**DISEÑO Y DESARROLLO DE SERVICIOS WEB - PROYECTO**

**APRENDIZ**

**BRAYAN STIVEN PEÑA QUINAYAS**

**ADSO**

**SENA**

**2023**

**INTRODUCCION**

En el contexto del proyecto formativo, nos proponemos diseñar y desarrollar APIs para nuestra tienda de videojuegos online. Se utilizará JavaScript como lenguaje de programación principal, aprovechando su versatilidad y amplio uso en el desarrollo web moderno. El objetivo es crear servicios web robustos y escalables que permitan gestionar productos, usuarios, pedidos y otras funcionalidades esenciales de una tienda online.

**HERRAMIENTAS:**

* Entorno de desarrollo: Visual Studio Code
* Herramienta de Versionamiento: Git, Github
* Documentación: Postman

**TECNOLOGIAS Y BIBLIOTECAS:**

* Node.js y Express para backend.
* MongoDB para almacenamiento de datos.

1. **DEFINICION DE REQUISITOS Y CARACTERISTICAS DEL SOFTWARE**

**Análisis de requisitos:**

Para una tienda online, los requisitos básicos serian:

* **Gestión de Productos:** Permitir añadir, modificar, eliminar y listar productos.
* **Gestión de Usuarios:** Registro, inicio de sesión, edición de perfil y gestión de pedidos.
* **Carrito de Compras:** Añadir productos al carito, modificar cantidades y finalizar compras.
* **Procesamiento de Pagos: i**ntegración con pasarelas de pago para procesar transacciones.
* **Seguridad:** Autenticación y autorización de usuarios, encriptación de datos sensibles.

**Selección de Tecnologías:**

Dado que se utilizará JavaScript podremos optar por las siguientes tecnologías:

* **Backend:** Node.js con express para crear el servidor y gestionar las APIs.
* **Base de datos:** MongoDB como base de datos para almacenar productos usuarios y pedidos.
* **Frontend:** React
* **Pasarela de pago:** PayPal o Stripe.

1. **DISEÑO DE LAS APIs**

**Definición de Endpoints:**

Basándonos en los requisitos identificados, podemos definir los siguientes endpoints para nuestra tienda online:

**PRODUCTOS:**

* **`GET / productos: `** Obtener la lista de todos los productos.
* **`GET / productos/:id`:** Actualizar un producto existente por ID.
* **`POST /productos`:** Añadir un nuevo producto.
* **`PUT / productos/:id`:** Actualizar un producto existente por ID.
* **`DELETE /productos/id`:** Eliminar un producto ID.

**USUARIOS:**

* **`POST /usuarios/registro`:** Registrar un nuevo usuario**.**
* **`POST /usuarios/login`:** Iniciar sesión de un usuario.
* **`GET /usuarios/id`:** Obtener información de un usuario por ID.
* **`PUT /usuarios/id`:** Actualizar información de un usuario por ID.

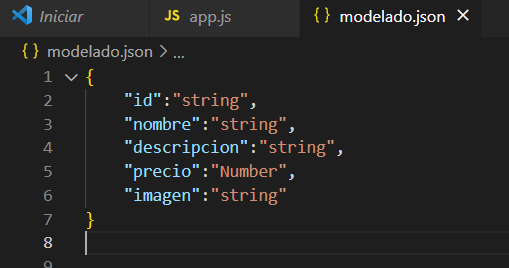
**PEDIDOS:**

* **`POST /pedidos`:** Crear un nuevo pedido.
* **`GET /pedidos/id`:** Obtener detalles de un pedido por ID.
* **`PUT /pedidos/:id`:** Actualizar estado de un pedido por ID.

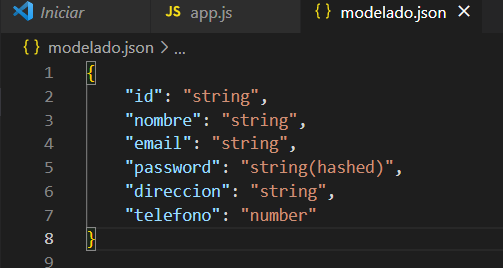
**Modelado de datos:**

Para el modelado de datos, consideramos los siguientes esquemas en formato JSON:

**PRODUCTO:**

****

**USUARIO:**

****

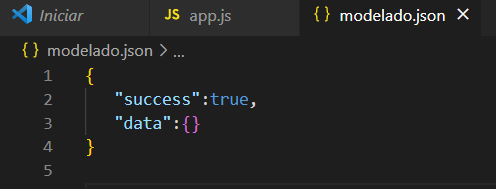
**PEDIDO:**

****

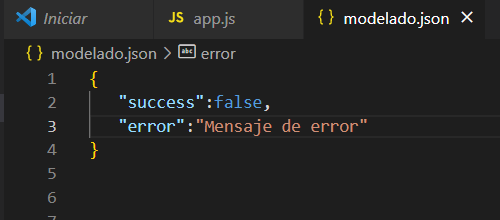
**Esquema de Respuesta:**

Respuesta de cada endpoint:

**Respuesta exitosa:**

****

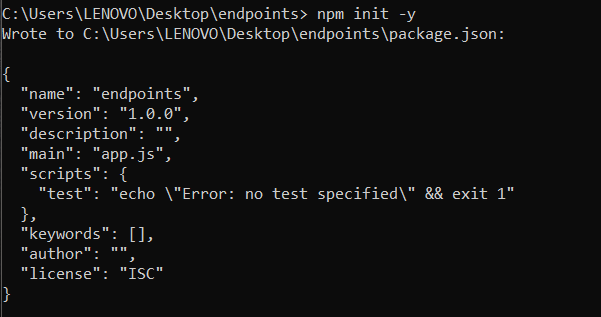
**Respuesta de error:**

****

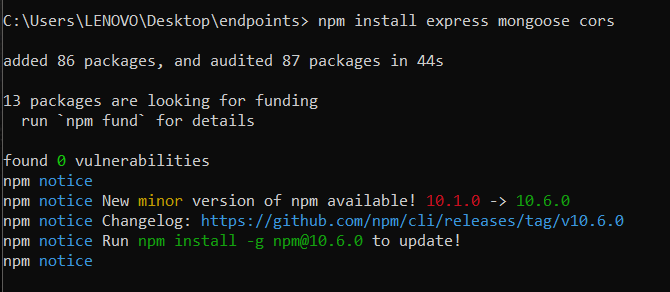
1. **CODIFICACION DE LAS APIs**

**Configuración del entorno:**

* Primero debemos tener instalado Node.js
* Creamos el nuevo proyecto de Node.js:

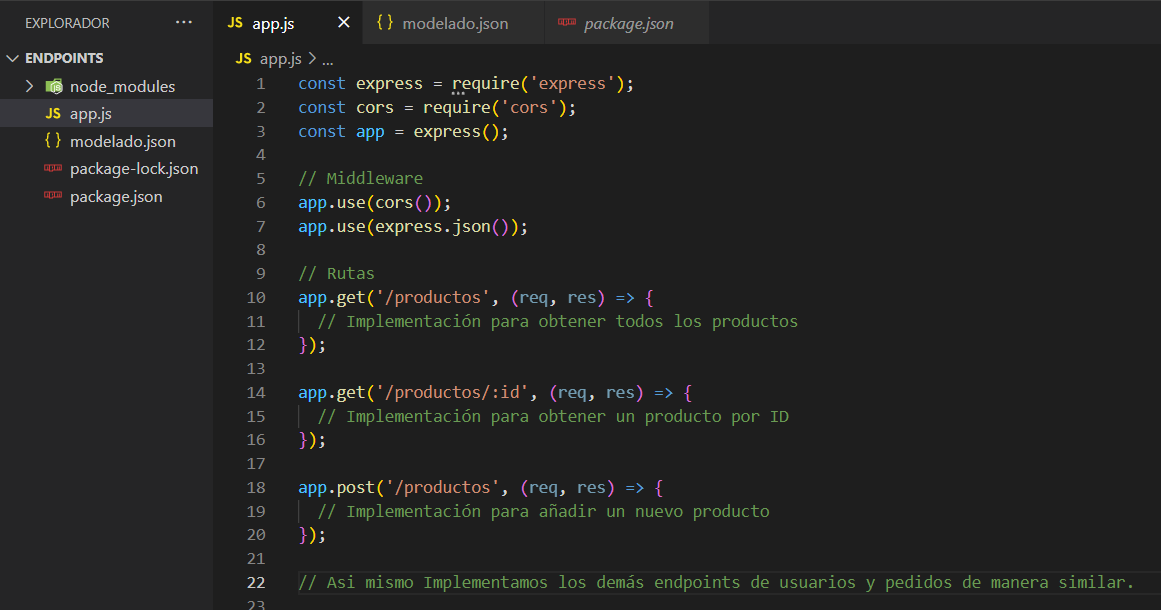
****

* Instalamos las respectivas dependencias:

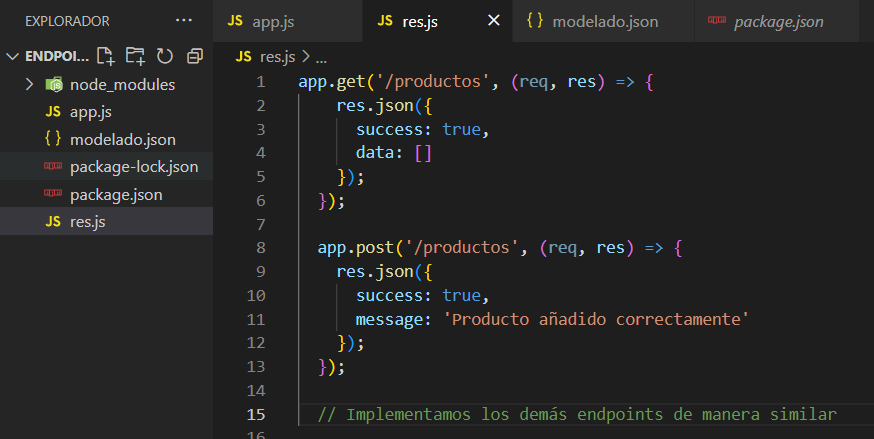
****

**Desarrollo de Endpoints**

Creamos un archivo js para empezar a definir los endpoints utilizando Express:

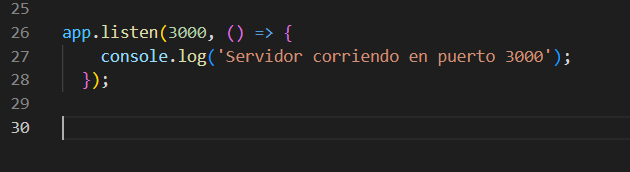
****

Implementamos los Endpoints:

****

**Pruebas locales:**

Ejecución del servidor:

****

1. **DOCUMENTACION DE ENDPOINTS**

* **Endpoint / productos:**

**Descripción:**

Este endpoint permite gestionar los productos de la tienda.

**Propósito:**

Administrar la creación, recuperación, actualización y eliminación de productos.

**Métodos permitidos:**

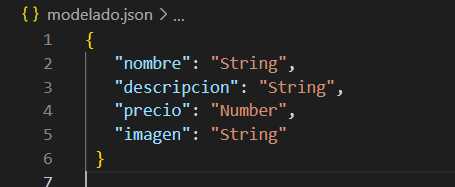
GET: obtener lista de productos

POST: Añadir un nuevo producto

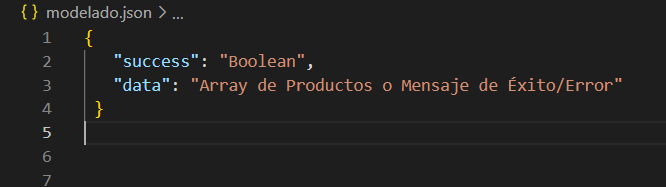
**Parámetros de entrada:**

GET: no requiere parámetros

POST:

****

**Formato de respuesta:**

****

* **Endpoint / Usuarios**

**Descripción:**

Este endpoint permite gestionar los usuarios de la tienda.

**Propósito:**

Administrar el registro, inicio de sesión, obtención y actualización de información de usuarios.

**Métodos Permitidos:**

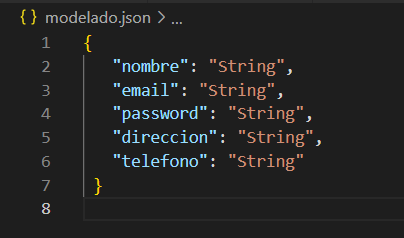
**GET:** Registrar un nuevo usuario

**GET**: Obtener información de un usuario por ID.

**PUT:** Actualizar información de un usuario por ID.

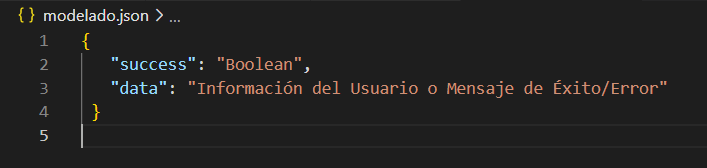
**Parámetros de entrada:**

**POST:**

****

**GET y PUT** no requieren parámetros adicionales.

**Formato de respuesta:**

****

* **Endpoint / pedidos**

**Descripción:**

Este endpoint permite gestionar los pedidos realizados en la tienda.

**Propósito:**

Administrar la creación, obtención y actualización de pedidos.

**Métodos permitidos:**

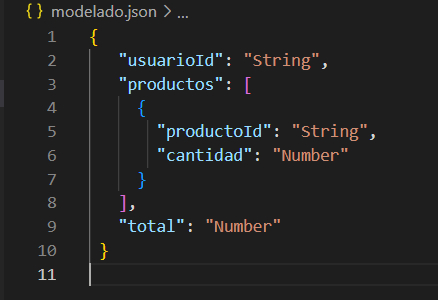
**POST:** Crear un nuevo pedido.

**GET:** Obtener detalles de un pedido por ID.

**PUT:** Actualizar estado de un pedido por ID.

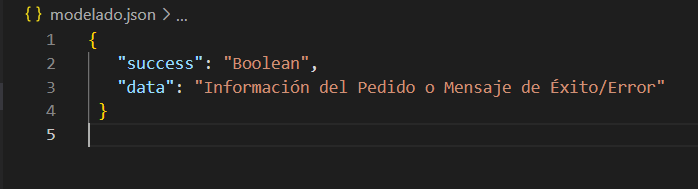
**Parámetros de entrada**

**POST:**

****

**GET y PUT:** No requieren parámetros adicionales.

**Formato de respuesta:**

****

1. **VERSIONAIENTO DEL PROYECTO.**

Inicialización del repositorio GIT

Creamos un nuevo repositorio y subimos el proyecto.

Link del repositorio:

<https://github.com/stivenrpzx/Endpoints.git>

**CONCLUSION**

En este taller hemos aprendido a diseñar y desarrollar APIs para una tienda online utilizando JavaScript como lenguaje de programación principal. Hemos abordado cada paso, desde la definición de requisitos hasta el versionamiento y organización.

Se han diseñado endpoints claros y funcionales para gestionar productos, usuarios y pedidos, y los hemos implementado utilizando Express.js y MongoDD. Además, hemos documentado detalladamente cada uno de los endpoints.