

índice

Tabla de contenido

[**Introducción y** Objetivo 3](#_Toc210216296)

[**Requerimientos Previos** 3](#_Toc210216297)

[**Desarrollo** 4](#_Toc210216298)

[**Estructura del Proyecto** 5](#_Toc210216299)

[**Implementación del Servidor Principal (index.js)** 5](#_Toc210216300)

[**Sistema de Enrutamiento Modular (rutas.js)** 6](#_Toc210216301)

[**Módulos Implementados** 6](#_Toc210216302)

[**Características Técnicas** 8](#_Toc210216303)

[**Ejecución del Proyecto** 8](#_Toc210216304)

[**Resultados Obtenidos** 9](#_Toc210216305)

[**Conclusiones** 15](#_Toc210216306)

# **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**

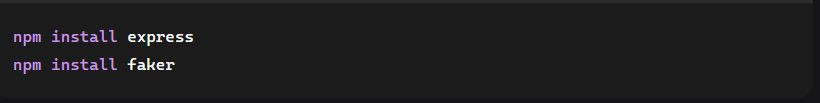
Para el desarrollo de este proyecto se requirió:

* **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:



Se configuró un archivo .gitignore utilizando [gitignore.io](https://gitignore.io/) para excluir archivos innecesarios del control de versiones.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# **Estructura del Proyecto**

Texto

El contenido generado por IA puede ser incorrecto.

# **Implementación del Servidor Principal (index.js)**

El archivo principal configura el servidor Express:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# **Sistema de Enrutamiento Modular (rutas.js)**

Archivo centralizador que organiza todas las rutas:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# **Módulos Implementados**

**1. ProductsRouter.js**

* **Endpoints**:
  + GET /products - Lista todos los productos
  + GET /products/:id - Obtiene producto por ID
  + GET /products/category/:categoryId - Filtra por categoría
  + GET /products/brand/:brandId - Filtra por marca
  + POST /products - Crea nuevo producto
  + PATCH /products/:id - Actualiza producto
  + DELETE /products/:id - Elimina producto

**2. UsersRouter.js**

* **Endpoints**:
  + GET /users - Lista todos los usuarios
  + GET /users/:id - Obtiene usuario por ID
  + POST /users - Crea nuevo usuario
  + PATCH /users/:id - Actualiza usuario
  + DELETE /users/:id - Elimina usuario

**3. CategoriesRouter.js**

* **Endpoints**:
  + GET /categories - Lista todas las categorías
  + GET /categories/:id - Obtiene categoría por ID
  + POST /categories - Crea nueva categoría
  + PATCH /categories/:id - Actualiza categoría
  + DELETE /categories/:id - Elimina categoría

**4. BrandsRouter.js**

* **Endpoints**:
  + GET /brands - Lista todas las marcas
  + GET /brands/:id - Obtiene marca por ID
  + POST /brands - Crea nueva marca
  + PATCH /brands/:id - Actualiza marca
  + DELETE /brands/:id - Elimina marca

**5. MoviesRouter.js (Módulo Adicional)**

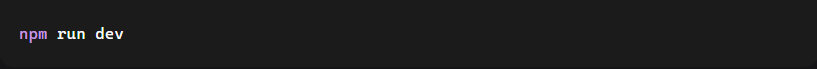
* **Endpoints**:
  + GET /movies - Lista todas las películas
  + GET /movies/:id - Obtiene película por ID
  + POST /movies - Crea nueva película
  + PATCH /movies/:id - Actualiza película
  + DELETE /movies/:id - Elimina película

# **Características Técnicas**

* **Generación de Datos**: Uso de Faker.js para datos de prueba
* **Validaciones**: Manejo de errores 404 para recursos no encontrados
* **Métodos HTTP**: Implementación completa de GET, POST, PATCH, DELETE
* **Estructura Modular**: Código organizado y mantenible
* **Middleware**: Parseo automático de JSON

# **Ejecución del Proyecto**

Para ejecutar el proyecto:



El servidor se inicia en: <http://localhost:4000>

# **Resultados Obtenidos**

**Endpoints Principales**

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

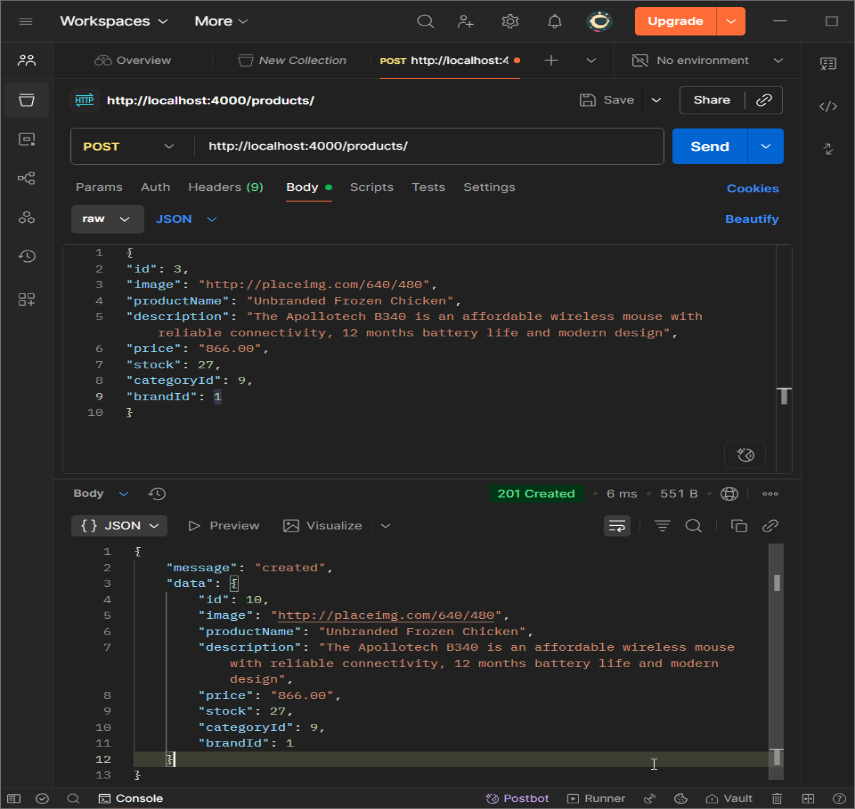
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Resultados 2 – vista CRUD para products

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Resultado 3 CRUD users

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

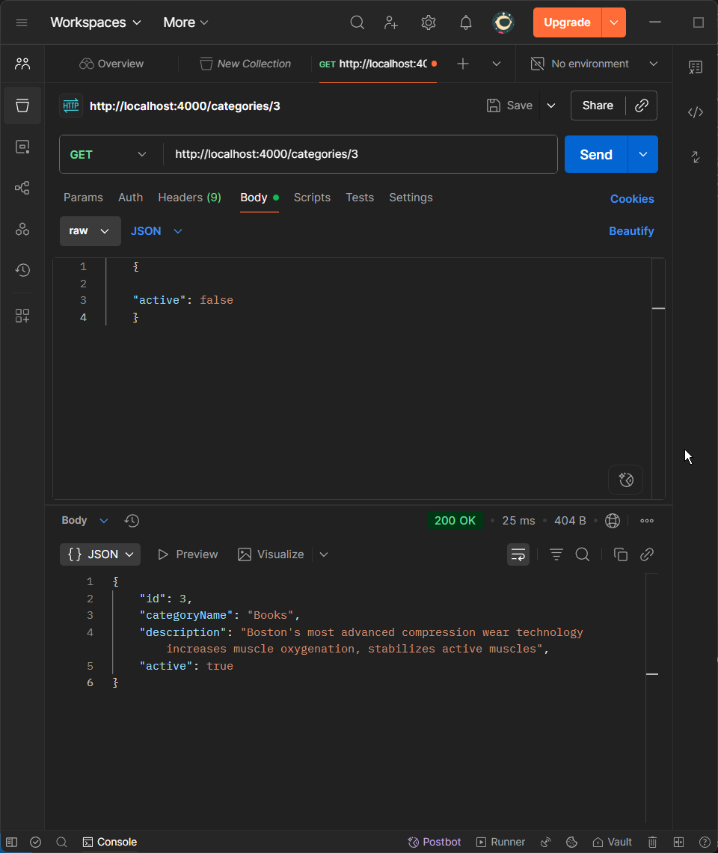
El contenido generado por IA puede ser incorrecto.Resultado 4 CRUD Brands

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Resultado 4 CRUD category



Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

# **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