Conceptos clave de JWT:

- **Header:** Contiene el tipo de token (JWT) y el algoritmo de cifrado (ej., HS256).
- **Payload:** Contiene las "claims" o afirmaciones sobre la entidad (generalmente el usuario) y datos adicionales. Aquí podrías incluir el ID del usuario, su rol (admin, cliente, etc.) y la fecha de expiración del token.
- **Signature:** Se crea combinando el header, el payload, y una "clave secreta" (secret key) del servidor. Esto asegura la integridad del token y que no ha sido alterado.

Pasos para implementar JWT en tu API de productos:

1. Instalar las dependencias necesarias:

   Necesitarás express (si aún no lo tienes), jsonwebtoken para manejar los JWT y dotenv para gestionar tus variables de entorno de forma segura.

   ```
   Bash
   npm install express jsonwebtoken dotenv
   ```

2. Configurar variables de entorno (.env):

   Es crucial que tu clave secreta JWT no esté directamente en tu código. Crea un archivo .env en la raíz de tu proyecto:

   ```
   JWT_SECRET=tu_clave_secreta_super_segura_aqui
   ```

   Y carga estas variables al inicio de tu aplicación:

   ```
   JavaScript
   // server.js o app.js
   require('dotenv').config();
   const express = require('express');
   const jwt = require('jsonwebtoken');
   const app = express();

   app.use(express.json()); // Para parsear el body de las peticiones JSON
   ```

3. Implementar el registro y/o login de usuarios:

   Antes de poder proteger tus rutas de productos, necesitas una forma para que los usuarios se registren y/o inicien sesión y obtengan un token.

   - **Registro (ejemplo simplificado, sin base de datos):**

     ```
     JavaScript
     // Esto es solo un ejemplo. En un caso real, guardarías usuarios en una BD.
     const users = []; // Array para simular una base de datos de usuarios

     app.post('/register', (req, res) => {
         const { username, password } = req.body;
         // Aquí deberías hashear la contraseña antes de guardarla
     ```

```
    // Por simplicidad, no lo haremos en este ejemplo
    users.push({ username, password });
    res.status(201).send('Usuario registrado exitosamente');
});
```

- Login y generación de JWT:

  Cuando un usuario inicia sesión correctamente, le generas un token.

  JavaScript

```
app.post('/login', (req, res) => {
    const { username, password } = req.body;

    // En un caso real, verificarías las credenciales contra tu base de datos
    const user = users.find(u => u.username === username && u.password
=== password);

    if (!user) {
        return res.status(401).send('Credenciales inválidas');
    }

    // Generar el token JWT
    const accessToken = jwt.sign(
        { username: user.username, role: 'user' }, // Payload: información que
quieres guardar en el token
        process.env.JWT_SECRET,                    // Clave secreta
        { expiresIn: '1h' }                        // Opciones: expira en 1 hora
    );

    res.json({ accessToken });
});
```

4. Crear un middleware para verificar el JWT:

   Este middleware se ejecutará antes de tus rutas protegidas de productos para validar el token.

   JavaScript

```
function authenticateToken(req, res, next) {
    const authHeader = req.headers['authorization'];
    const token = authHeader && authHeader.split(
```