César Cázares

CRUD USERS | CATEGORY | BRANDS | PRODUCTS

Universidad La Salle® Bajío | Facultad de Ingenierías y Tecnologías

índice

Tabla de contenido

Introducción y Objetivo

Este documento presenta el desarrollo de un proyecto universitario sobre la creación de una API REST modular utilizando Node.js, Express.js y otras tecnologías. El proyecto implementa un sistema de gestión para una tienda en línea con entidades modulares.

Objetivo Principal: Crear endpoints modulados para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las siguientes entidades:

- **USERS**: Gestión de usuarios
- **CATEGORIES**: Gestión de categorías de productos
- **BRANDS**: Gestión de marcas
- **PRODUCTS**: Gestión de productos
- **MOVIES**: Gestión de películas (módulo adicional)

Requerimientos Previos

Para el desarrollo de este proyecto se requirió:

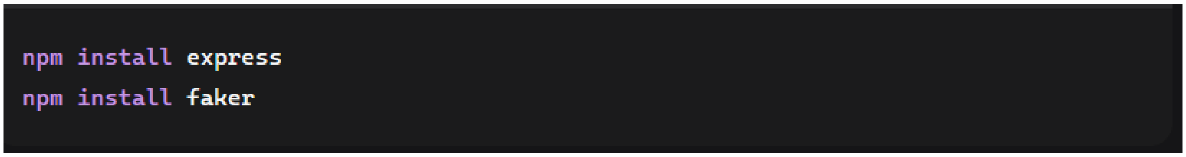
- **Editor de código**: Visual Studio Code
- **Tecnologías**: Node.js, Express.js, Faker.js

- **Conocimientos**: JavaScript, manejo de terminal, conceptos de APIs REST
- **Dependencias**: express, faker, nodemon, eslint, prettier

Desarrollo

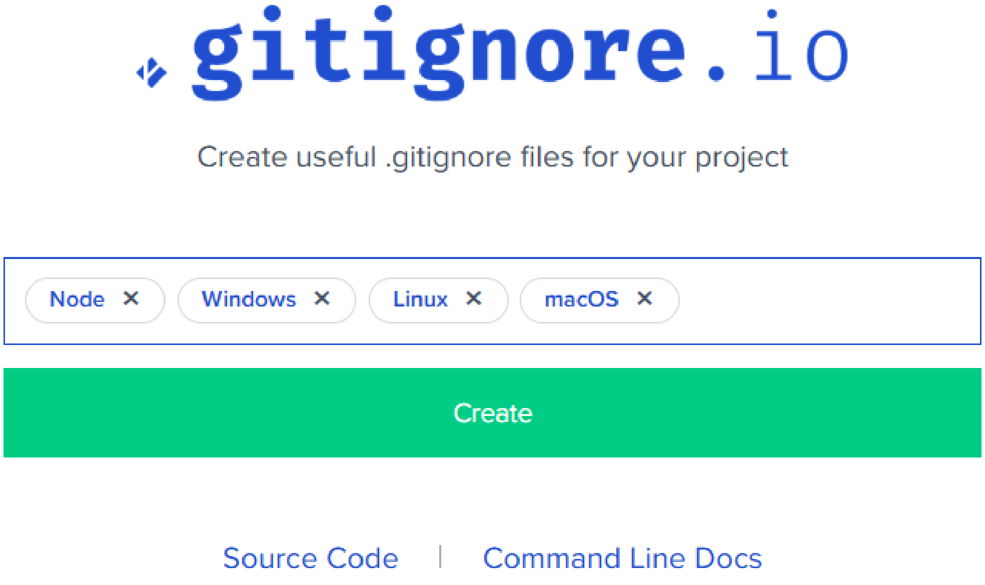
Configuración del Entorno

Se inició el proyecto con los siguientes comandos:



```
npm install express
npm install faker
```

Se configuró un archivo .gitignore utilizando gitignore.io para excluir archivos innecesarios del control de versiones.

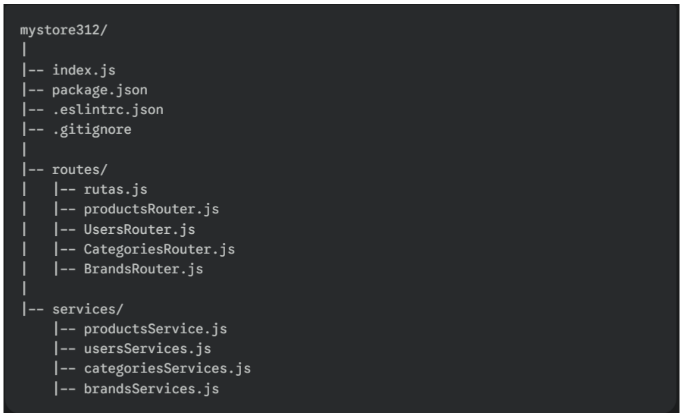


gitignore.io

Create useful .gitignore files for your project

Node ✕ Windows ✕ Linux ✕ macOS ✕

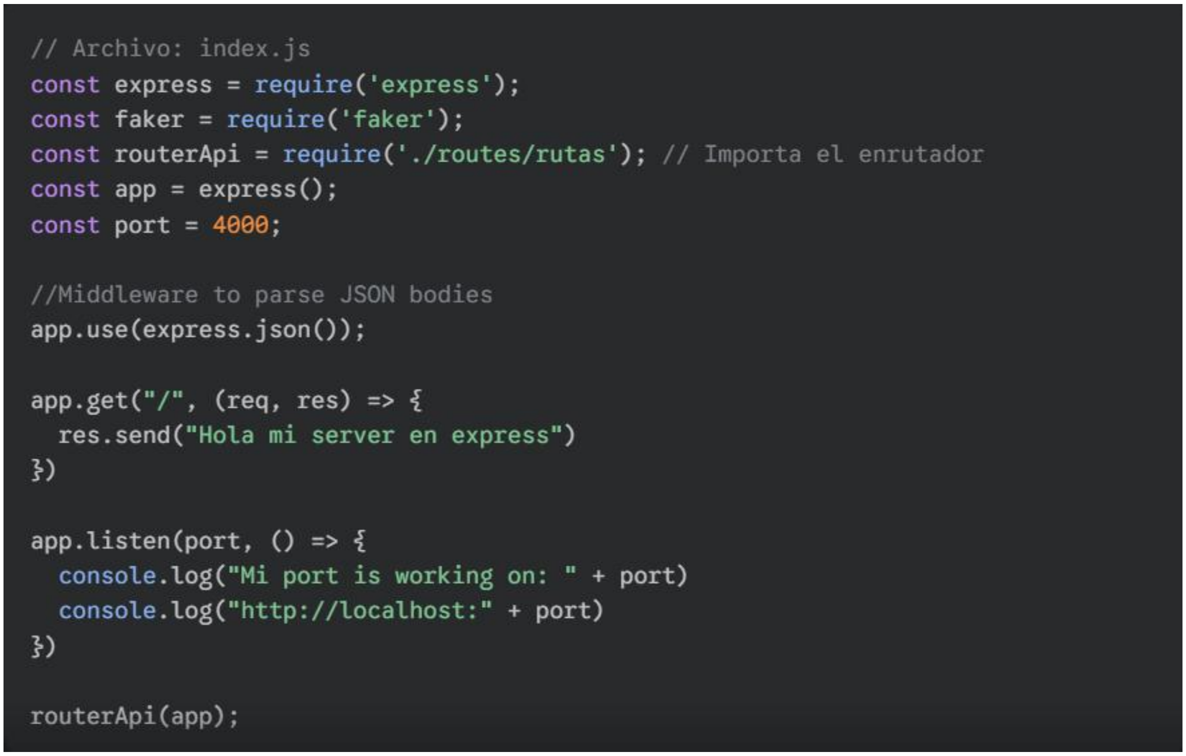
Estructura del Proyecto



```
mystore312/
|
|-- index.js
|-- package.json
|-- .eslintrc.json
|-- .gitignore
|
|-- routes/
|   |-- rutas.js
|   |-- productsRouter.js
|   |-- UsersRouter.js
|   |-- CategoriesRouter.js
|   |-- BrandsRouter.js
|
|-- services/
    |-- productsService.js
    |-- usersServices.js
    |-- categoriesServices.js
    |-- brandsServices.js
```

Implementación del Servidor Principal (index.js)

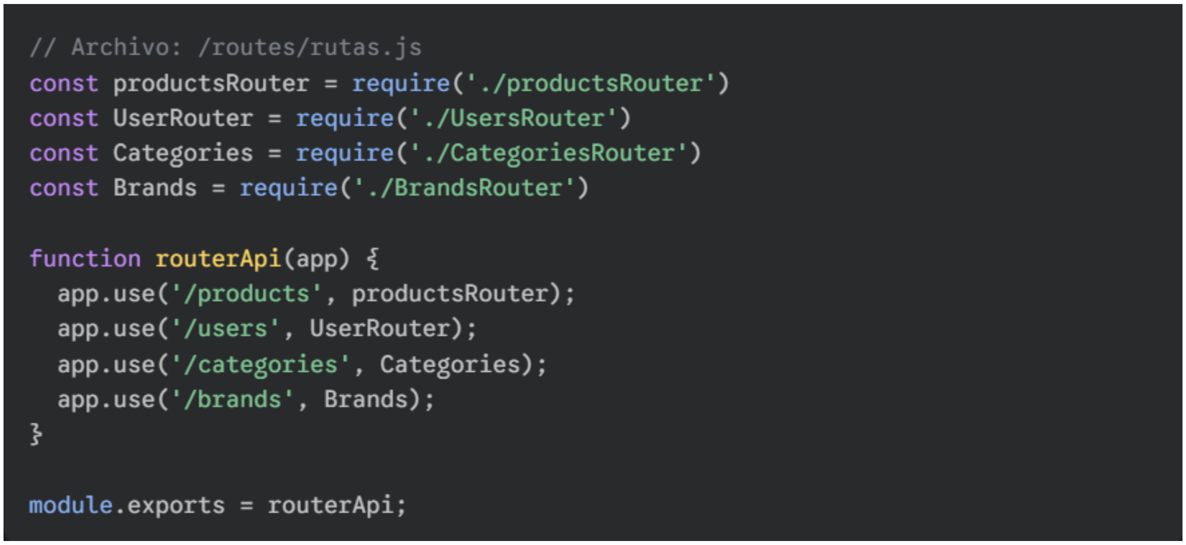
El archivo principal index.js es el punto de entrada que configura y levanta el servidor Express. Define el puerto (4000), aplica middlewares (como express.json()) e invoca al routerApi para gestionar las rutas.



```
// Archivo: index.js
const express = require('express');
const faker = require('faker');
const routerApi = require('./routes/rutas'); // Importa el enrutador
const app = express();
const port = 4000;

//Middleware to parse JSON bodies
app.use(express.json());

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

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

routerApi(app);
```

Sistema de Enrutamiento Modular (rutas.js)

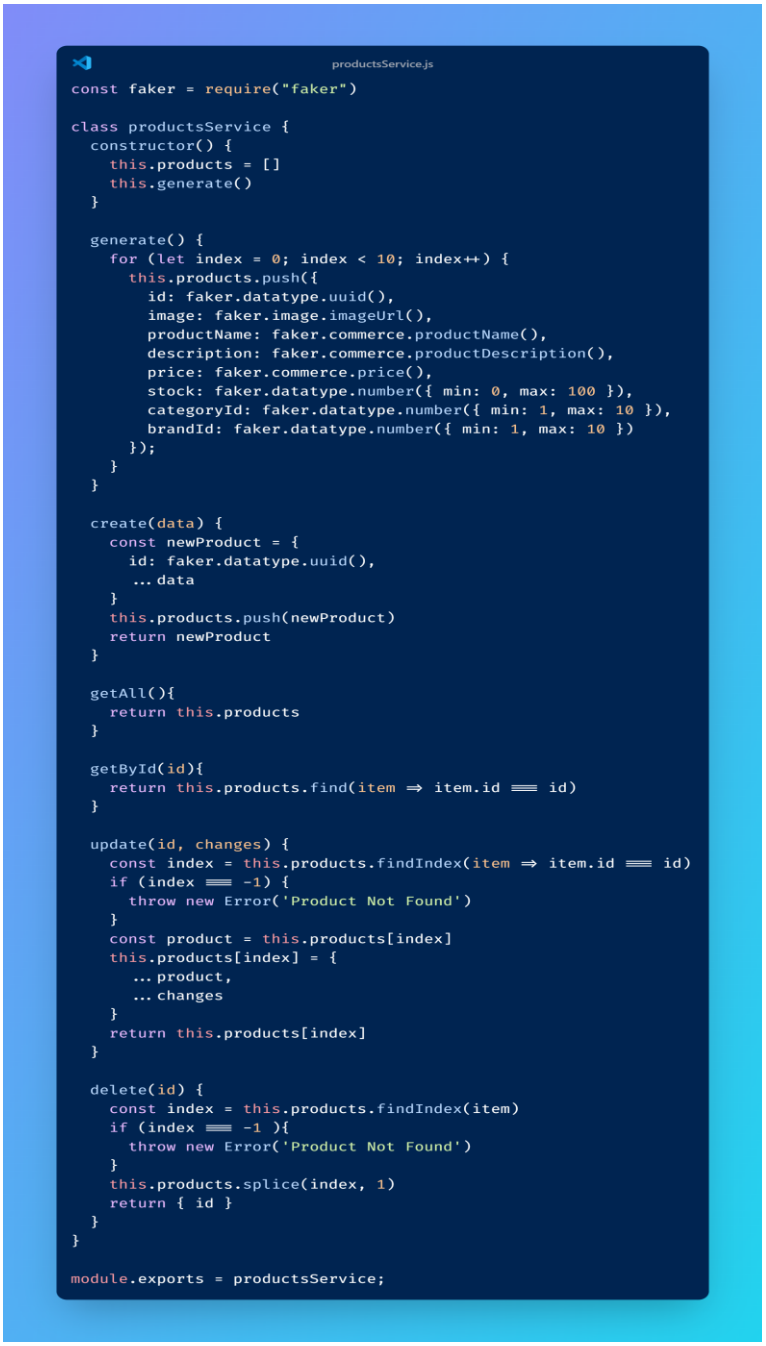
Archivo centralizador que organiza todas las rutas:



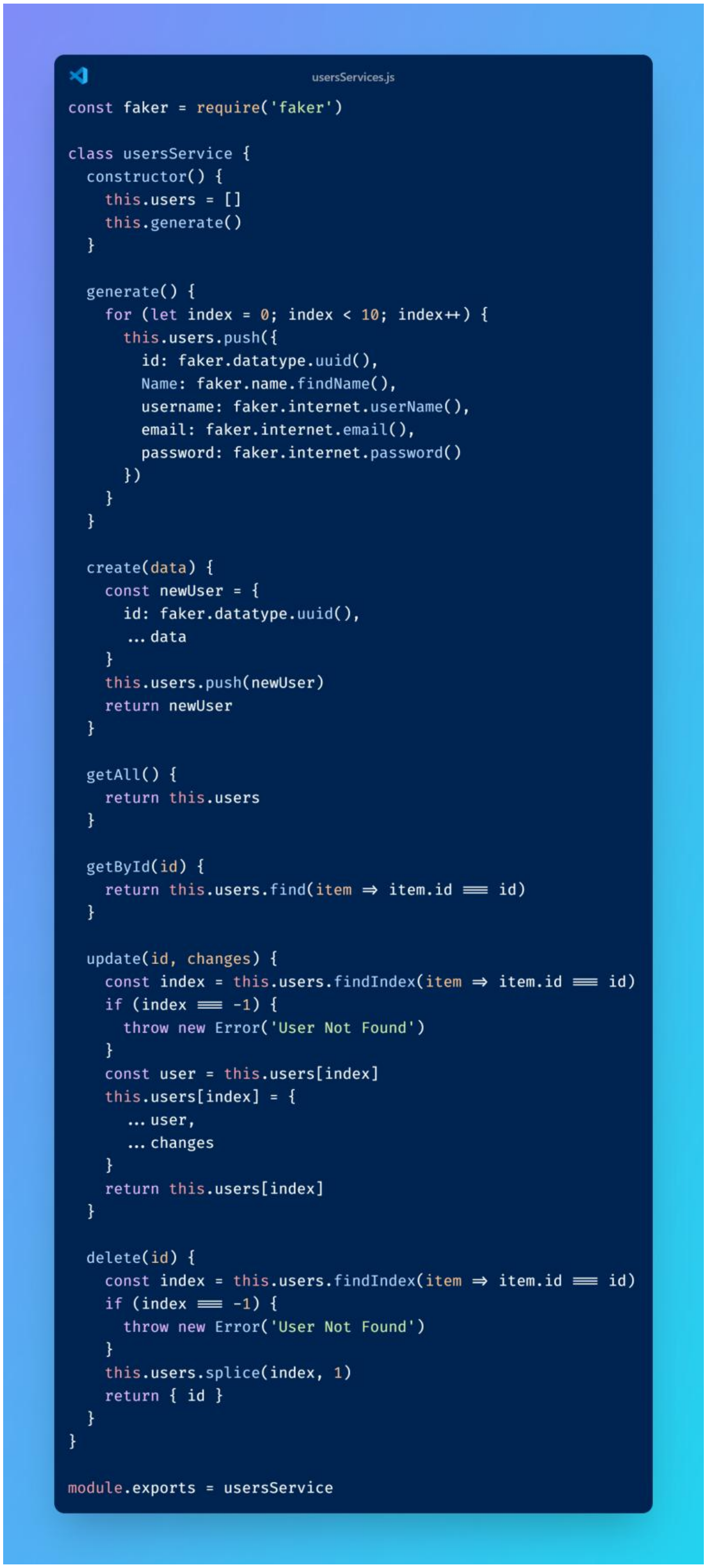
```javascript
// Archivo: /routes/rutas.js
const productsRouter = require('./productsRouter')
const UserRouter = require('./UsersRouter')
const Categories = require('./CategoriesRouter')
const Brands = require('./BrandsRouter')

function routerApi(app) {
  app.use('/products', productsRouter);
  app.use('/users', UserRouter);
  app.use('/categories', Categories);
  app.use('/brands', Brands);
}

module.exports = routerApi;
```

Implementación de la Capa de Servicios (Services)

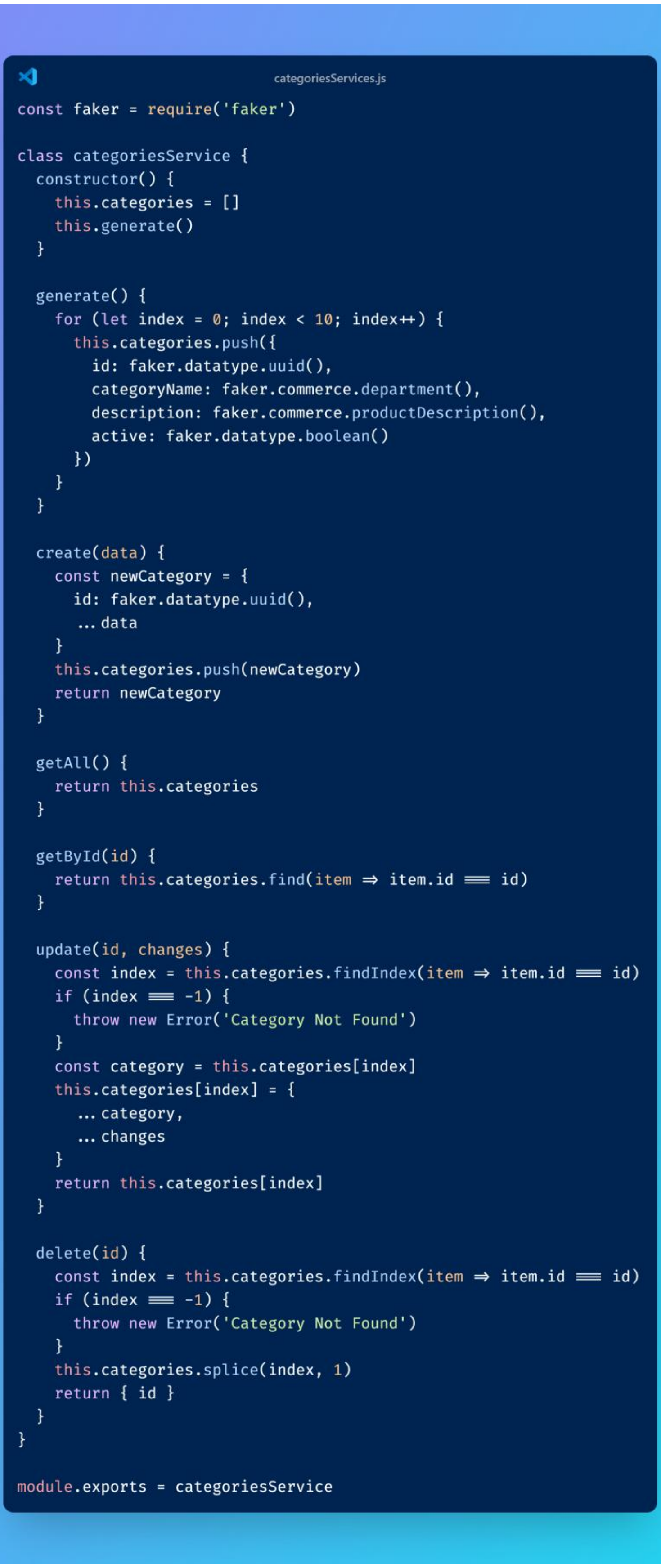
Esta es la adición principal de la refactorización. Se crea una clase de servicio para cada entidad (Products, Users, Categories, Brands). Estas clases encapsulan toda la lógica de negocio y el manejo de datos (en este caso, generados con faker).



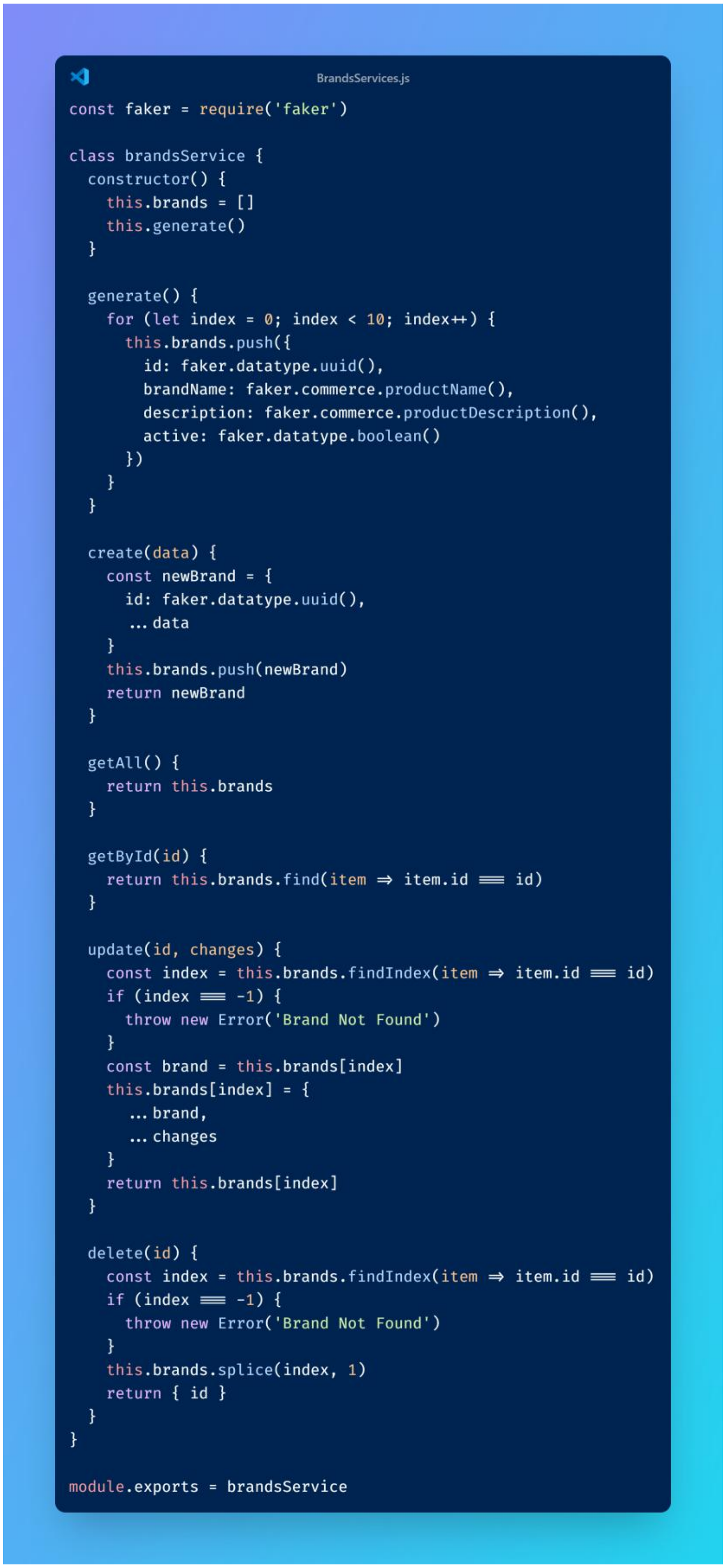
```javascript
const faker = require("faker")

class productsService {
  constructor() {
    this.products = []
    this.generate()
  }

  generate() {
    for (let index = 0; index < 10; index++) {
      this.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 })
      });
    }
  }

  create(data) {
    const newProduct = {
      id: faker.datatype.uuid(),
      ...data
    }
    this.products.push(newProduct)
    return newProduct
  }

  getAll(){
    return this.products
  }

  getById(id){
    return this.products.find(item => item.id === id)
  }

  update(id, changes) {
    const index = this.products.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('Product Not Found')
    }
    const product = this.products[index]
    this.products[index] = {
      ...product,
      ...changes
    }
    return this.products[index]
  }

  delete(id) {
    const index = this.products.findIndex(item)
    if (index === -1 ){
      throw new Error('Product Not Found')
    }
    this.products.splice(index, 1)
    return { id }
  }
}

module.exports = productsService;
```



```javascript
const faker = require('faker')

class usersService {
  constructor() {
    this.users = []
    this.generate()
  }

  generate() {
    for (let index = 0; index < 10; index++) {
      this.users.push({
        id: faker.datatype.uuid(),
        Name: faker.name.findName(),
        username: faker.internet.userName(),
        email: faker.internet.email(),
        password: faker.internet.password()
      })
    }
  }

  create(data) {
    const newUser = {
      id: faker.datatype.uuid(),
      ...data
    }
    this.users.push(newUser)
    return newUser
  }

  getAll() {
    return this.users
  }

  getById(id) {
    return this.users.find(item => item.id === id)
  }

  update(id, changes) {
    const index = this.users.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('User Not Found')
    }
    const user = this.users[index]
    this.users[index] = {
      ...user,
      ...changes
    }
    return this.users[index]
  }

  delete(id) {
    const index = this.users.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('User Not Found')
    }
    this.users.splice(index, 1)
    return { id }
  }
}

module.exports = usersService
```



```javascript
const faker = require('faker')

class categoriesService {
  constructor() {
    this.categories = []
    this.generate()
  }

  generate() {
    for (let index = 0; index < 10; index++) {
      this.categories.push({
        id: faker.datatype.uuid(),
        categoryName: faker.commerce.department(),
        description: faker.commerce.productDescription(),
        active: faker.datatype.boolean()
      })
    }
  }

  create(data) {
    const newCategory = {
      id: faker.datatype.uuid(),
      ...data
    }
    this.categories.push(newCategory)
    return newCategory
  }

  getAll() {
    return this.categories
  }

  getById(id) {
    return this.categories.find(item => item.id === id)
  }

  update(id, changes) {
    const index = this.categories.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('Category Not Found')
    }
    const category = this.categories[index]
    this.categories[index] = {
      ...category,
      ...changes
    }
    return this.categories[index]
  }

  delete(id) {
    const index = this.categories.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('Category Not Found')
    }
    this.categories.splice(index, 1)
    return { id }
  }
}

module.exports = categoriesService
```

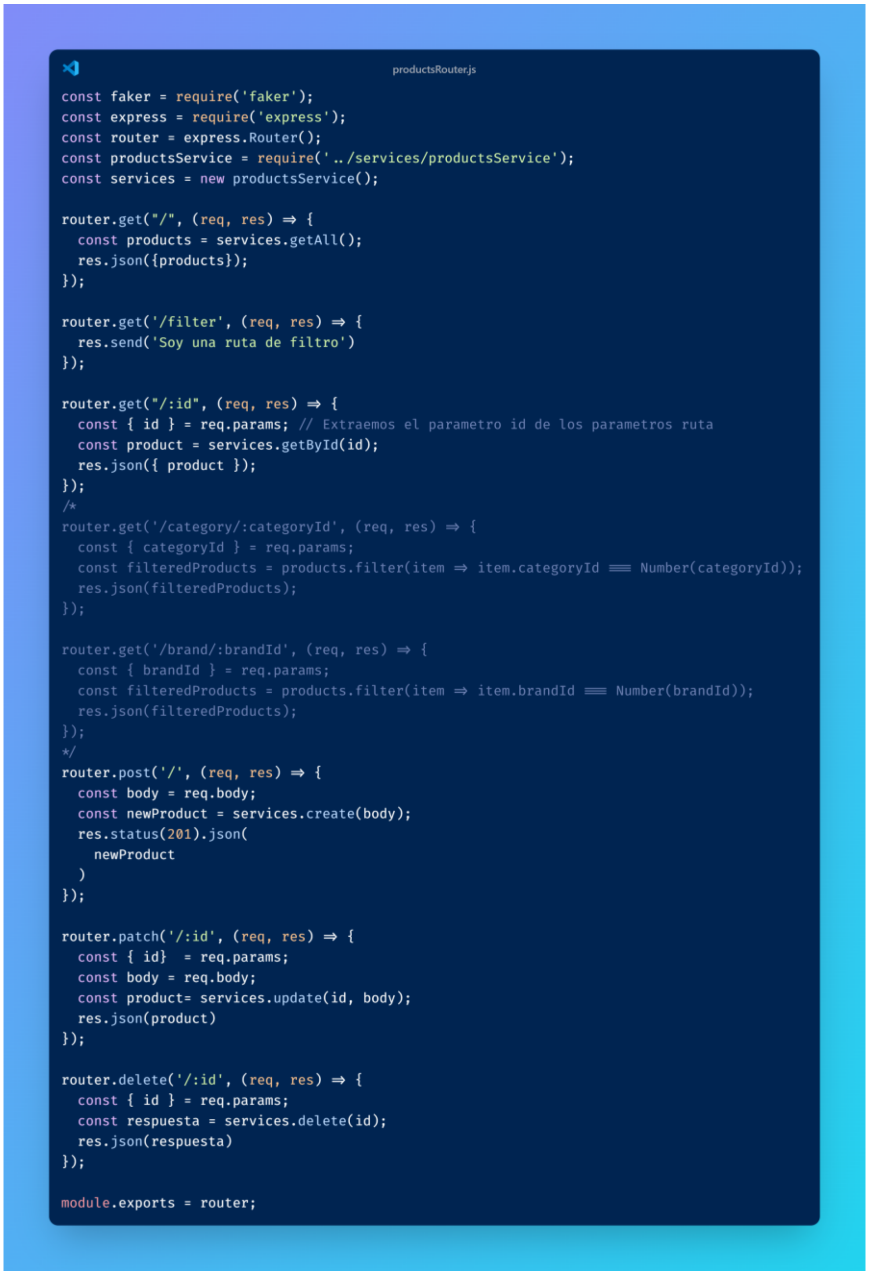


```javascript
const faker = require('faker')

class brandsService {
  constructor() {
    this.brands = []
    this.generate()
  }

  generate() {
    for (let index = 0; index < 10; index++) {
      this.brands.push({
        id: faker.datatype.uuid(),
        brandName: faker.commerce.productName(),
        description: faker.commerce.productDescription(),
        active: faker.datatype.boolean()
      })
    }
  }

  create(data) {
    const newBrand = {
      id: faker.datatype.uuid(),
      ...data
    }
    this.brands.push(newBrand)
    return newBrand
  }

  getAll() {
    return this.brands
  }

  getById(id) {
    return this.brands.find(item => item.id === id)
  }

  update(id, changes) {
    const index = this.brands.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('Brand Not Found')
    }
    const brand = this.brands[index]
    this.brands[index] = {
      ...brand,
      ...changes
    }
    return this.brands[index]
  }

  delete(id) {
    const index = this.brands.findIndex(item => item.id === id)
    if (index === -1) {
      throw new Error('Brand Not Found')
    }
    this.brands.splice(index, 1)
    return { id }
  }
}

module.exports = brandsService
```

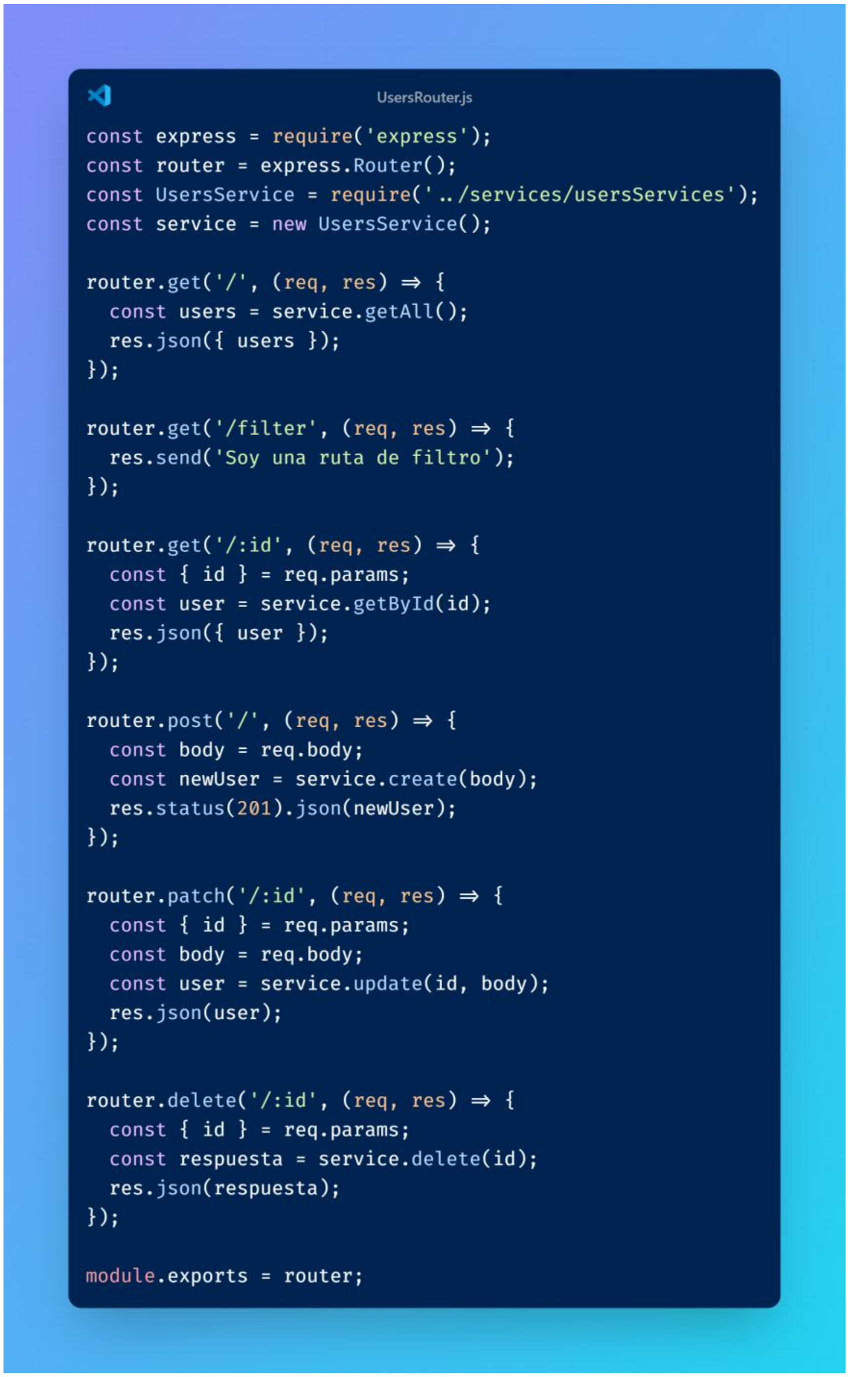
Nota: Los archivos usersServices.js, categoriesServices.js y brandsServices.js siguen esta misma estructura de clase.

Implementación de Rutas (Controllers)

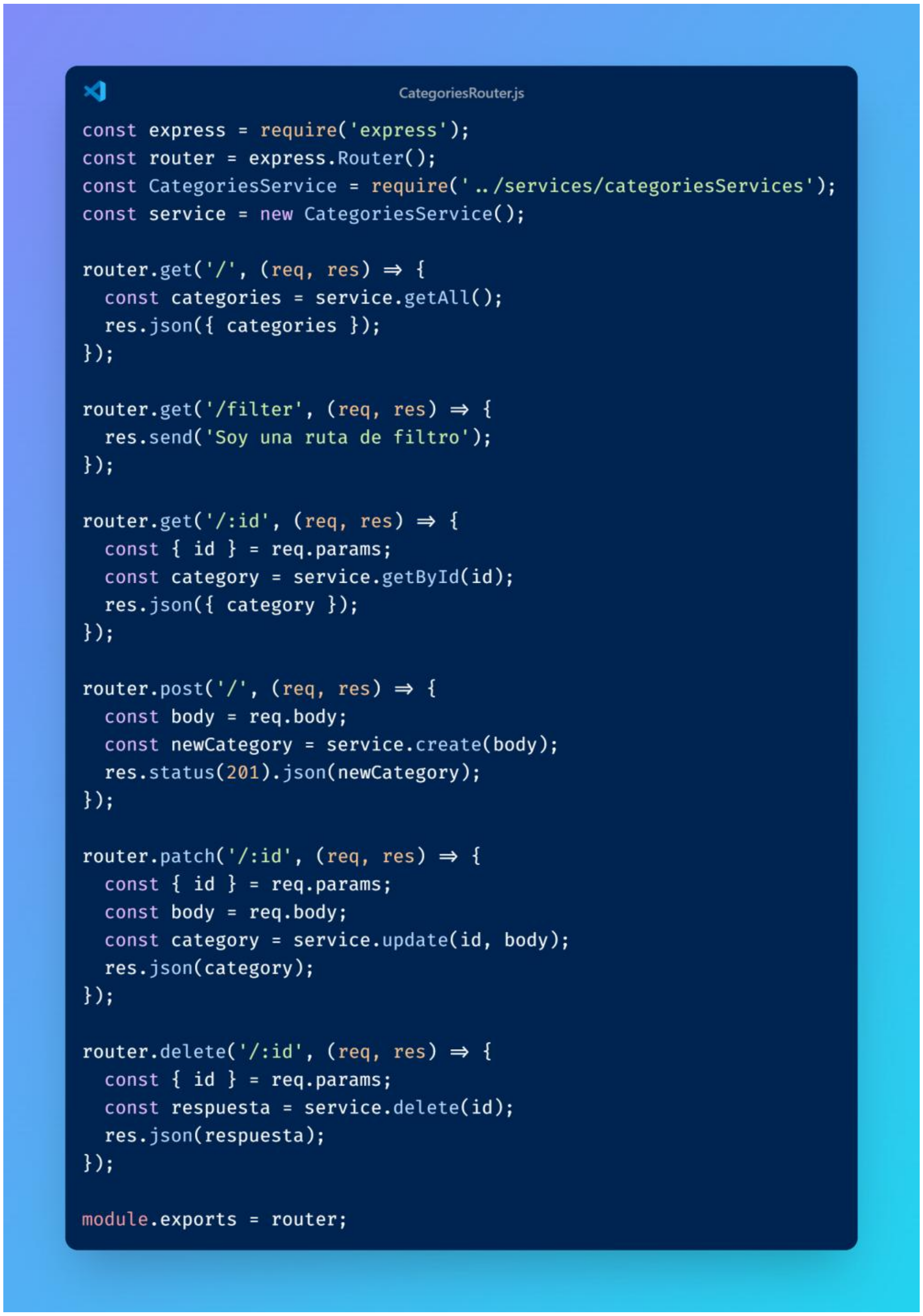
Con la lógica de negocio ahora en los servicios, los archivos de rutas (ej. productsRouter.js) se convierten en **"controladores delgados" (thin controllers)**.



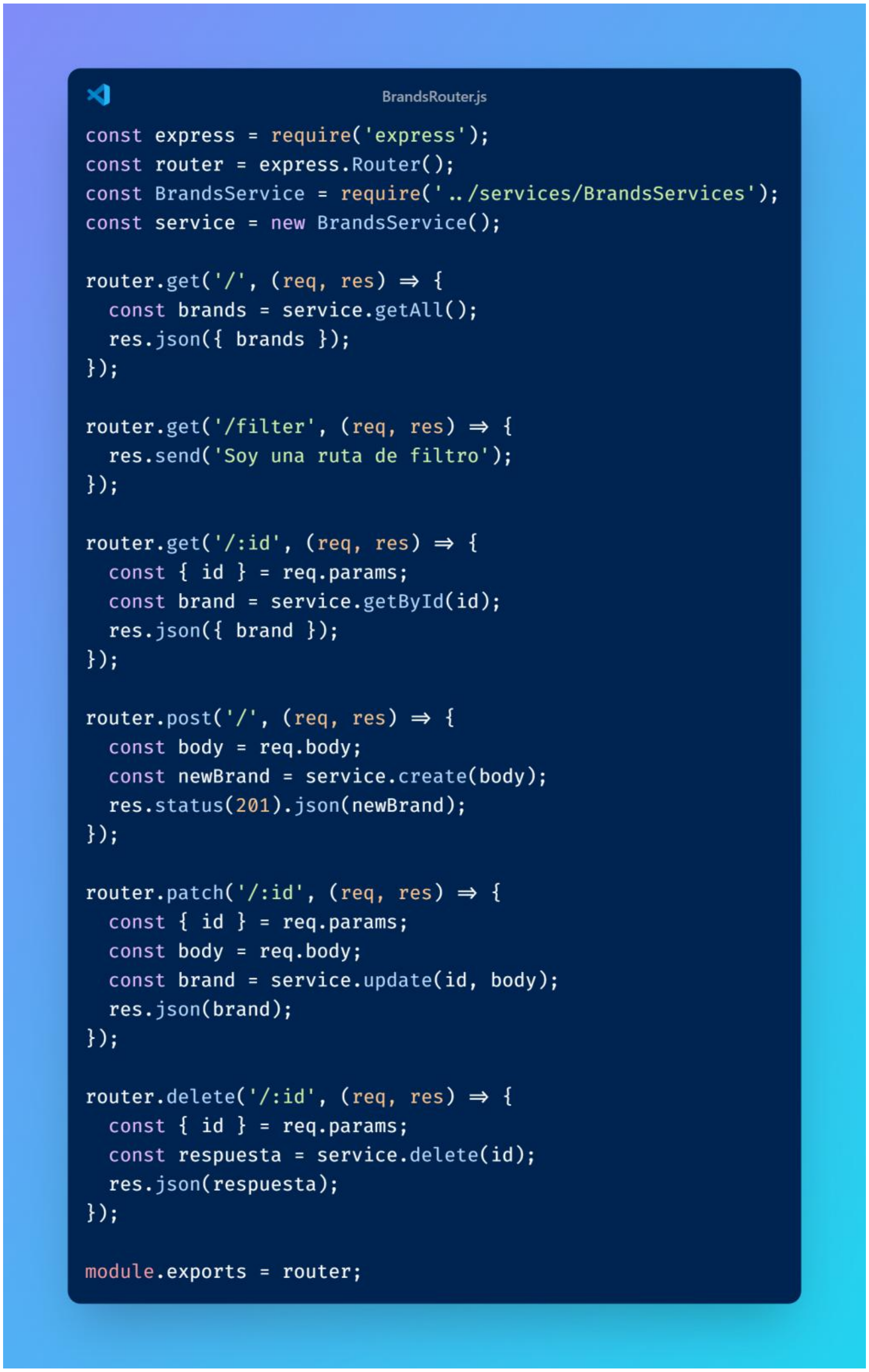
```javascript
const faker = require('faker');
const express = require('express');
const router = express.Router();
const productsService = require('../services/productsService');
const services = new productsService();

router.get("/", (req, res) => {
  const products = services.getAll();
  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
  const product = services.getById(id);
  res.json({ product });
});
/*
router.get('/category/:categoryId', (req, res) => {
  const { categoryId } = req.params;
  const filteredProducts = products.filter(item => item.categoryId === Number(categoryId));
  res.json(filteredProducts);
});

router.get('/brand/:brandId', (req, res) => {
  const { brandId } = req.params;
  const filteredProducts = products.filter(item => item.brandId === Number(brandId));
  res.json(filteredProducts);
});
*/
router.post('/', (req, res) => {
  const body = req.body;
  const newProduct = services.create(body);
  res.status(201).json(
    newProduct
  )
});

router.patch('/:id', (req, res) => {
  const { id}  = req.params;
  const body = req.body;
  const product= services.update(id, body);
  res.json(product)
});

router.delete('/:id', (req, res) => {
  const { id } = req.params;
  const respuesta = services.delete(id);
  res.json(respuesta)
});

module.exports = router;
```



```javascript
const express = require('express');
const router = express.Router();
const UsersService = require('../services/usersServices');
const service = new UsersService();

router.get('/', (req, res) => {
  const users = service.getAll();
  res.json({ users });
});

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

router.get('/:id', (req, res) => {
  const { id } = req.params;
  const user = service.getById(id);
  res.json({ user });
});

router.post('/', (req, res) => {
  const body = req.body;
  const newUser = service.create(body);
  res.status(201).json(newUser);
});

router.patch('/:id', (req, res) => {
  const { id } = req.params;
  const body = req.body;
  const user = service.update(id, body);
  res.json(user);
});

router.delete('/:id', (req, res) => {
  const { id } = req.params;
  const respuesta = service.delete(id);
  res.json(respuesta);
});

module.exports = router;
```



```javascript
const express = require('express');
const router = express.Router();
const CategoriesService = require('../services/categoriesServices');
const service = new CategoriesService();

router.get('/', (req, res) => {
  const categories = service.getAll();
  res.json({ categories });
});

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

router.get('/:id', (req, res) => {
  const { id } = req.params;
  const category = service.getById(id);
  res.json({ category });
});

router.post('/', (req, res) => {
  const body = req.body;
  const newCategory = service.create(body);
  res.status(201).json(newCategory);
});

router.patch('/:id', (req, res) => {
  const { id } = req.params;
  const body = req.body;
  const category = service.update(id, body);
  res.json(category);
});

router.delete('/:id', (req, res) => {
  const { id } = req.params;
  const respuesta = service.delete(id);
  res.json(respuesta);
});

module.exports = router;
```



BrandsRouter.js

```javascript
const express = require('express');
const router = express.Router();
const BrandsService = require('../services/BrandsServices');
const service = new BrandsService();

router.get('/', (req, res) => {
  const brands = service.getAll();
  res.json({ brands });
});

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

router.get('/:id', (req, res) => {
  const { id } = req.params;
  const brand = service.getById(id);
  res.json({ brand });
});

router.post('/', (req, res) => {
  const body = req.body;
  const newBrand = service.create(body);
  res.status(201).json(newBrand);
});

router.patch('/:id', (req, res) => {
  const { id } = req.params;
  const body = req.body;
  const brand = service.update(id, body);
  res.json(brand);
});

router.delete('/:id', (req, res) => {
  const { id } = req.params;
  const respuesta = service.delete(id);
  res.json(respuesta);
});

module.exports = router;
```

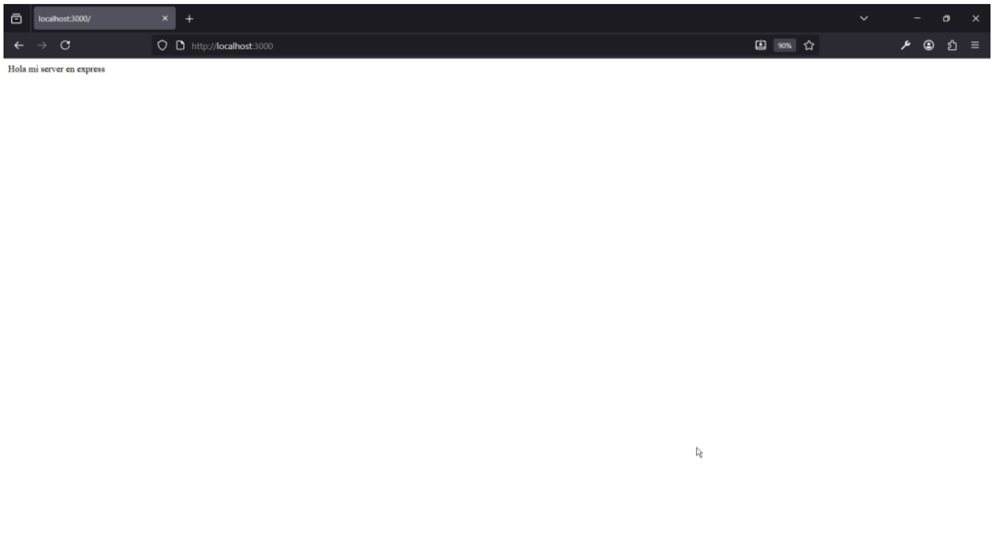
Resultados Obtenidos

Endpoints Principales

Prueba del uso de las funciones CRUD

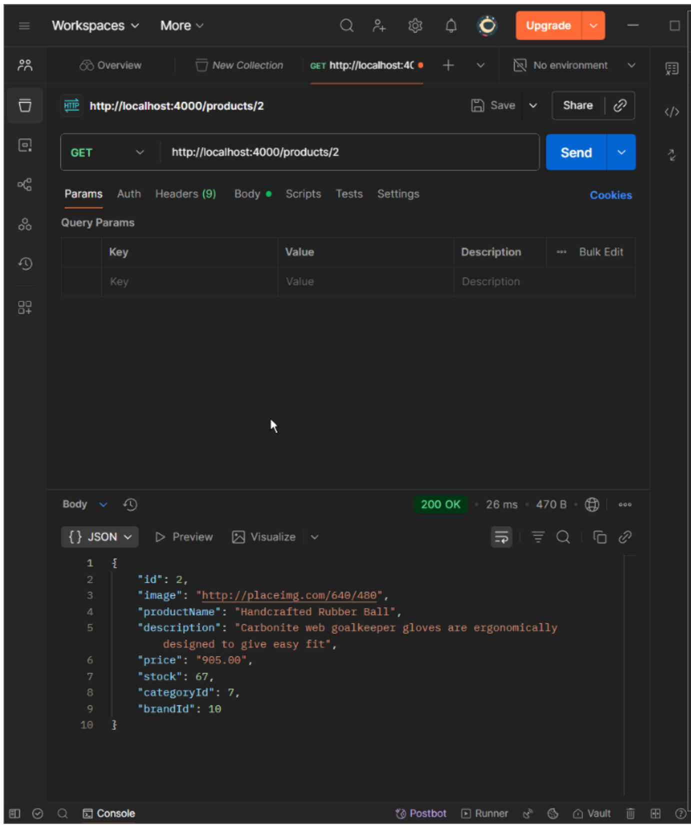
1. **Endpoint Raíz**: GET / - Mensaje de bienvenida
2. **Products**: Operaciones CRUD completas con filtros por categoría y marca
3. **Users**: Gestión completa de usuarios
4. **Categories**: Administración de categorías
5. **Brands**: Administración de marcas
6. **Movies**: Módulo adicional para gestión de películas

Resultados 1 - Vista principal de la documentación automática



Hola mi server en express

Resultados 2 – vista CRUD para products



http://localhost:4000/products/2

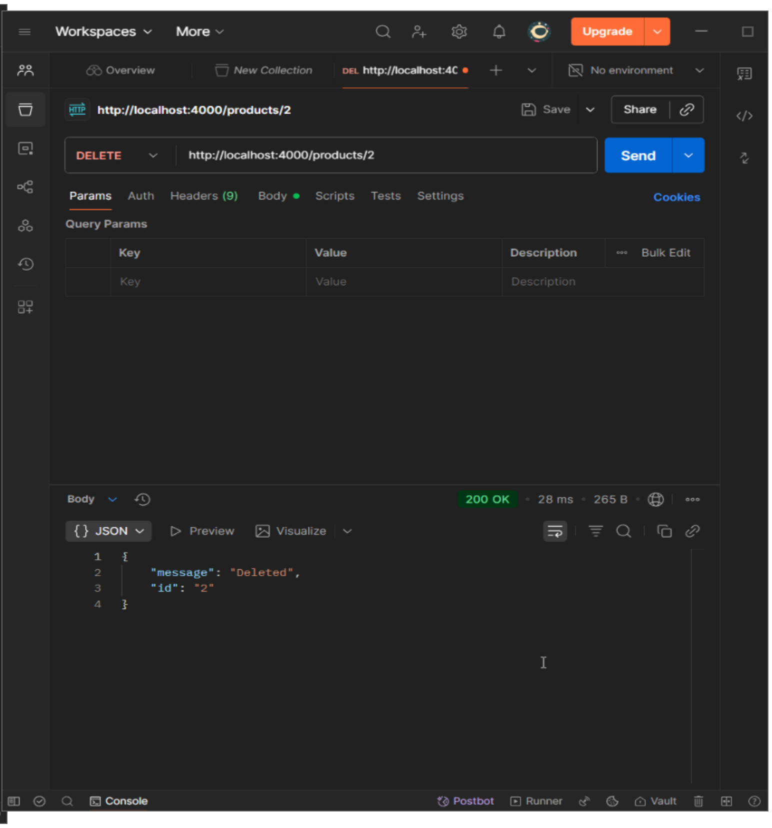
GET http://localhost:4000/products/2

Query Params

Key	Value	Description
Key	Value	Description

Body · JSON · 200 OK · 26 ms · 470 B

```
{
    "id": 2,
    "image": "http://placeimg.com/640/480",
    "productName": "Handcrafted Rubber Ball",
    "description": "Carbonite web goalkeeper gloves are ergonomically
        designed to give easy fit",
    "price": "905.00",
    "stock": 67,
    "categoryId": 7,
    "brandId": 10
}
```



http://localhost:4000/products/2

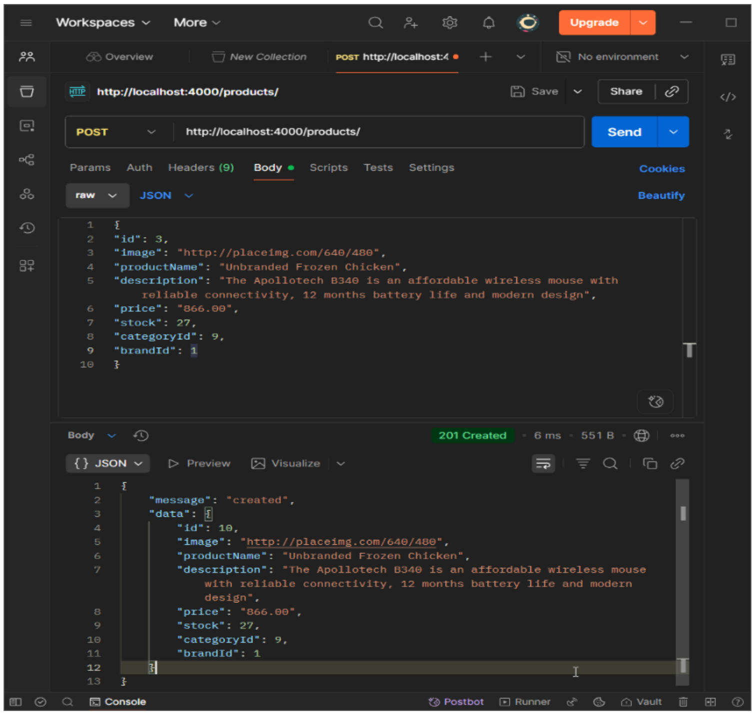
DELETE http://localhost:4000/products/2

Query Params

Key	Value	Description
Key	Value	Description

Body · JSON · 200 OK · 28 ms · 265 B

```
{
    "message": "Deleted",
    "id": "2"
}
```



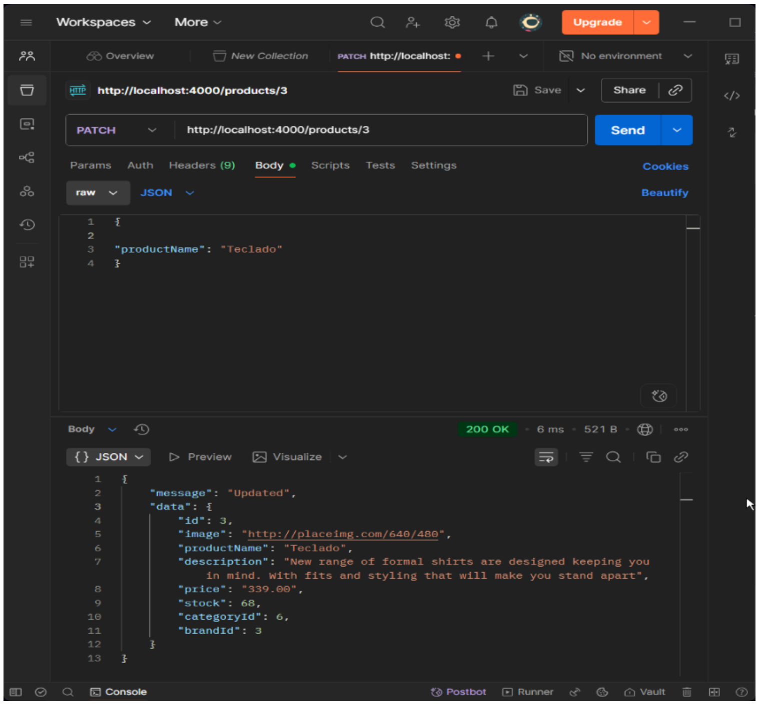
http://localhost:4000/products/

POST http://localhost:4000/products/

```
{
"id": 3,
"image": "http://placeimg.com/640/480",
"productName": "Unbranded Frozen Chicken",
"description": "The Apollotech B340 is an affordable wireless mouse with
    reliable connectivity, 12 months battery life and modern design",
"price": "866.00",
"stock": 27,
"categoryId": 9,
"brandId": 1
}
```

201 Created · 6 ms · 551 B

```
{
    "message": "created",
    "data": {
        "id": 10,
        "image": "http://placeimg.com/640/480",
        "productName": "Unbranded Frozen Chicken",
        "description": "The Apollotech B340 is an affordable wireless mouse
            with reliable connectivity, 12 months battery life and modern
            design",
        "price": "866.00",
        "stock": 27,
        "categoryId": 9,
        "brandId": 1
    }
}
```



http://localhost:4000/products/3

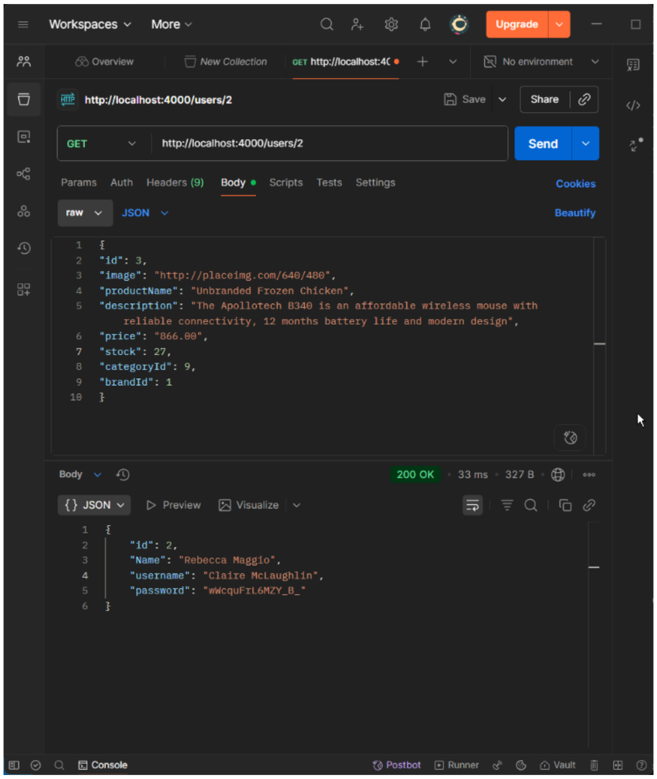
PATCH http://localhost:4000/products/3

```
{

"productName": "Teclado"
}
```

200 OK · 6 ms · 521 B

```
{
    "message": "Updated",
    "data": {
        "id": 3,
        "image": "http://placeimg.com/640/480",
        "productName": "Teclado",
        "description": "New range of formal shirts are designed keeping you
            in mind. With fits and styling that will make you stand apart",
        "price": "339.00",
        "stock": 68,
        "categoryId": 6,
        "brandId": 3
    }
}
```

Resultado 3 CRUD users



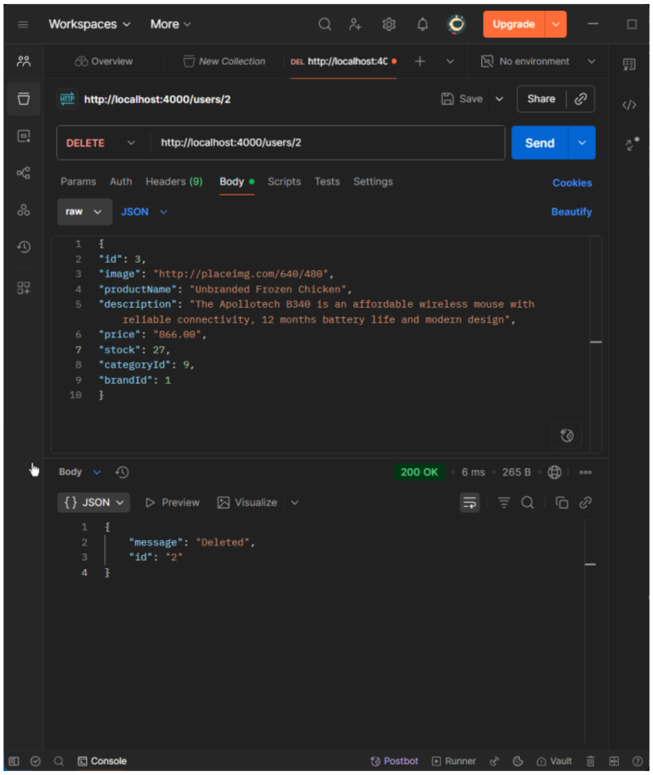
http://localhost:4000/users/2

GET http://localhost:4000/users/2

```
{
"id": 3,
"image": "http://placeimg.com/640/480",
"productName": "Unbranded Frozen Chicken",
"description": "The Apollotech B340 is an affordable wireless mouse with
    reliable connectivity, 12 months battery life and modern design",
"price": "866.00",
"stock": 27,
"categoryId": 9,
"brandId": 1
}
```

200 OK · 33 ms · 327 B

```
{
    "id": 2,
    "Name": "Rebecca Maggio",
    "username": "Claire McLaughlin",
    "password": "wWcquFrL6MZY_B_"
}
```



http://localhost:4000/users/2

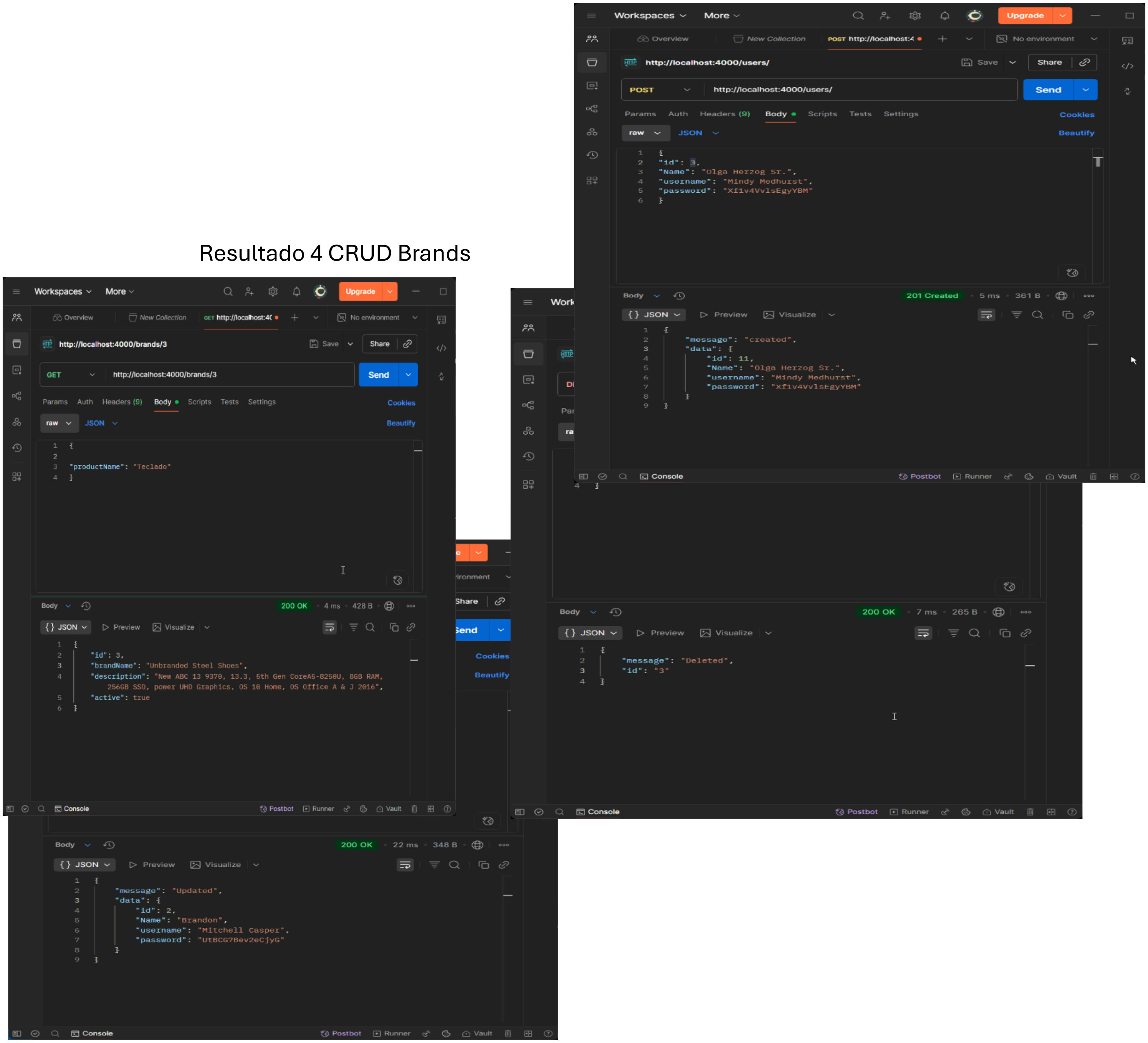
DELETE http://localhost:4000/users/2

```
{
"id": 3,
"image": "http://placeimg.com/640/480",
"productName": "Unbranded Frozen Chicken",
"description": "The Apollotech B340 is an affordable wireless mouse with
    reliable connectivity, 12 months battery life and modern design",
"price": "866.00",
"stock": 27,
"categoryId": 9,
"brandId": 1
}
```

200 OK · 6 ms · 265 B

```
{
    "message": "Deleted",
    "id": "2"
}
```

Resultado 4 CRUD Brands



POST http://localhost:4000/users/

```
{
"id": 3,
"Name": "Olga Herzog Sr.",
"username": "Mindy Medhurst",
"password": "Xf1v4VvlsEgyYBM"
}
```

201 Created · 5 ms · 361 B

```
{
    "message": "created",
    "data": {
        "id": 11,
        "Name": "Olga Herzog Sr.",
        "username": "Mindy Medhurst",
        "password": "Xf1v4VvlsEgyYBM"
    }
}
```

GET http://localhost:4000/brands/3

```
{

"productName": "Teclado"
}
```

200 OK · 4 ms · 428 B

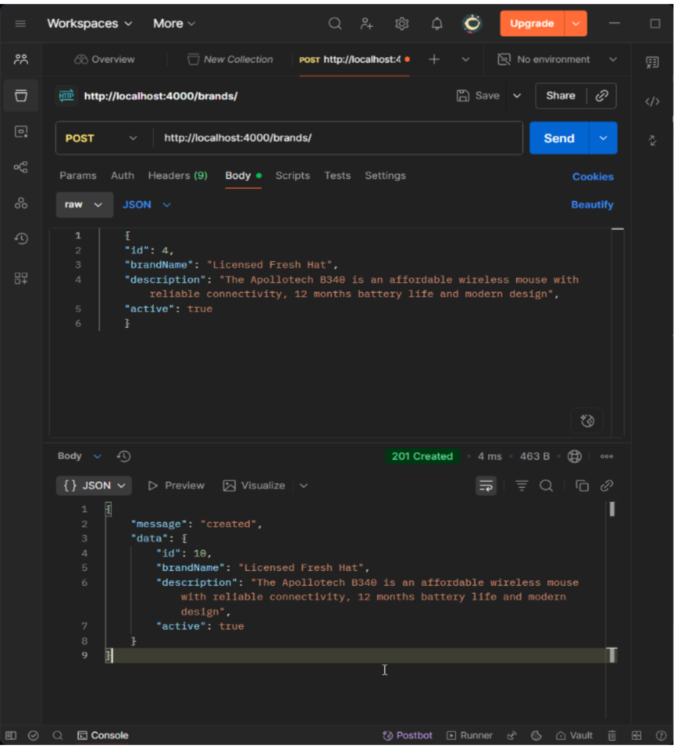
```
{
    "id": 3,
    "brandName": "Unbranded Steel Shoes",
    "description": "New ABC 13 9370, 13.3, 5th Gen CoreA5-8250U, 8GB RAM,
        256GB SSD, power UHD Graphics, OS 10 Home, OS Office A & J 2016",
    "active": true
}
```

200 OK · 7 ms · 265 B

```
{
    "message": "Deleted",
    "id": "3"
}
```

200 OK · 22 ms · 348 B

```
{
    "message": "Updated",
    "data": {
        "id": 2,
        "Name": "Brandon",
        "username": "Mitchell Casper",
        "password": "UtBCG7Bev2eCjyG"
    }
}
```

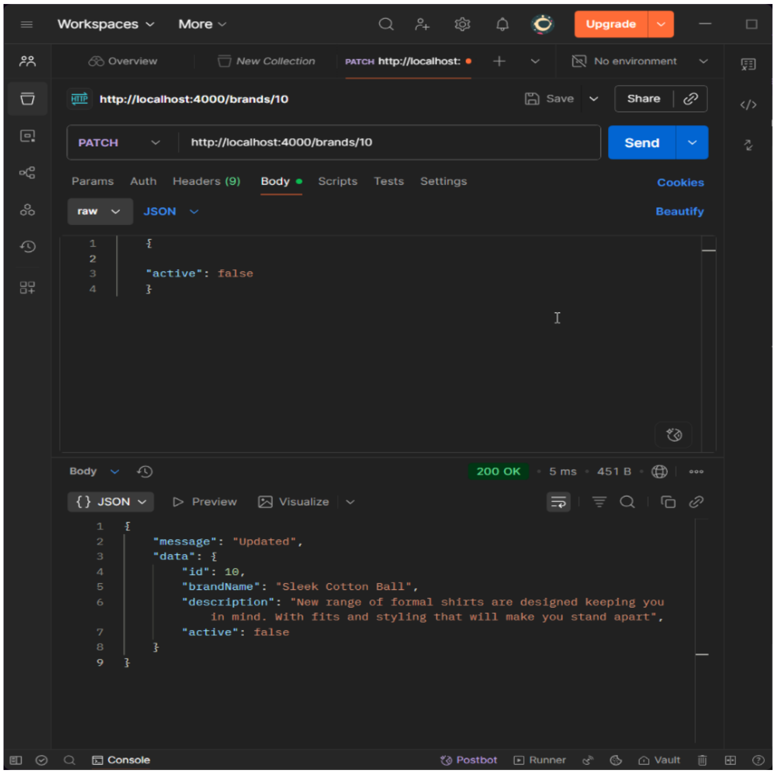


POST http://localhost:4000/brands/

```
{
"id": 4,
"brandName": "Licensed Fresh Hat",
"description": "The Apollotech B340 is an affordable wireless mouse with
   reliable connectivity, 12 months battery life and modern design",
"active": true
}
```

201 Created • 4 ms • 463 B

```
{
    "message": "created",
    "data": {
        "id": 10,
        "brandName": "Licensed Fresh Hat",
        "description": "The Apollotech B340 is an affordable wireless mouse
            with reliable connectivity, 12 months battery life and modern
            design",
        "active": true
    }
}
```



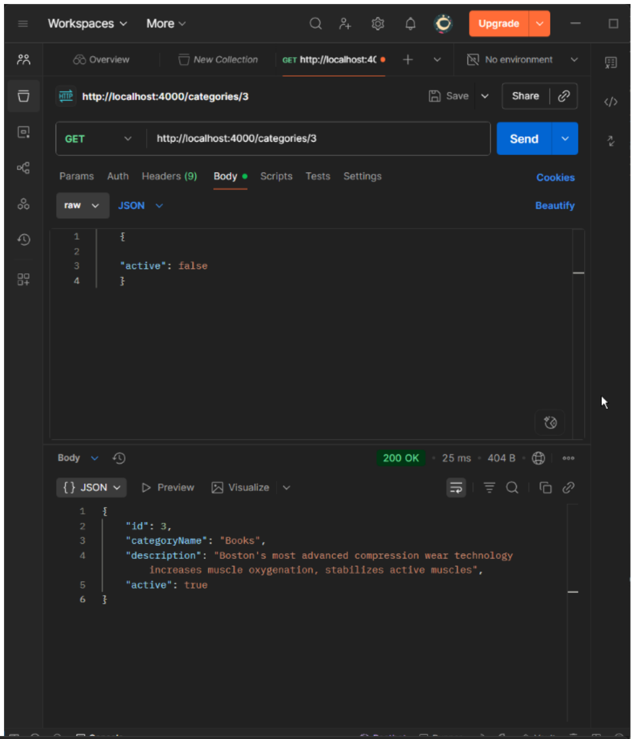
PATCH http://localhost:4000/brands/10

```
{

"active": false
}
```

200 OK • 5 ms • 451 B

```
{
    "message": "Updated",
    "data": {
        "id": 10,
        "brandName": "Sleek Cotton Ball",
        "description": "New range of formal shirts are designed keeping you
            in mind. With fits and styling that will make you stand apart",
        "active": false
    }
}
```

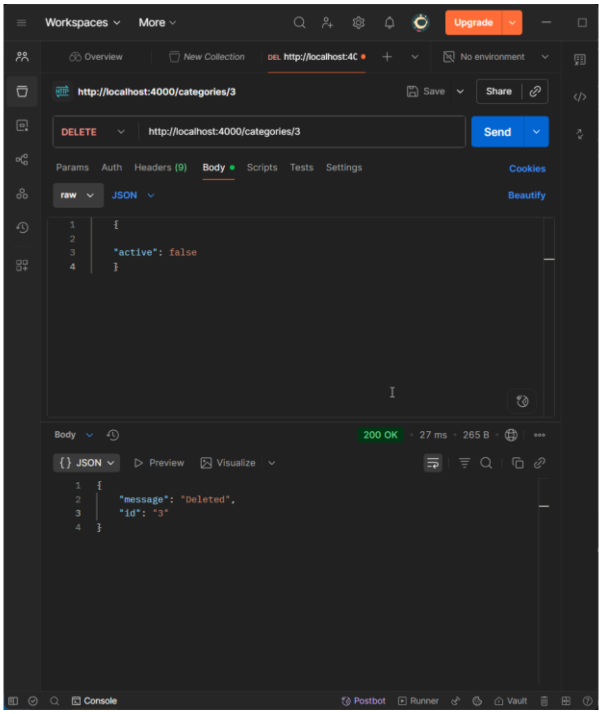
Resultado 4 CRUD category


GET http://localhost:4000/categories/3

```
{

"active": false
}
```

200 OK · 25 ms · 404 B

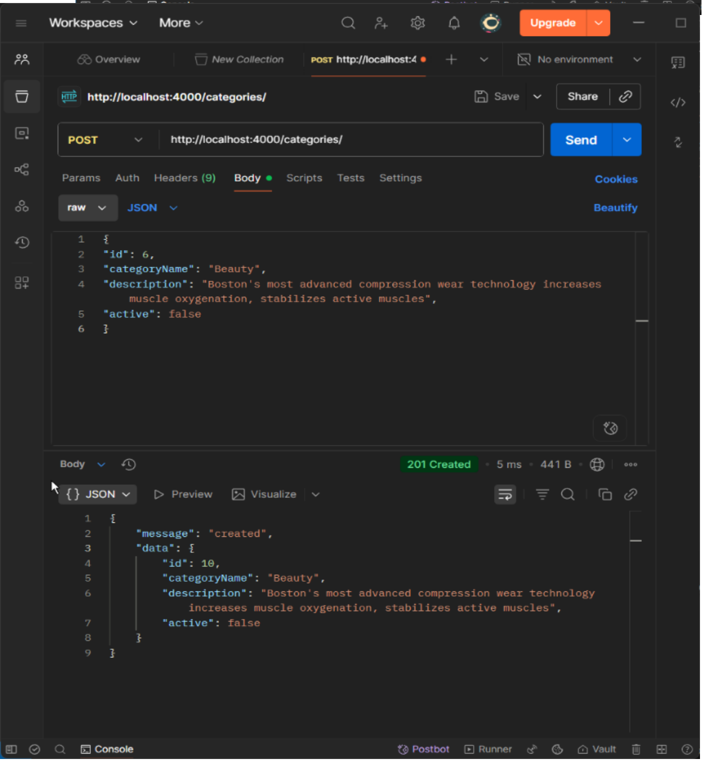
```
{
  "id": 3,
  "categoryName": "Books",
  "description": "Boston's most advanced compression wear technology
      increases muscle oxygenation, stabilizes active muscles",
  "active": true
}
```


DELETE http://localhost:4000/categories/3

```
{

"active": false
}
```

200 OK · 27 ms · 265 B

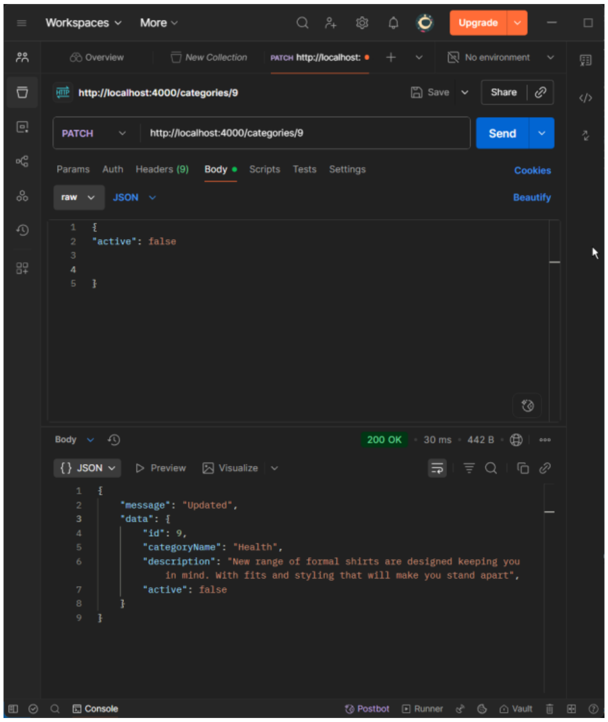
```
{
   "message": "Deleted",
   "id": "3"
}
```


POST http://localhost:4000/categories/

```
{
"id": 6,
"categoryName": "Beauty",
"description": "Boston's most advanced compression wear technology increases
    muscle oxygenation, stabilizes active muscles",
"active": false
}
```

201 Created · 5 ms · 441 B

```
{
    "message": "created",
    "data": {
        "id": 10,
        "categoryName": "Beauty",
        "description": "Boston's most advanced compression wear technology
            increases muscle oxygenation, stabilizes active muscles",
        "active": false
    }
}
```


PATCH http://localhost:4000/categories/9

```
{
"active": false



}
```

200 OK · 30 ms · 442 B

```
{
    "message": "Updated",
    "data": {
        "id": 9,
        "categoryName": "Health",
        "description": "New range of formal shirts are designed keeping you
            in mind. With fits and styling that will make you stand apart",
        "active": false
    }
}
```

Conclusiones

Este proyecto se desarrolló lo siguiente para el correcto funcionamiento del proyecto y también para el aprendizaje de una ApiRest:

1. **Aprendizaje Práctico**: Desarrollo manual desde cero de un servidor Express
2. **Modularización**: Organización efectiva del código en routers especializados
3. **API REST Completa**: Implementación de todos los verbos HTTP y operaciones CRUD
4. **Manejo de Dependencias**: Uso correcto de npm para gestión de paquetes
5. **Calidad de Código**: Configuración de herramientas como ESLint y Prettier
6. **Documentación**: Creación de reportes detallados del progreso
7. ⍰ **Separación de Responsabilidades (SoC**): Se logró el objetivo principal de separar la capa de presentación (rutas/controladores) de la capa de lógica de negocio (servicios).
8. **Mantenibilidad:** El código es mucho más fácil de mantener. Si se necesita cambiar la lógica de cómo se crea un usuario (ej. añadir validación o hashear una contraseña), solo se modifica usersServices.js sin tocar UsersRouter.js.
9. **Escalabilidad**: El proyecto está ahora preparado para escalar. Si en el futuro se decide conectar a una base de datos (como MongoDB o PostgreSQL), solo será necesario modificar los métodos dentro de las clases de servicio, sin afectar las rutas.
10. **Controladores Delgados:** Se implementó el patrón "Thin Controller", donde las rutas son limpias, legibles y solo se ocupan de orquestar el flujo de la petición-respuesta.