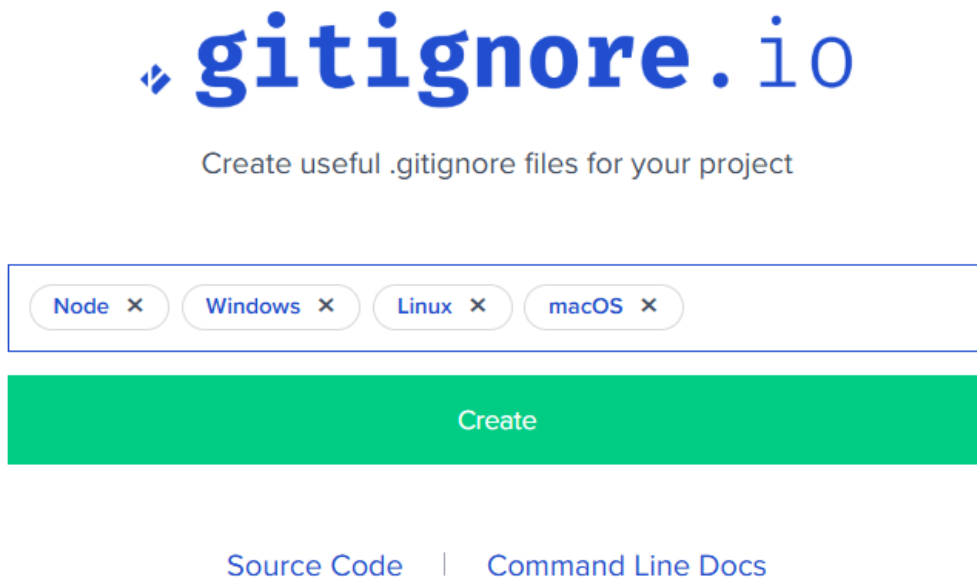
```
PS C:\Users\cesar\Downloads\MYSTORE_302 - Copy\MYSTORE_302 - Copy> npm i faker

up to date, audited 181 packages in 1s

46 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Una vez creado el entorno crearemos un archivo de nombre `.gitignore` y con el uso de la página `gitignore.io` se hará lo siguiente



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



- **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 modularización 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.

```
index.js
app.get("/", (req, res) => {
  res.send("Hola mi server en express")
})

app.get("/nuevaruta", (req, res) => {
  res.send("Hola soy una nueva ruta")
})

app.listen(port, () => {
  console.log("Mi port is working on: " + port)
  console.log("http://localhost:" + port)
})

app.get('/category/:categoryId/products/:productId',
(req, res) => {
  const { categoryId, productId } = req.params;
  res.json({
    categoryId,
    productId
  });
});

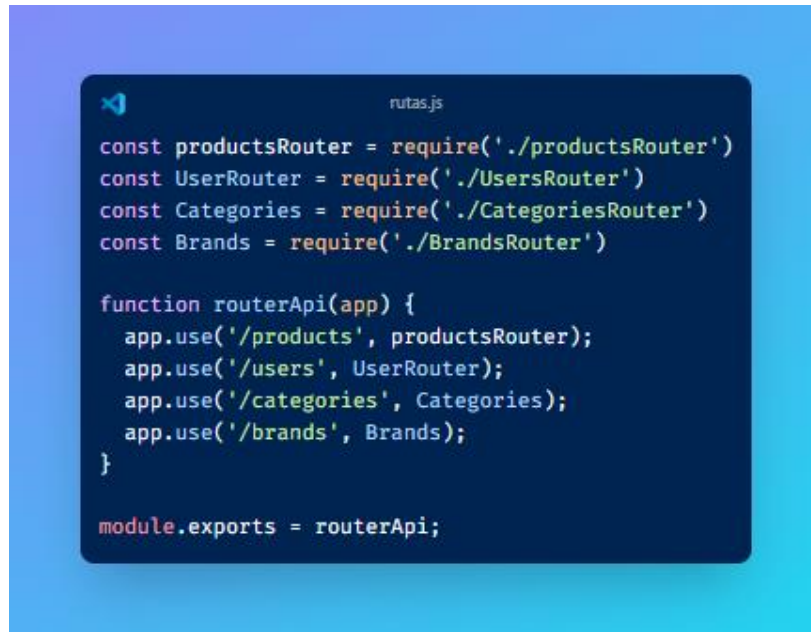
app.get('/brand/:brandId/products/:productId', (req,
res) => {
  const { brandId, productId } = req.params;
  res.json({
    brandId,
    productId
  });
});
```

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()` 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



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.


```

productsRouter.js

router.get("/", (req, res) => {
  const products = [];
  const { size } = req.query;
  const limit = size || 10;
  for (let index = 0; index < limit; index++) {
    products.push({
      id: faker.datatype.uuid(),
      image: faker.image.imageUrl(),
      productName: faker.commerce.productName(),
      description: faker.commerce.productDescription(),
      price: faker.commerce.price(),
      stock: faker.datatype.number({ min: 0, max: 100 }),
      categoryId: faker.datatype.number({ min: 1, max: 10 }),
      brandId: faker.datatype.number({ min: 1, max: 10 })
    });
  }
  res.json(products);
});

router.get('/filter', (req, res) => {
  res.send('Soy una ruta de filtro')
})

router.get("/:id", (req, res) => {
  const { id } = req.params; // Extraemos el parametro id de los parametros ruta
  res.json({
    id: id,
    image: faker.image.imageUrl(),
    productName: faker.commerce.productName(),
    description: faker.commerce.productDescription(),
    price: faker.commerce.price(),
    stock: faker.datatype.number({ min: 0, max: 100 }),
    categoryId: faker.datatype.number({ min: 1, max: 5 }),
    brandId: faker.datatype.number({ min: 1, max: 10 })
  });
});

module.exports = router;

```

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.

```
UsersRouter.js

const faker = require('faker');
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  const users = [];
  const { size } = req.query;
  const limit = size || 10;
  for (let index = 0; index < limit; index++) {
    users.push({
      id: index + 1,
      Name: faker.name.findName(),
      username: faker.name.findName(),
      password: faker.internet.password(),
    });
  }
  res.json(users);
});

router.get('/filter', (req, res) => {
  res.send('Soy una ruta de filtro');
});

router.get('/:id', (req, res) => {
  const { id } = req.params;
  res.json({
    id: id,
    Name: faker.name.findName(),
    username: faker.name.findName(),
    password: faker.internet.password()
  });
});

module.exports = router;
```

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.



```
CategoriesRouter.js

const faker = require('faker');
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  const categories = [];
  const { size } = req.query;
  const limit = size || 10;
  for (let index = 0; index < limit; index++) {
    categories.push({
      Id: index + 1,
      categoryName: faker.commerce.department(),
      description: faker.commerce.productDescription(),
      active: faker.datatype.boolean()
    });
  }
  res.json(categories);
});

router.get('/filter', (req, res) => {
  res.send('Soy una ruta de filtro')
});

router.get('/:id', (req, res) => {
  const { id } = req.params;
  res.json({
    id: id,
    categoryName: faker.commerce.department(),
    description: faker.commerce.productDescription(),
    active: faker.datatype.boolean()
  });
});

module.exports = router;
```

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.

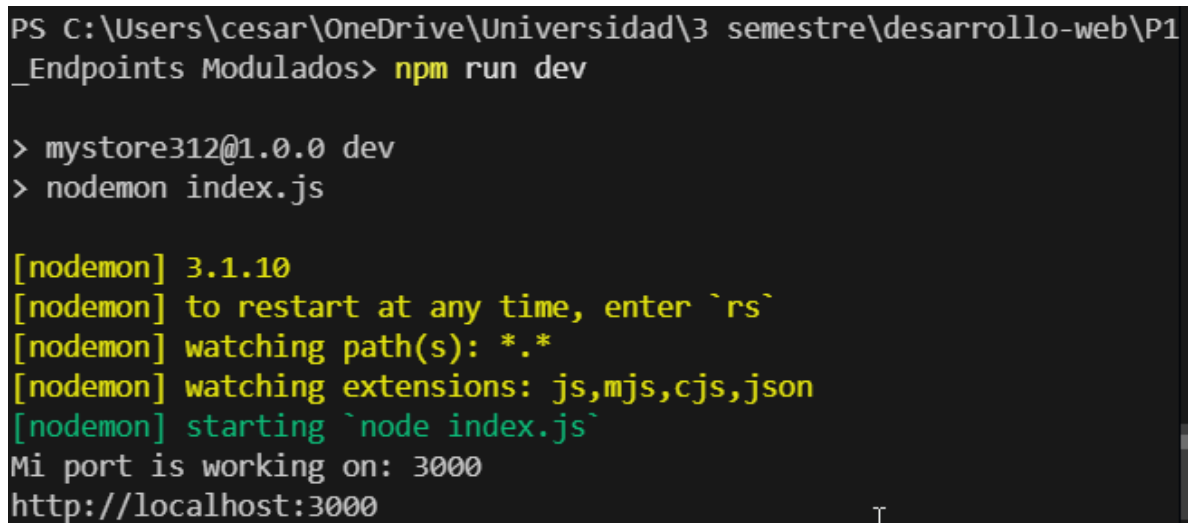
Rutas implementadas:

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`**



```
PS C:\Users\cesar\OneDrive\Universidad\3 semestre\desarrollo-web\P1_Endpoints Modulados> npm run dev

> mystore312@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Mi port is working on: 3000
http://localhost:3000
```

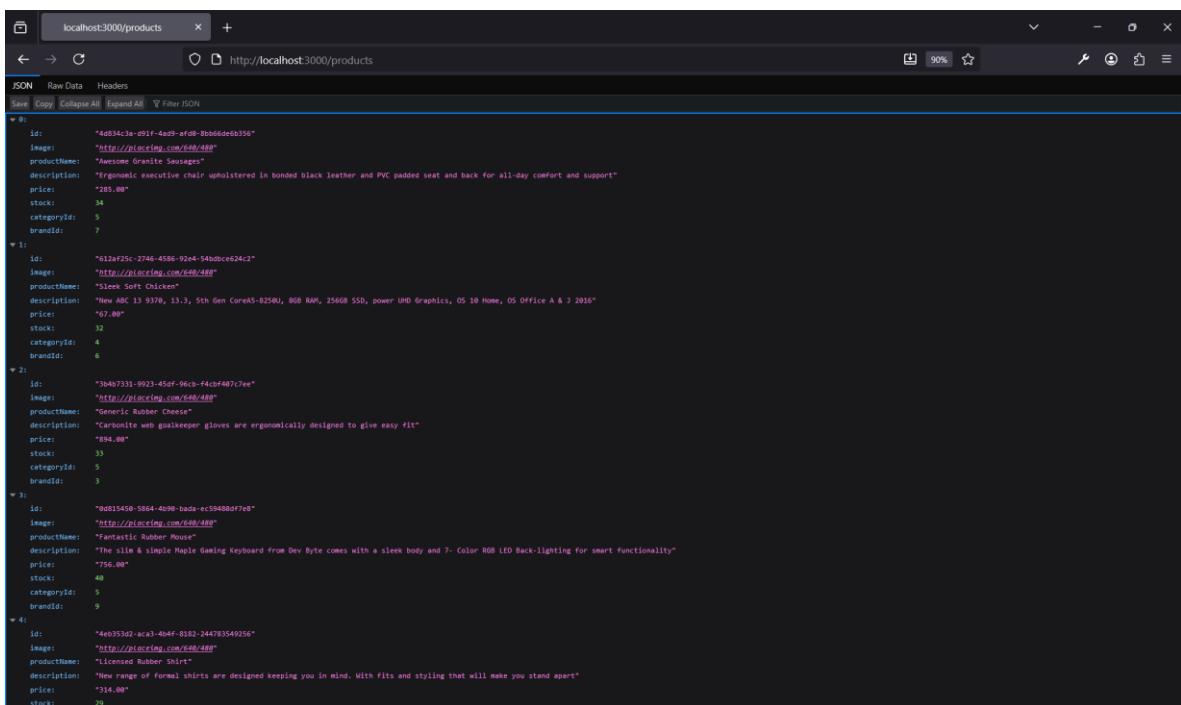
Y una vez el proyecto corra sin errores estará listo para su uso

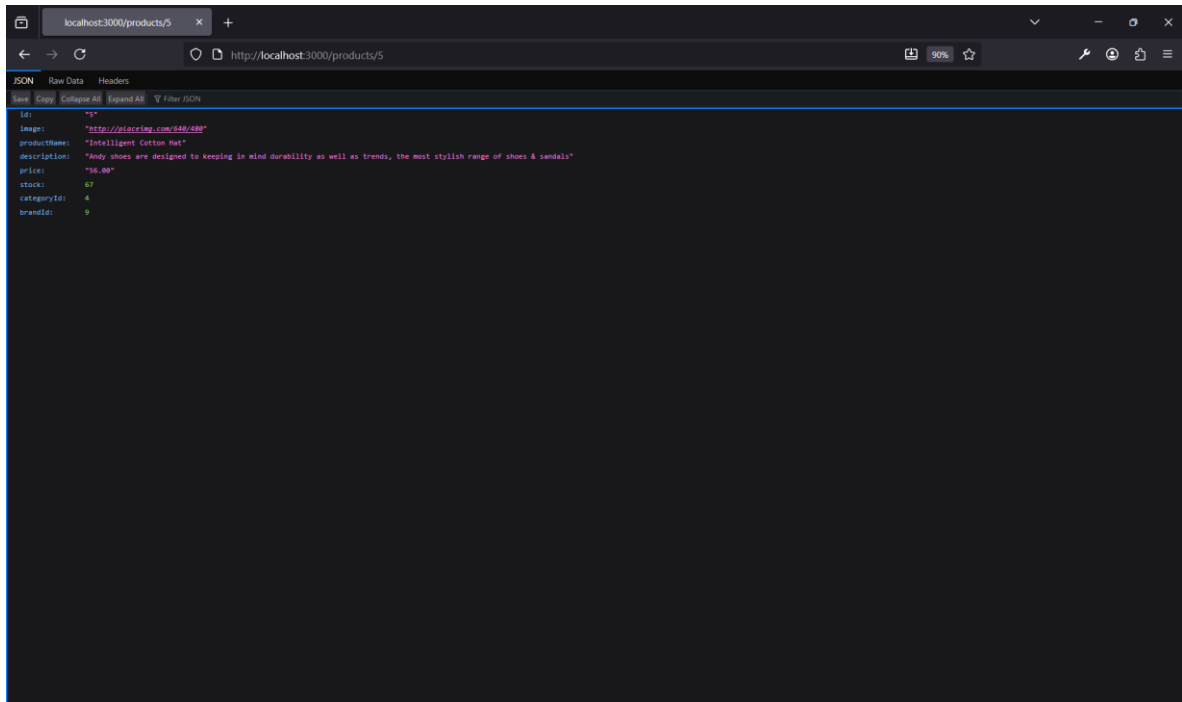
Resultados

Resultados 1 - Vista principal de la documentación automática

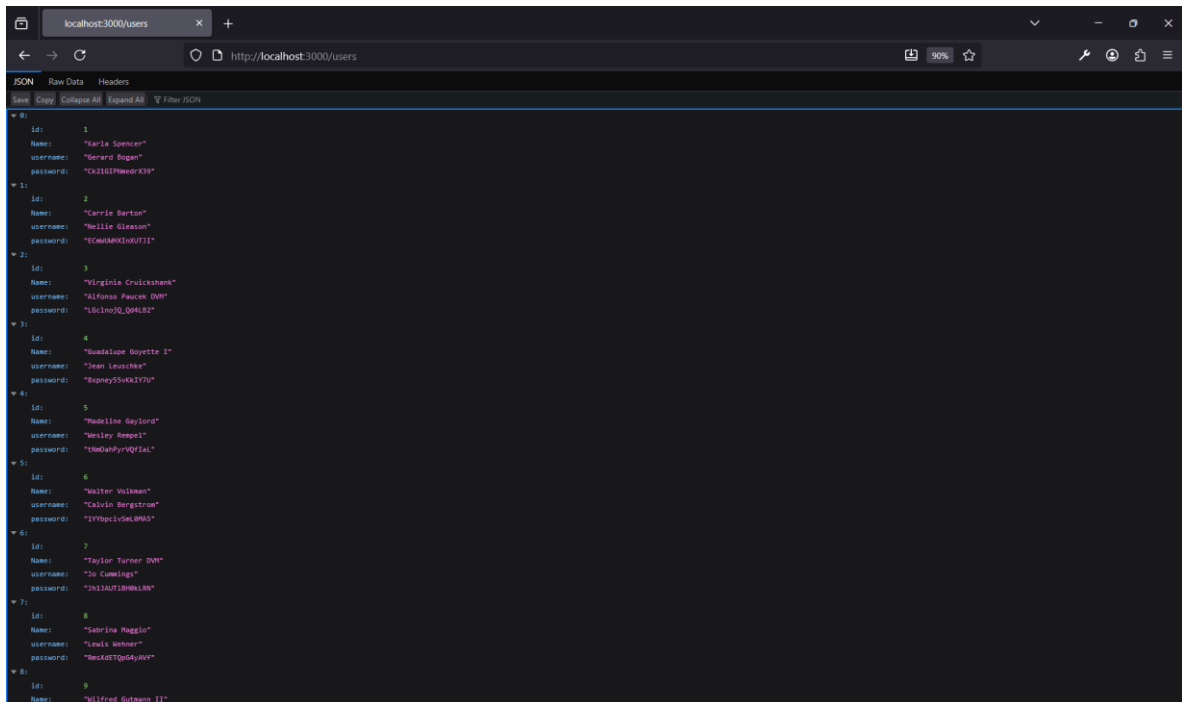


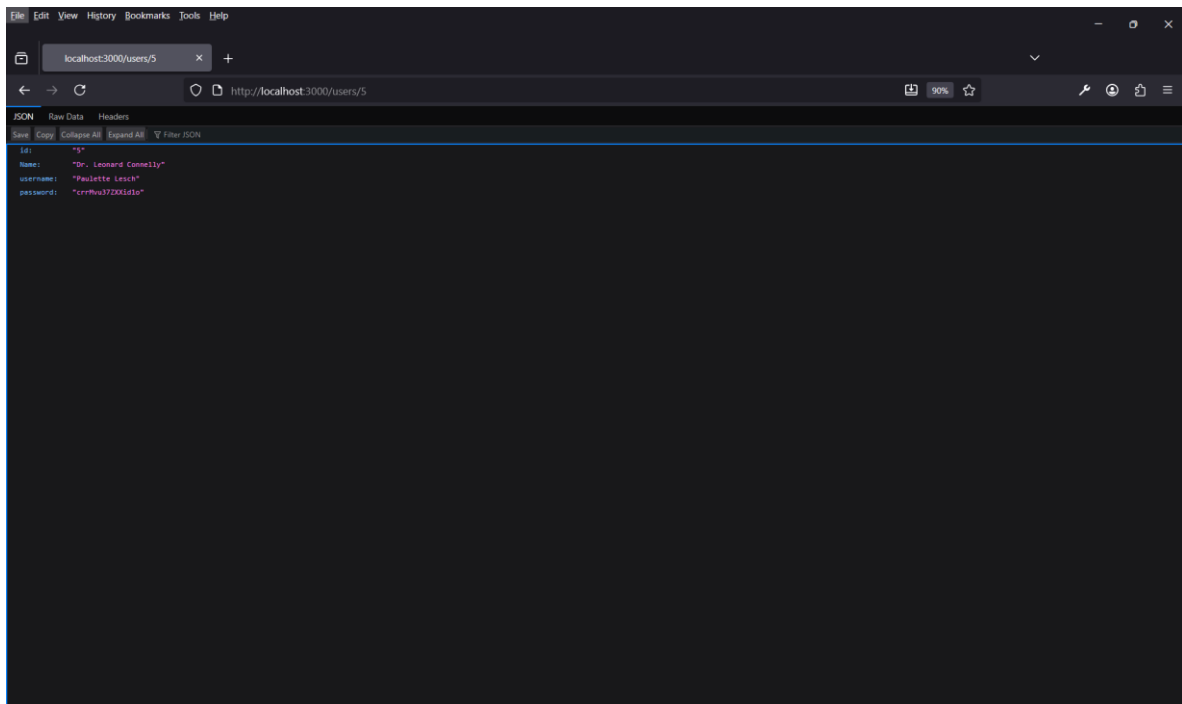
Resultados 2 - Vista Endpoint products, búsqueda por id





Resultados 3 - Vista Endpoint users, búsqueda por id



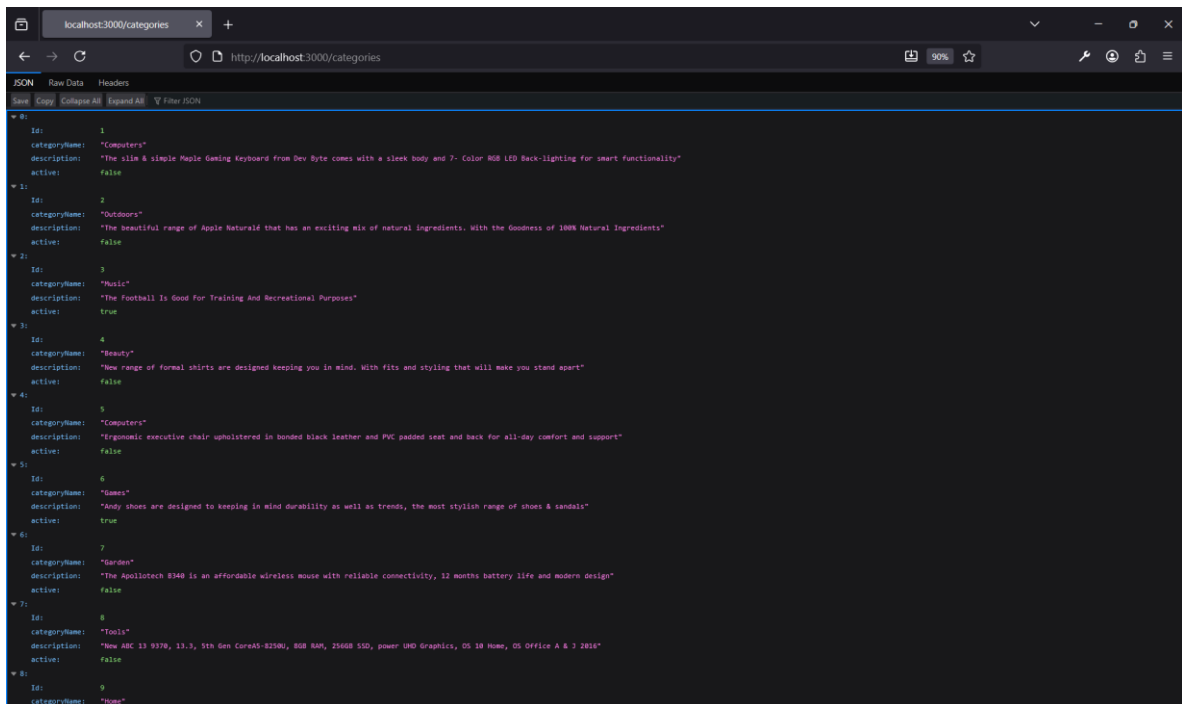


Resultados 4 - Vista Endpoint brand, búsqueda por id

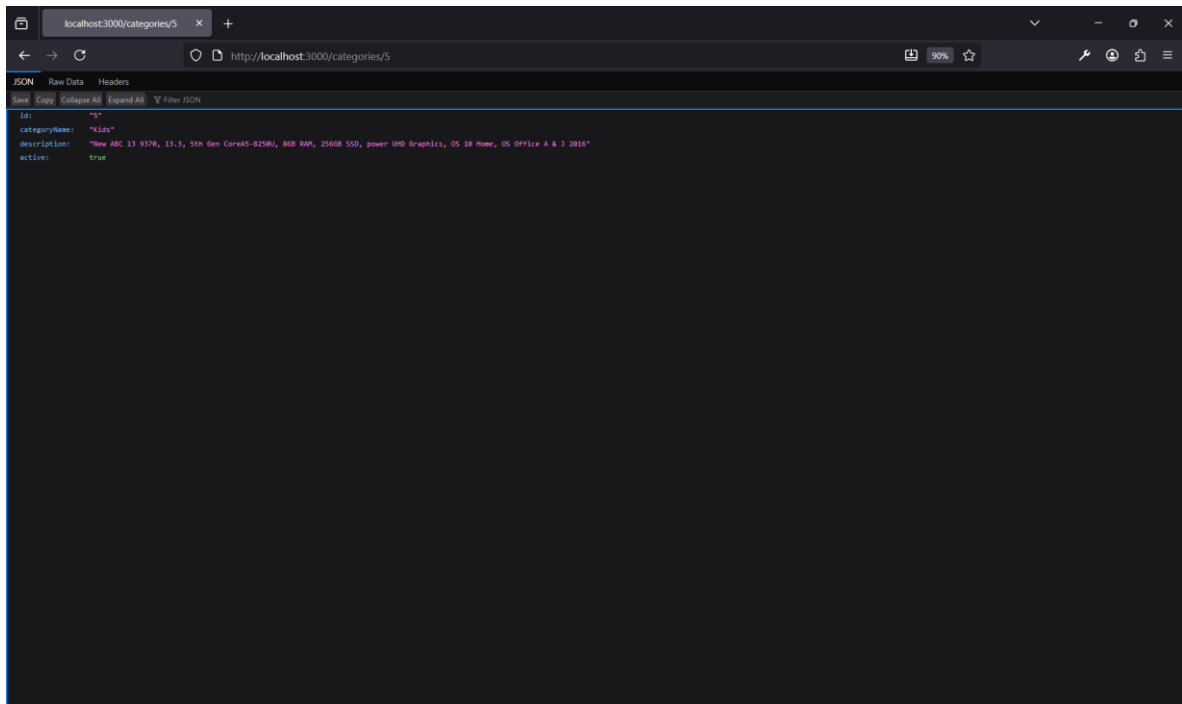
```
localhost:3000/brands/
http://localhost:3000/brands/
JSON Raw Data Headers
id: "693d838c-cde2-42ad-a88e-ef7961d6c863"
brandName: "Gorgeous Cotton Soap"
description: "The slim & simple Hupie Gaming Keyboard from Dev Byte comes with a sleek body and 7- Color RGB LED Back-lighting for smart functionality"
active: true
id: "ca57f2d3-b58c-4a8d-a5ad-cb255e651204"
brandName: "Handmade Steel Chips"
description: "The Hagasaki Lander is the trademarked name of several series of Hagasaki sport bikes, that started with the 1984 ABC8003"
active: false
id: "9d92115e-4388-4377-acaa-94e08e6a9298"
brandName: "Refined Plastic Chair"
description: "The slim & simple Hupie Gaming Keyboard from Dev Byte comes with a sleek body and 7- Color RGB LED Back-lighting for smart functionality"
active: true
id: "884e5541-365d-435f-a742-c841d6c6185e"
brandName: "Awesome Metal Pants"
description: "The Hagasaki Lander is the trademarked name of several series of Hagasaki sport bikes, that started with the 1984 ABC8003"
active: true
id: "465c68d0-1a1c-4c8b-a878-a239748d2274"
brandName: "Gorgeous Soft Chips"
description: "Ergonomic executive chair upholstered in bonded black leather and PVC padded seat and back for all-day comfort and support"
active: true
id: "161c7846-ff43-4ae4-b8aa-161fa2371e78"
brandName: "Ergonomic Frozen Poutine"
description: "The beautiful range of Apple Naturald that has an exciting mix of natural ingredients. With the Goodness of 100% Natural Ingredients"
active: false
id: "94d0db04-b7de-4cc1-98d5-3413f12e350d"
brandName: "Refined Fresh Bike"
description: "The beautiful range of Apple Naturald that has an exciting mix of natural ingredients. With the Goodness of 100% Natural Ingredients"
active: false
id: "7ef13f24-61c8-4c3a-a888-8135d8d595b4"
brandName: "Awesome Cotton Shoes"
description: "The Apulitech B340 is an affordable wireless mouse with reliable connectivity, 12 months battery life and modern design"
active: true
id: "9463d816-b49f-4441-b598-2cdca19feb04"
brandName: "Refined Wooden Ball"
```

```
localhost:3000/brands/5
http://localhost:3000/brands/5
JSON Raw Data Headers
id: "5"
brandName: "Sleek Granite Cheese"
description: "The automobile layout consists of a front-engine design, with transaxle-type transmissions mounted at the rear of the engine and four wheel drive"
active: false
```

Resultados 5 - Vista Endpoint categories, búsqueda por id

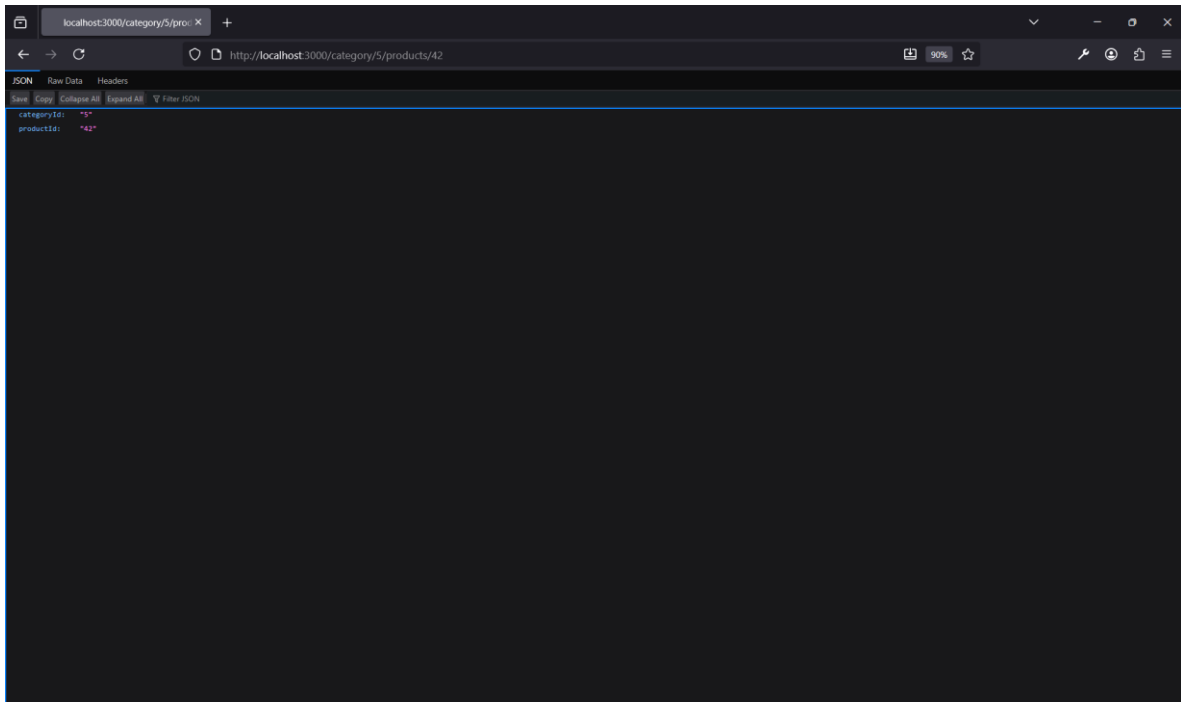


```
localhost:3000/categories x +
http://localhost:3000/categories
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
[
  {
    "Id": 1,
    "categoryName": "Computers",
    "description": "The slim & simple Maple Gaming Keyboard from Dev Byte comes with a sleek body and 7- Color RGB LED Back-lighting for smart functionality",
    "active": false
  },
  {
    "Id": 2,
    "categoryName": "Outdoors",
    "description": "The beautiful range of Apple Naturald that has an exciting mix of natural ingredients. With the Goodness of 100% Natural Ingredients",
    "active": false
  },
  {
    "Id": 3,
    "categoryName": "Music",
    "description": "The football Is Good for Training And Recreational Purposes",
    "active": true
  },
  {
    "Id": 4,
    "categoryName": "Beauty",
    "description": "New range of formal shirts are designed keeping you in mind. With fits and styling that will make you stand apart",
    "active": false
  },
  {
    "Id": 5,
    "categoryName": "Computers",
    "description": "Ergonomic executive chair upholstered in bonded black leather and PVC padded seat and back for all-day comfort and support",
    "active": false
  },
  {
    "Id": 6,
    "categoryName": "Games",
    "description": "Sleady shoes are designed to keeping in mind durability as well as trends, the most stylish range of shoes & sandals",
    "active": true
  },
  {
    "Id": 7,
    "categoryName": "Garden",
    "description": "The Apollotech B340 is an affordable wireless mouse with reliable connectivity, 12 months battery life and modern design",
    "active": false
  },
  {
    "Id": 8,
    "categoryName": "Tools",
    "description": "New ABC 13 9370, 13.3, 5th Gen Corei5-8250U, 8GB RAM, 256GB SSD, power UHD Graphics, OS 10 Home, OS Office A & J 2016",
    "active": false
  },
  {
    "Id": 9,
    "categoryName": "Home",
    "description": null,
    "active": null
  }
]
```

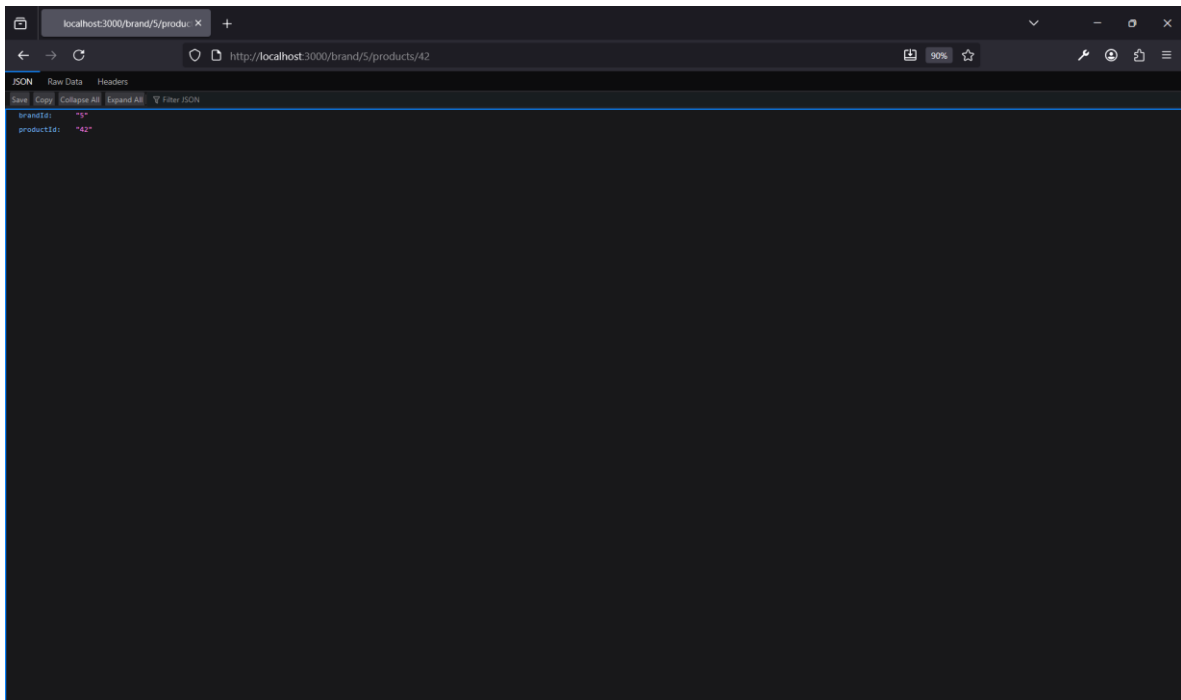


```
localhost:3000/categories/5 x +
http://localhost:3000/categories/5
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
{
  "Id": 5,
  "categoryName": "Computers",
  "description": "Ergonomic executive chair upholstered in bonded black leather and PVC padded seat and back for all-day comfort and support",
  "active": false
}
```

Resultados 6 - Vista Endpoint búsqueda category por id



Resultados 7 - Vista Endpoint búsquedabrand por id



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.