Descripción del código API – Gestión de Productos (Node.js + Express)

Este código implementa una **API REST básica** para gestionar productos utilizando **Node.js** y el framework **Express**. La API permite realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre una lista simulada de productos.

⚙️ 1. Configuración inicial

```
const express = require("express");

const app = express();

const PORT = 3000;

app.use(express.json());
```

- Se importa Express.
- Se crea una aplicación Express (app).
- Se define el puerto (3000).
- Se habilita el uso de JSON en las solicitudes (express.json()).

🗄️ 2. Simulación de base de datos

```
let productos = [

  { id: 1, nombre: "Arroz", precio: 2500, stock: 50 },

  { id: 2, nombre: "Aceite", precio: 5800, stock: 20 }

];
```

El sistema usa un **arreglo en memoria** como base de datos temporal para almacenar productos.
Cada producto tiene:

- id
- nombre
- precio
- stock

📌 3. Endpoints de la API

✅ GET /api/productos – Listar productos

```
app.get("/api/productos", (req, res) => {

  res.status(200).json(productos);
```

});

- Devuelve la lista completa de productos.
- Respuesta: código **200** + JSON.

🆕 **POST /api/productos – Crear producto**

```
app.post("/api/productos", (req, res) => {
  const nuevo = {
    id: productos.length + 1,
    nombre: req.body.nombre,
    precio: req.body.precio,
    stock: req.body.stock
  };
  productos.push(nuevo);
  res.status(201).json({ message: "Producto creado", producto: nuevo });
});
```

- Recibe los datos desde req.body.
- Crea un nuevo objeto producto.
- Lo agrega al arreglo.
- Respuesta: código **201** + mensaje + producto creado.

✏️ **PUT /api/productos/:id – Actualizar producto**

app.put("/api/productos/:id", (req, res) => {

- Busca un producto por id.
- Si no existe → responde con **404**.
- Si existe, actualiza solo los campos recibidos en el body.
- Respuesta: **200** + producto actualizado.

🗑️ **DELETE /api/productos/:id – Eliminar producto**

```
app.delete("/api/productos/:id", (req, res) => {
  productos = productos.filter(p => p.id !== id);
  res.status(200).json({ message: "Producto eliminado" });
```

});

- Filtra el arreglo para eliminar el producto con el id indicado.
- Respuesta: **200** + mensaje de confirmación.

🚀 4. Inicialización del servidor

```
app.listen(PORT, () => {
  console.log(`Servidor corriendo en http://localhost:${PORT}`);
});
```

- Arranca el servidor en el puerto definido.
- Muestra mensaje en consola al iniciar.

🎯 Resumen general

Este código construye una API REST funcional que permite:

Operación	Método	Endpoint	Descripción
Leer	GET	/api/productos	Lista todos los productos
Crear	POST	/api/productos	Agrega un nuevo producto
Actualizar	PUT	/api/productos/:id	Modifica datos de un producto
Eliminar	DELETE	/api/productos/:id	Borra un producto por id

Es ideal para prácticas de CRUD, testing en Postman y proyectos formativos del SENA.