**Universidad Tecnológica de** **Ciudad Juárez**

**Materia: DESARROLLO WEB INTEGRAL**

**Unidad l V**

**Grupo: IDSM31**

**Docente: IBARRA MUÑOZ ELIXENA**

**Nombre:**

Soto Manzano Juan Ramon

**28/08/2025**

**Descripción general del proyecto**

Este proyecto corresponde a la continuación de la Unidad III, donde se desarrolló un carrito de compras con autenticación híbrida:

* **JWT (local)** y **OAuth (Google)** para proteger endpoints.
* **Frontend y Backend** implementados con Node.js, Express y HTML/CSS/JS.

En la Unidad IV, la aplicación se **contenerizó utilizando Docker y Docker Compose**, con configuración de variables de entorno y pruebas completas de su funcionamiento.

**Arquitectura y Tecnologías**

**Tecnologías utilizadas:**

* **Backend y Frontend:** Node.js con Express.
* **Autenticación:** JWT y Google OAuth.
* **Contenerización:** Docker y Docker Compose.
* **Variables de entorno:** archivo .env para claves y puertos.

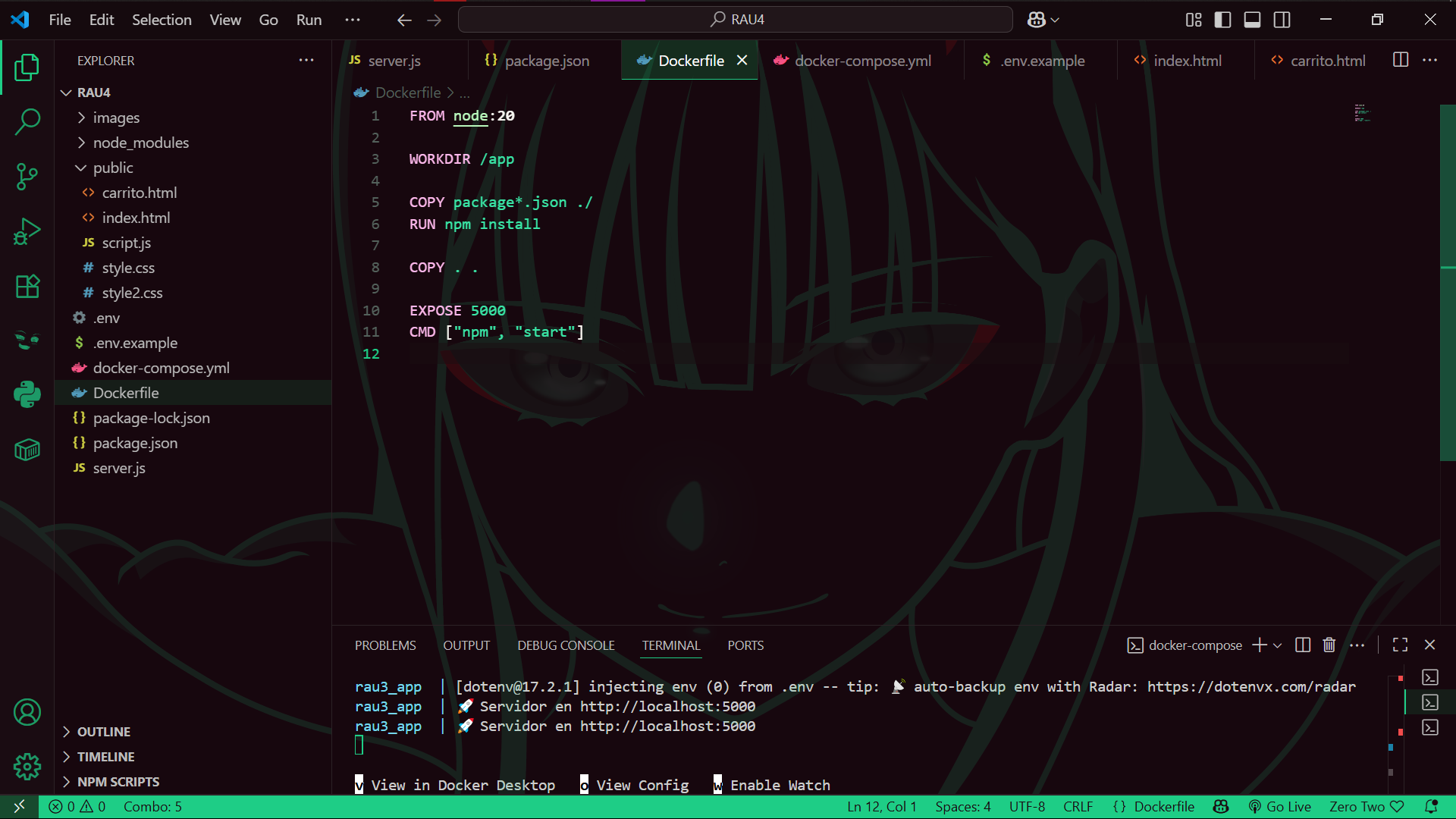
**Flujo general actualizado:**

1. La aplicación se ejecuta dentro de un contenedor Docker.
2. Las variables sensibles (SECRET\_KEY, GOOGLE\_CLIENT\_ID, PORT) se configuran desde .env.
3. El contenedor expone el puerto 5000 para acceso desde navegador y Postman.

**Implementación de Docker**

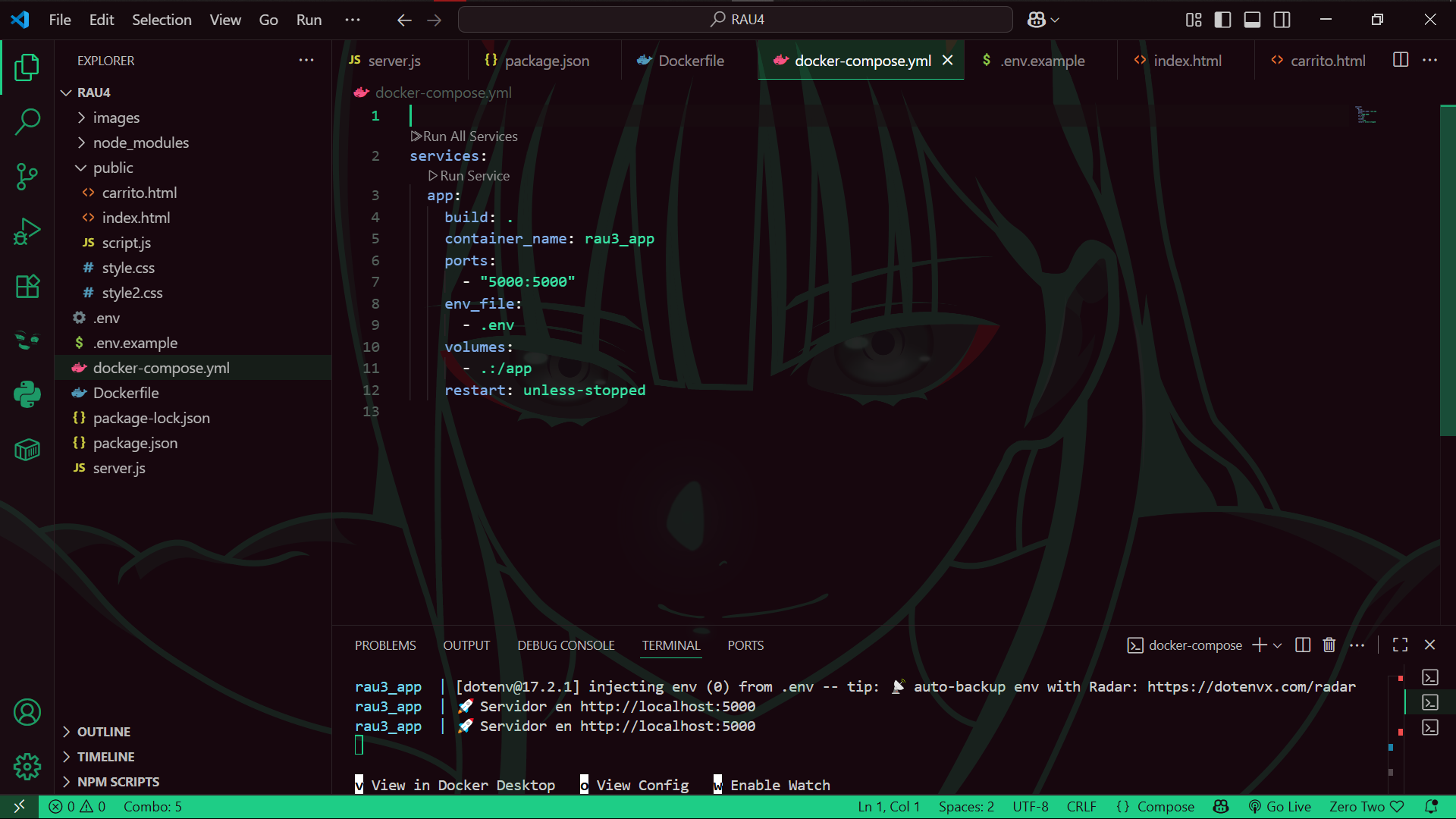
**3.1 Creación del Dockerfile**

* Se definió un contenedor basado en Node.js.
* Se copiaron los archivos del proyecto (server.js, frontend y dependencias).
* Se instalaron dependencias con npm install.
* Se expuso el puerto 3000.
* Se configuró el comando npm start para iniciar el servidor.



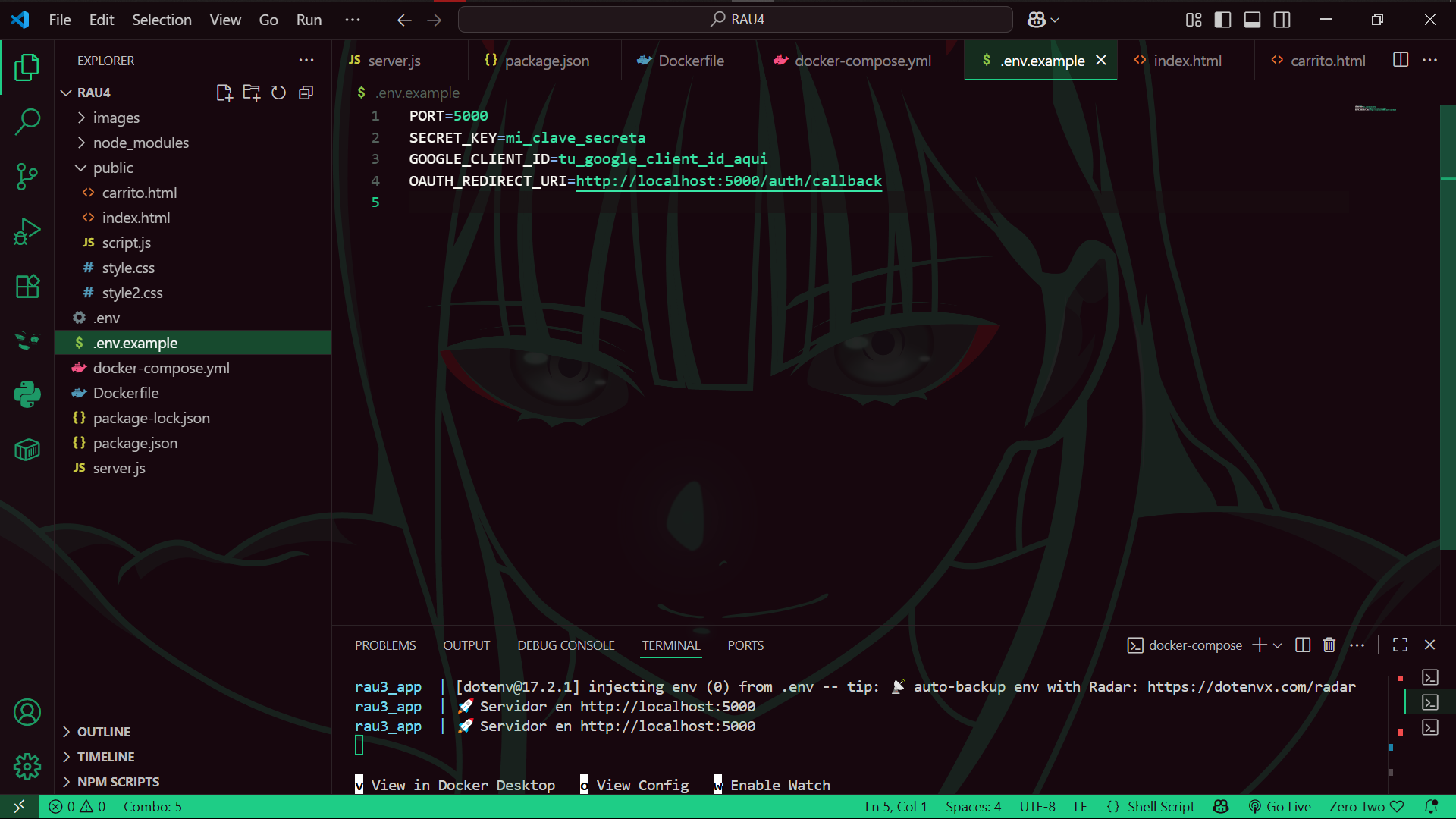
**Creación de docker-compose.yml**

* Se definió un servicio llamado app.
* Se configuró el mapeo de puertos (5000:5000).
* Se montó el proyecto como volumen para desarrollo en caliente (opcional).
* Se cargaron variables desde .env.



**Configuración de Variables de Entorno (.env)**

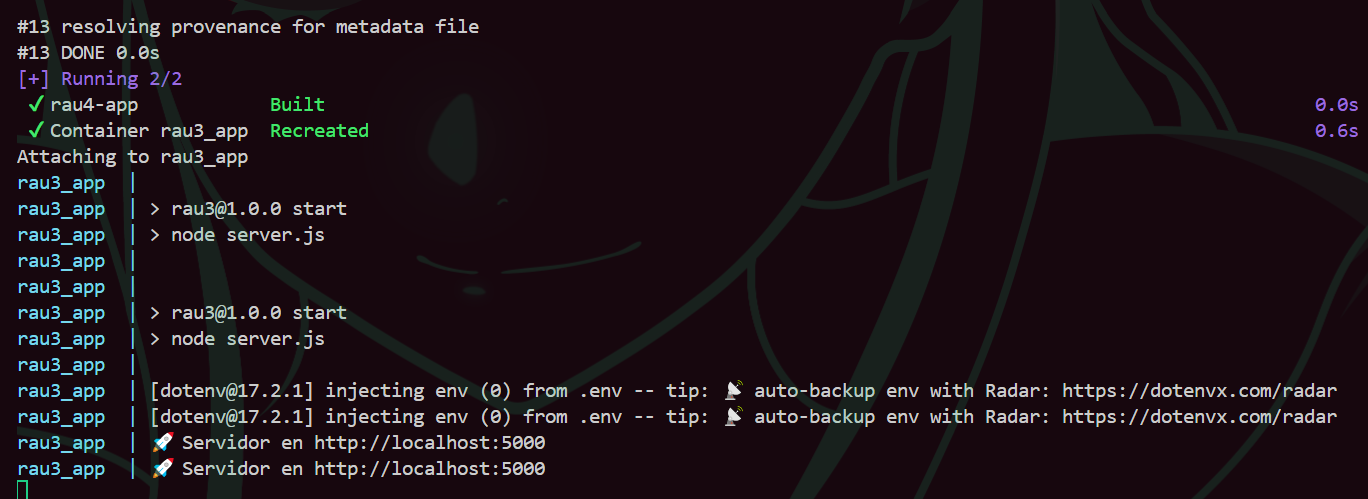
* Se creó un archivo .env con las variables:
  + PORT=5000
  + SECRET\_KEY=<clave\_secreta>
  + GOOGLE\_CLIENT\_ID=<tu\_client\_id\_de\_google>
* Se creó un .env.example sin valores sensibles para documentación.



**Ejecución y pruebas del contenedor**

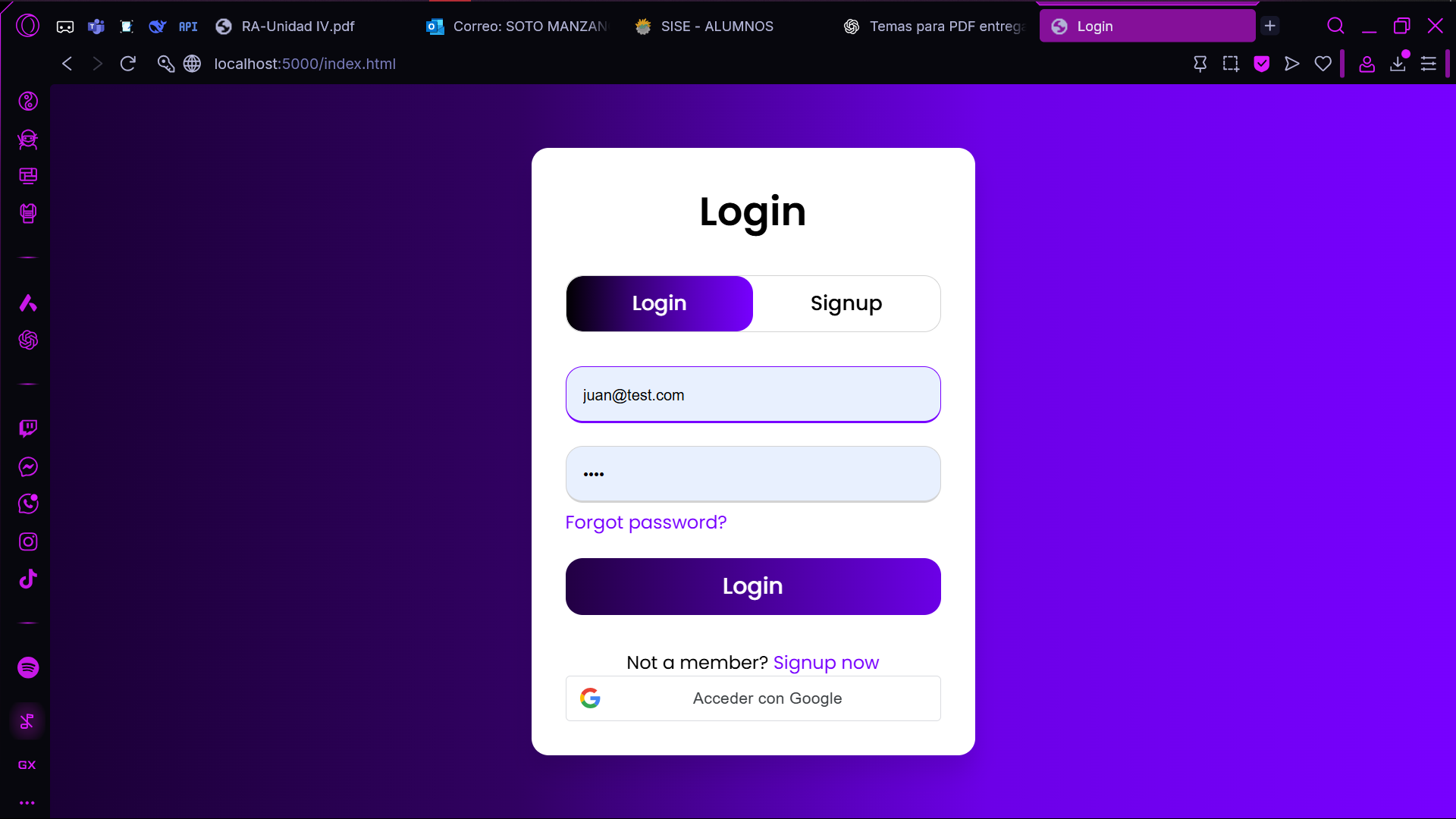
**4.1 Levantar el contenedor**

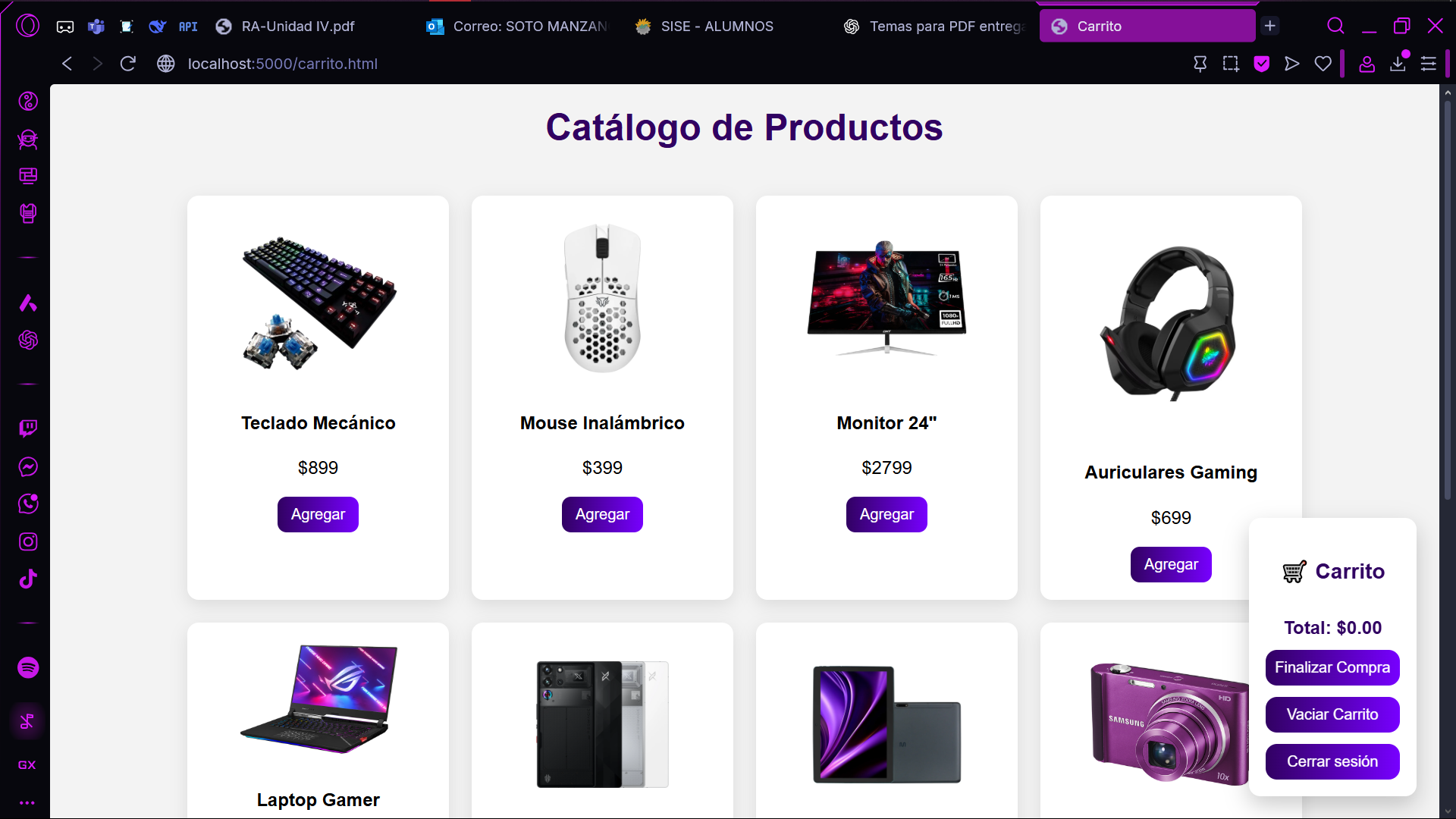
* Se ejecutó el comando:
* docker-compose up --build
* Se verificó que la aplicación se construyera sin errores.
* Se comprobó que el contenedor quedara corriendo correctamente.



**Verificar aplicación en navegador**

* Se accedió a <http://localhost:5000>.
* Se cargó la página de login y el catálogo de productos en carrito.html.



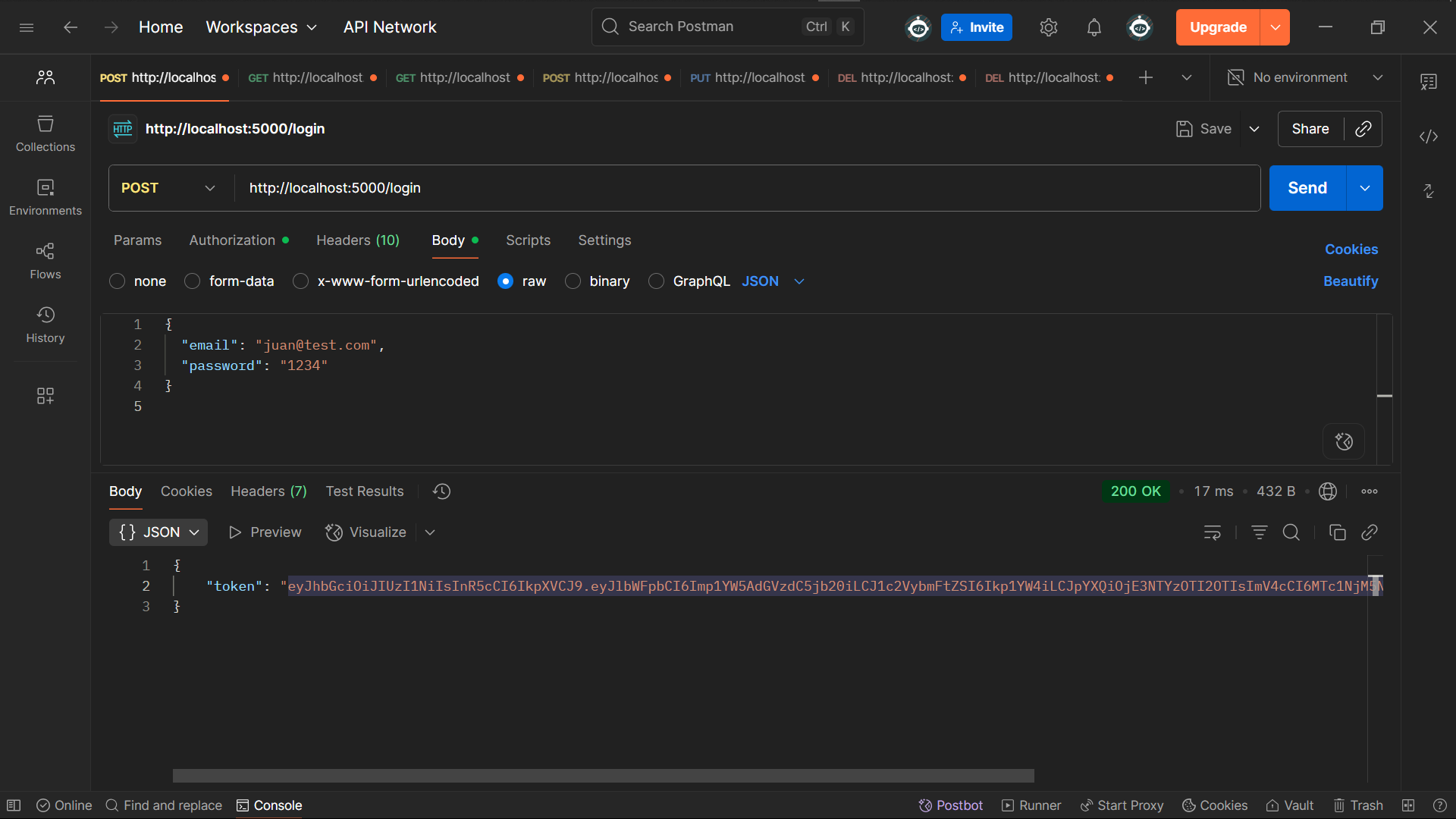


Pruebas de autenticación y funcionamiento

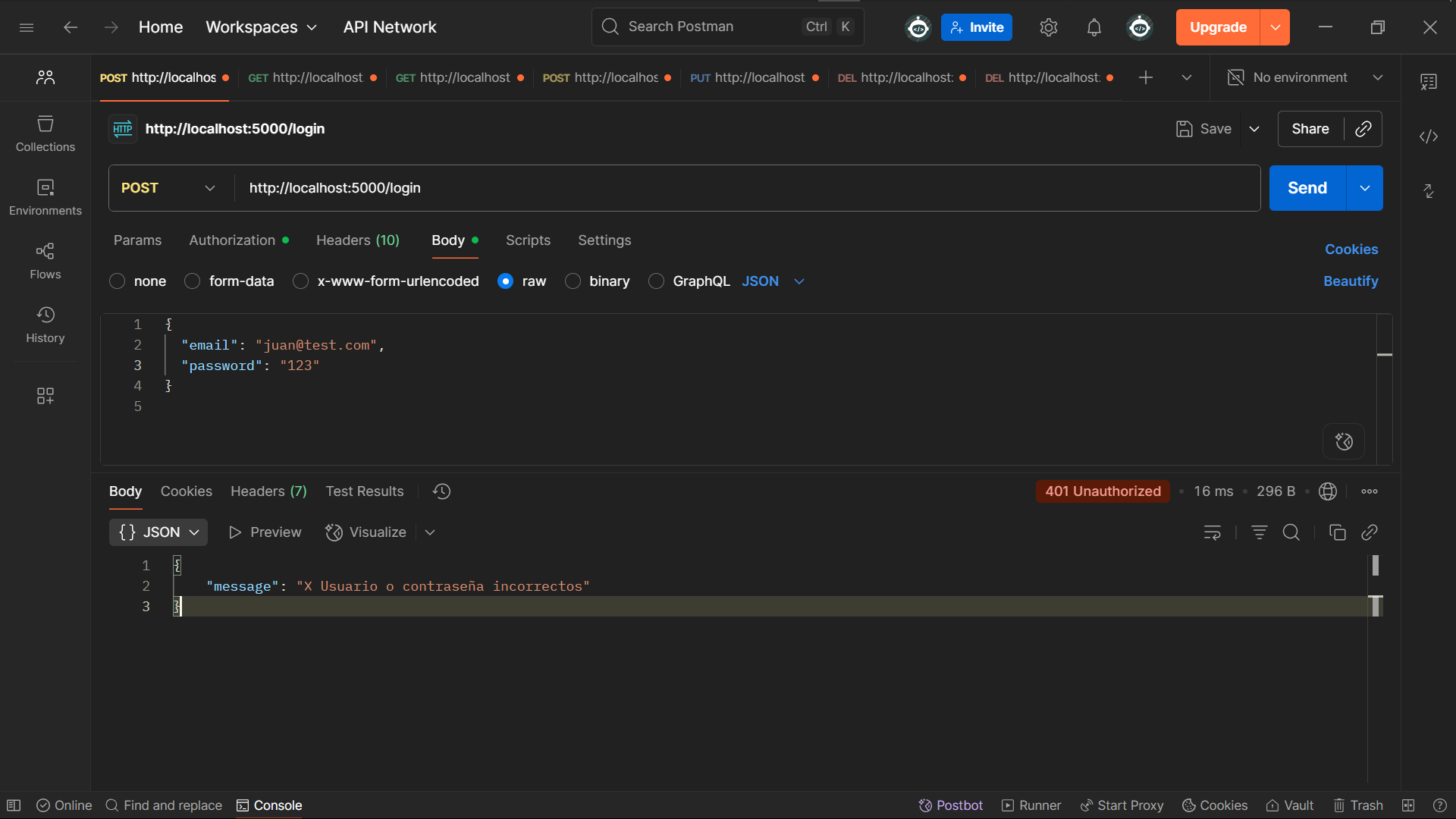
**Pruebas con Postman**

Se realizaron pruebas tanto con JWT como con Google OAuth.

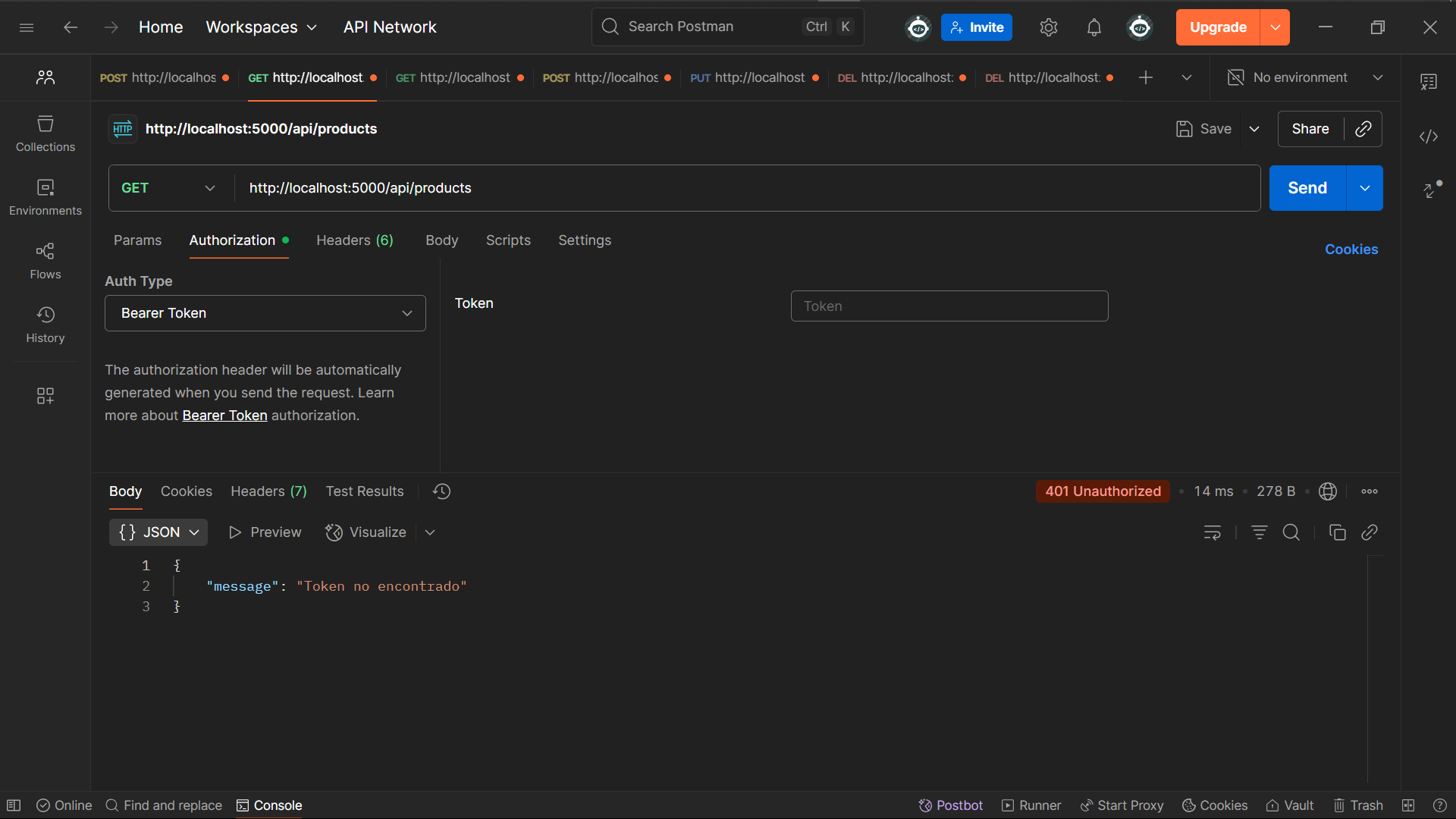
* **Login con credenciales correctas:** retorna token válido.



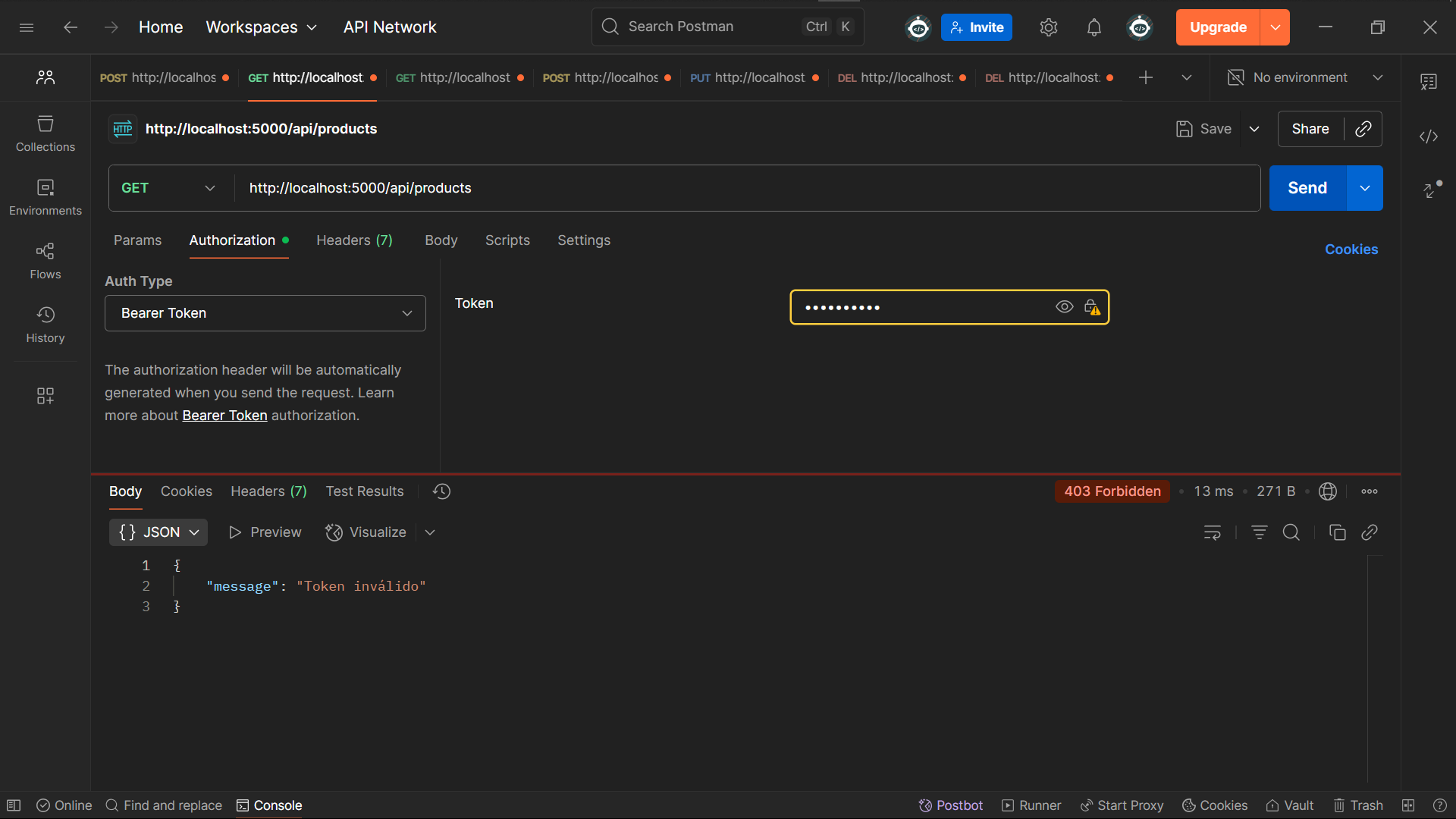
* **Login con credenciales incorrectas:** retorna 401 Unauthorized.

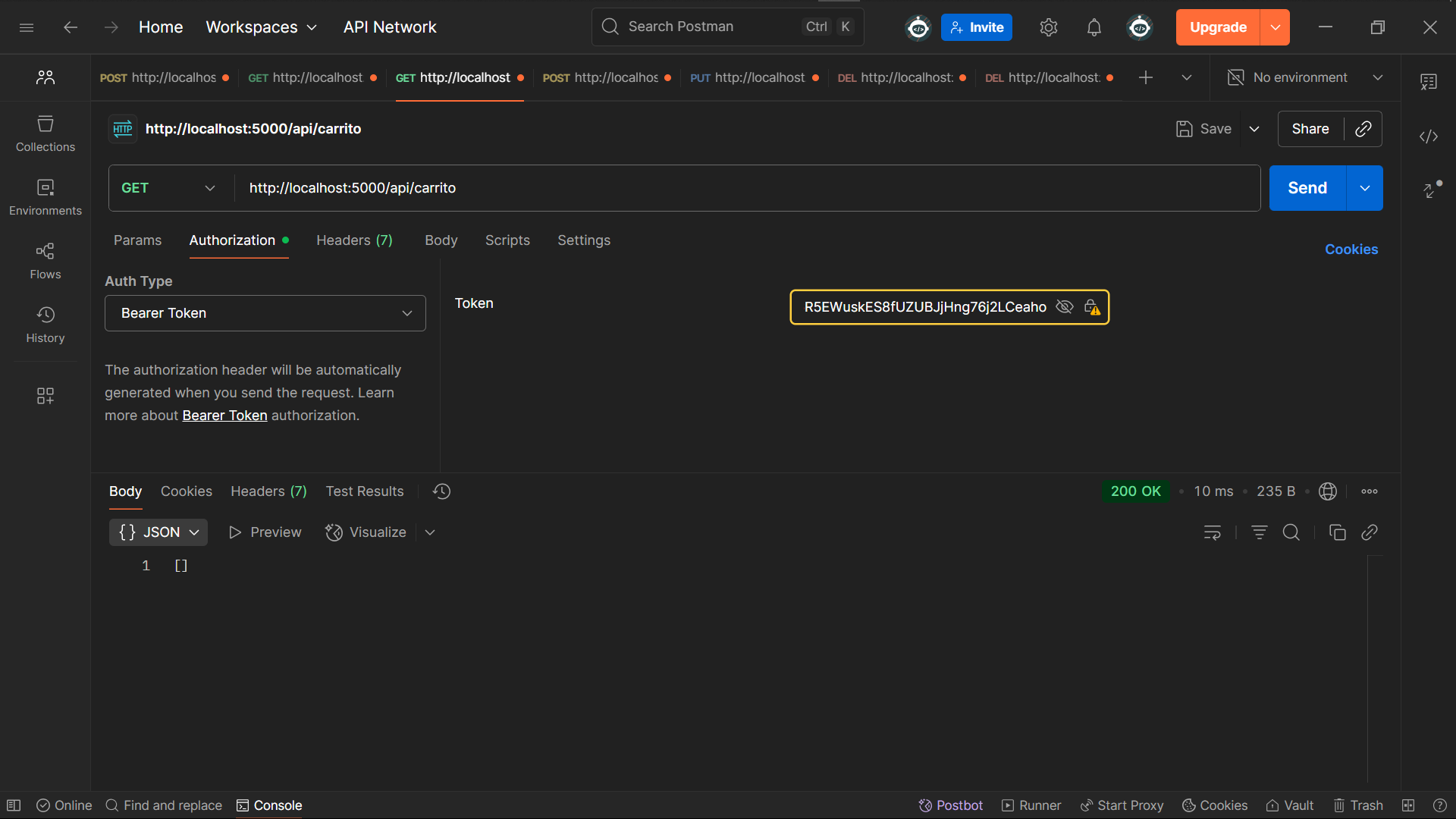
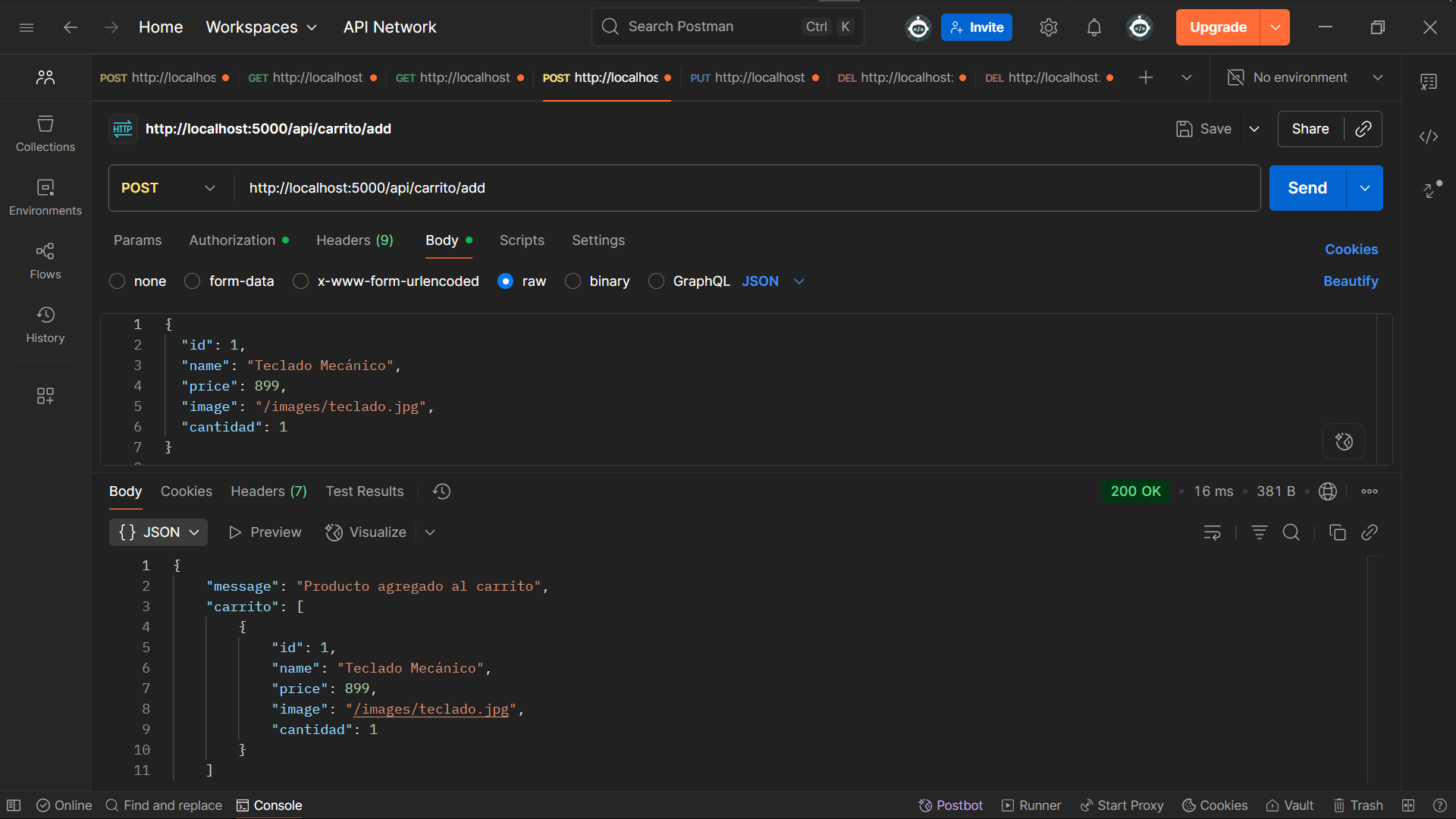
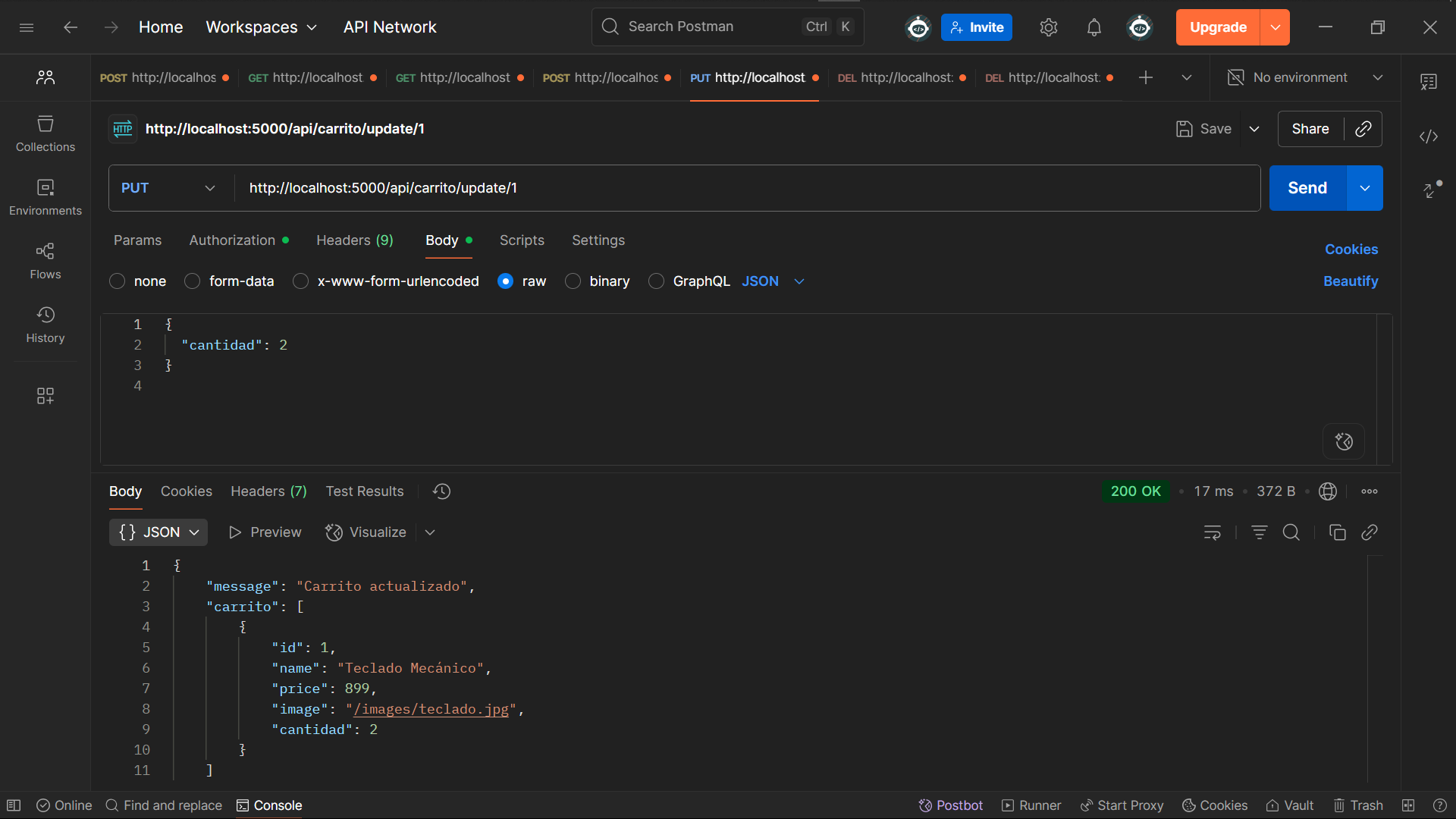
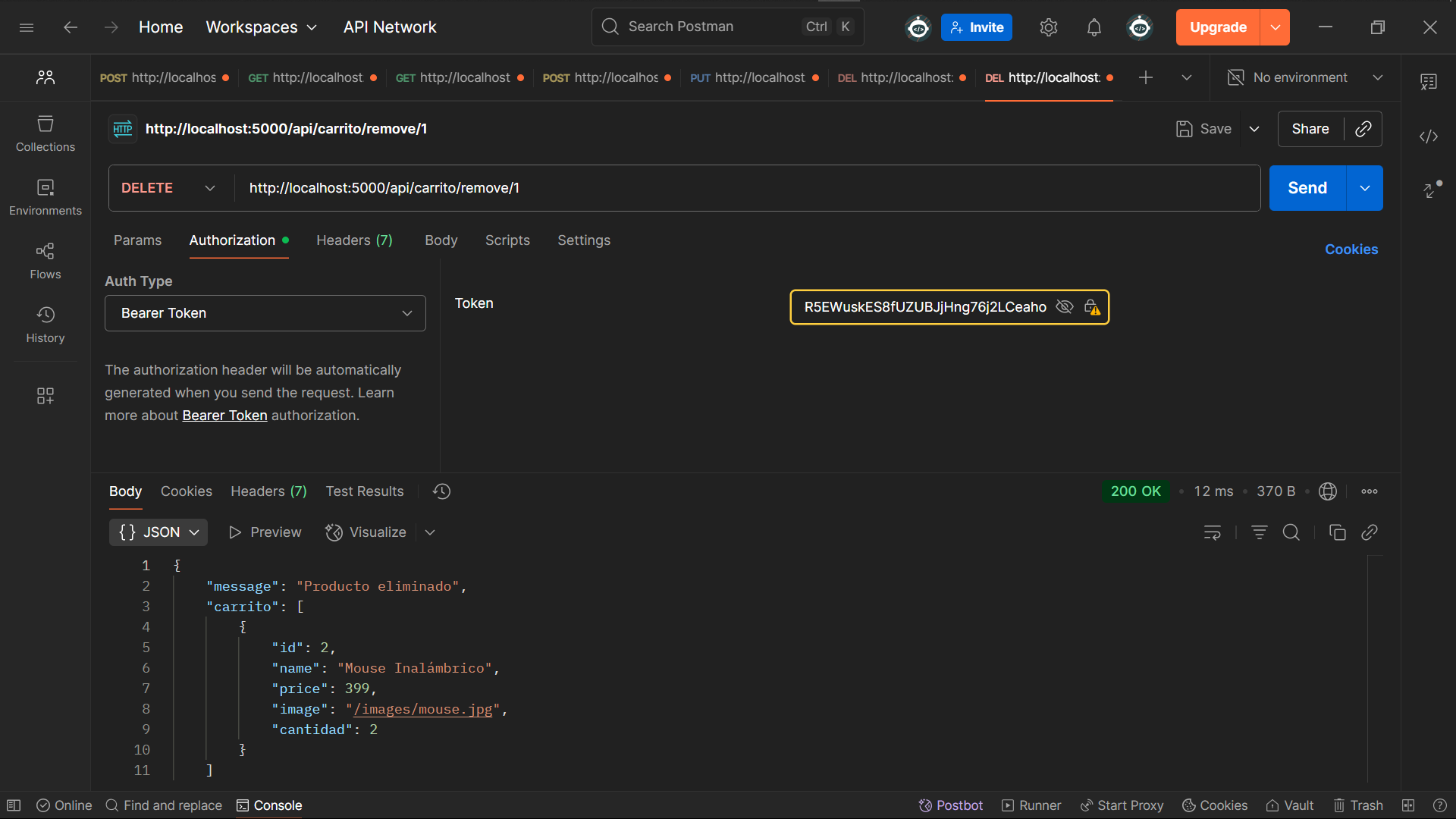
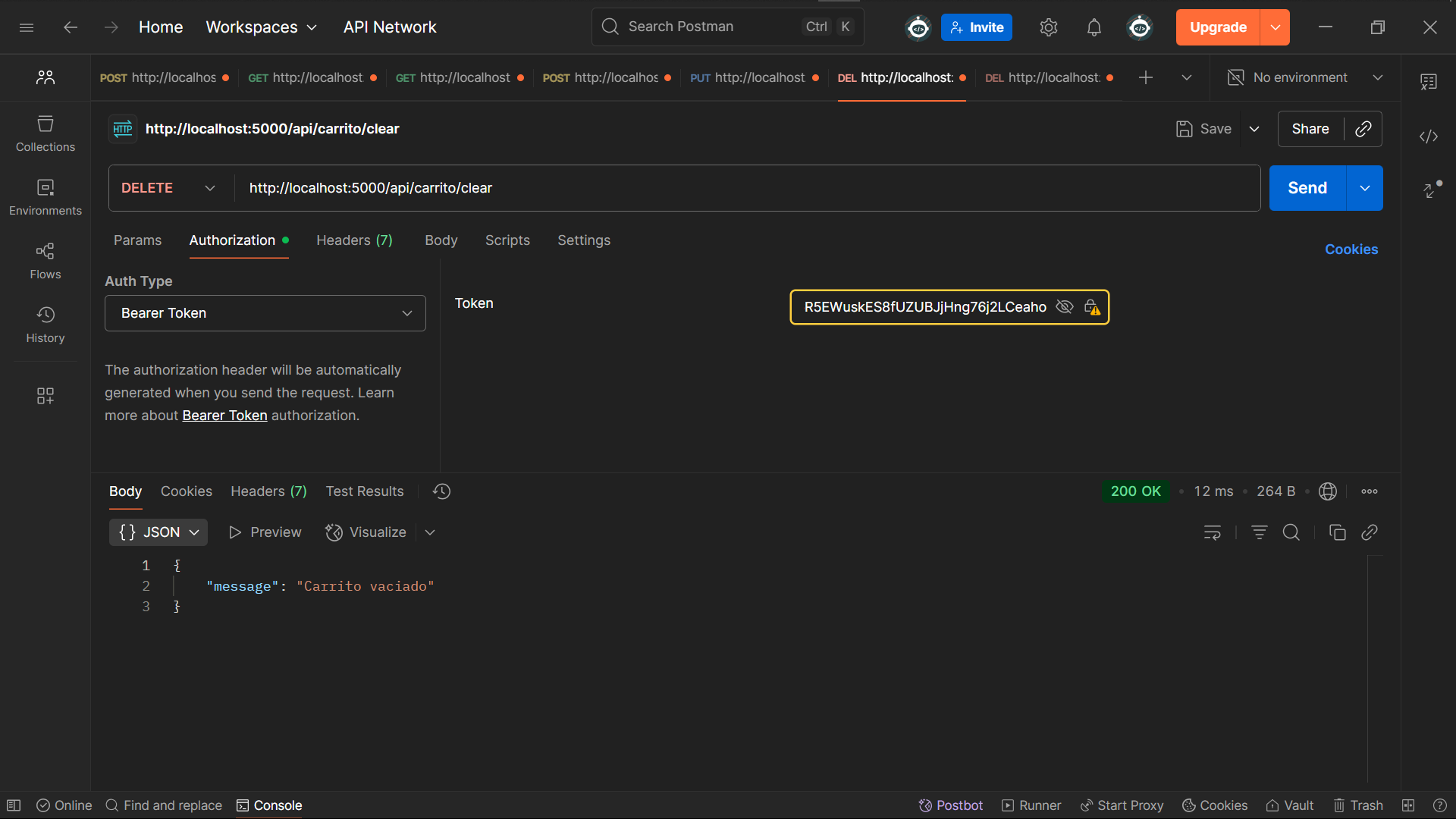
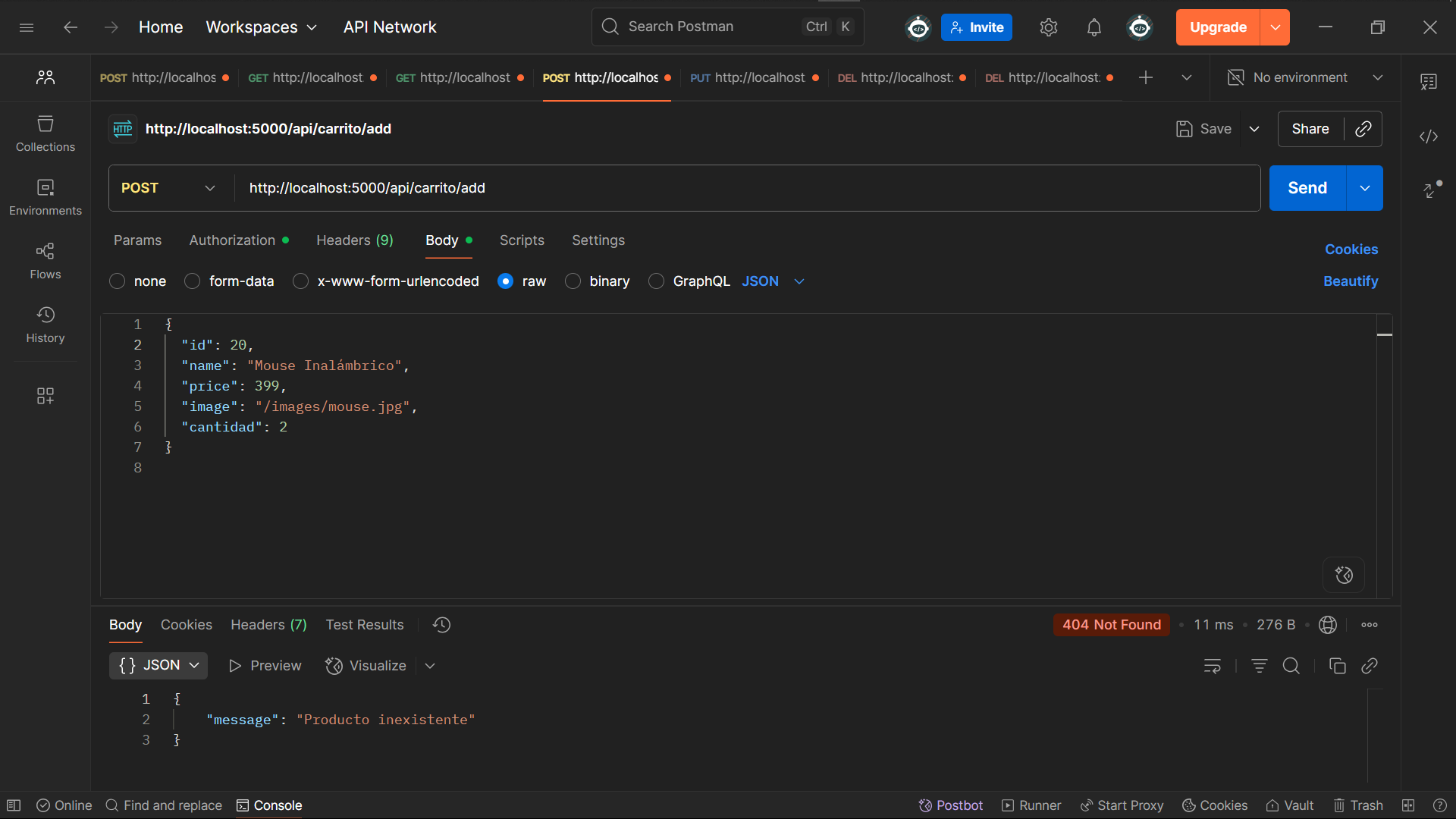
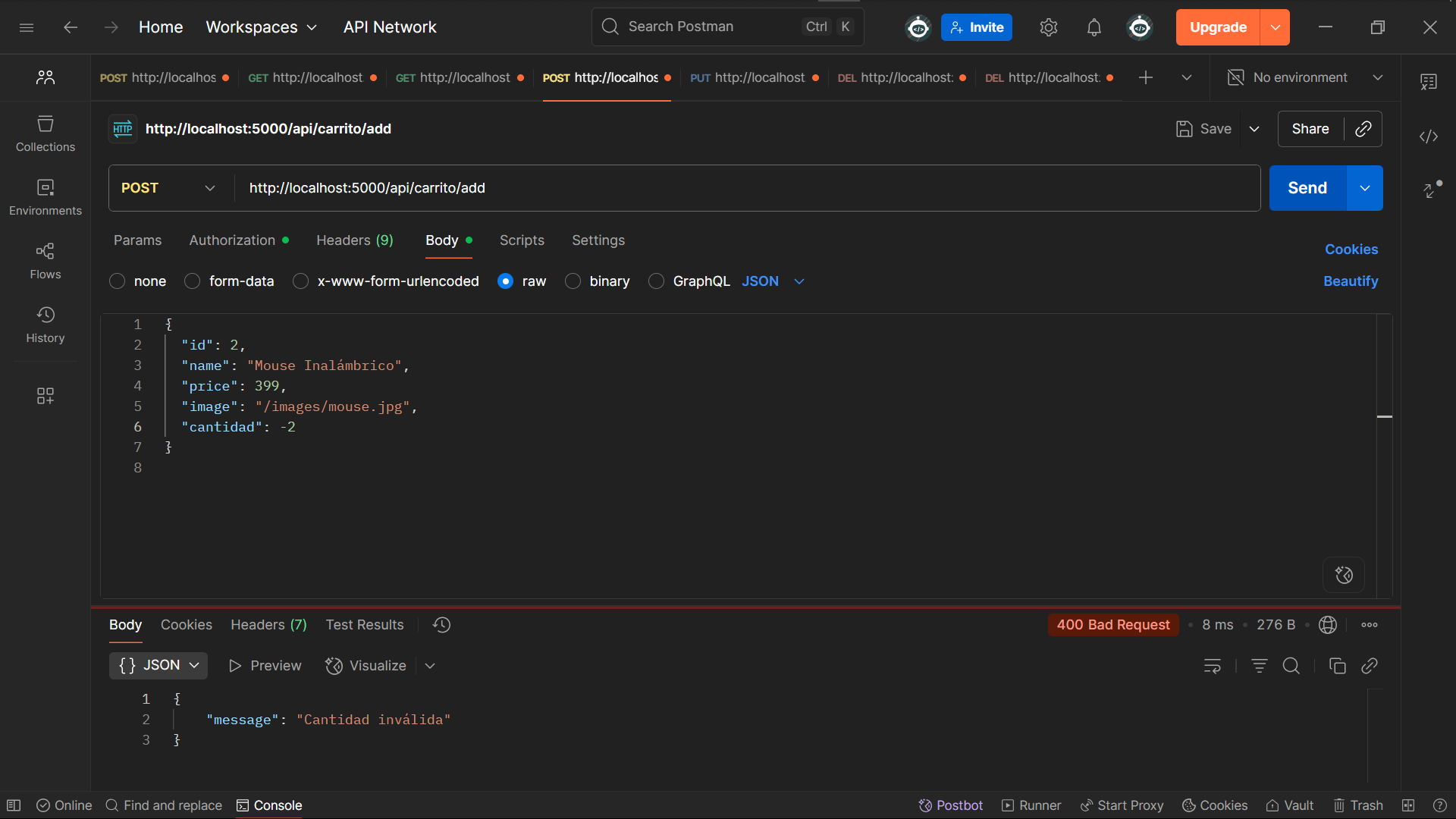
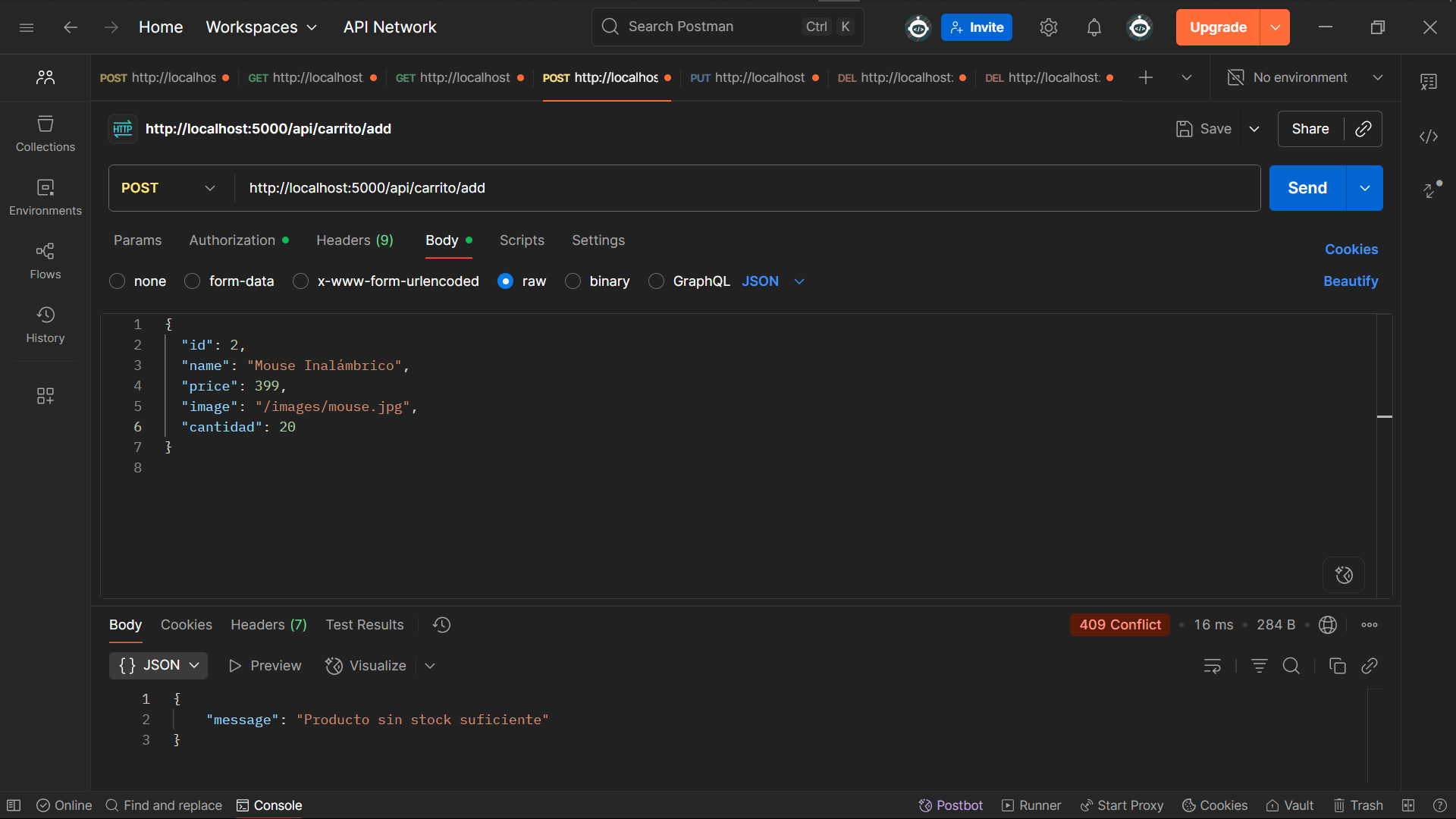


* **Acceso sin token:** retorna 401 Unauthorized.

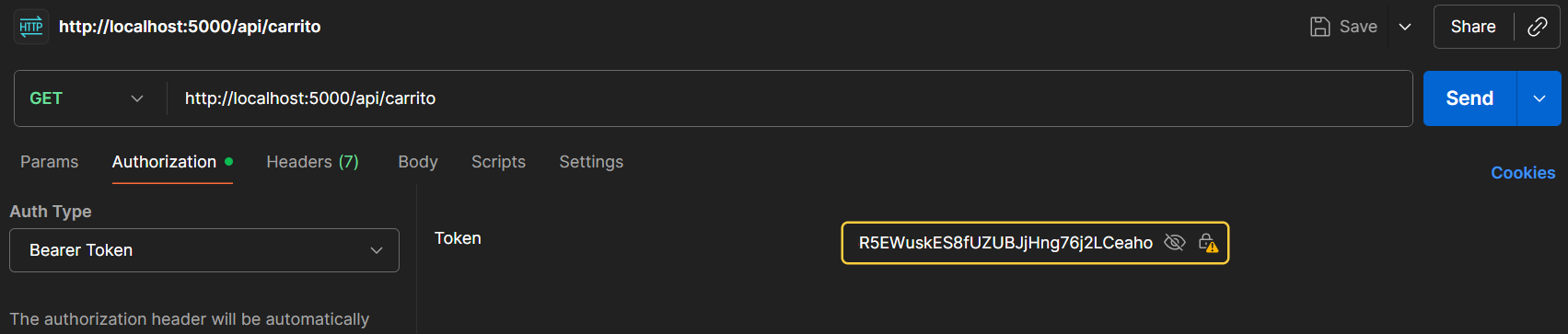


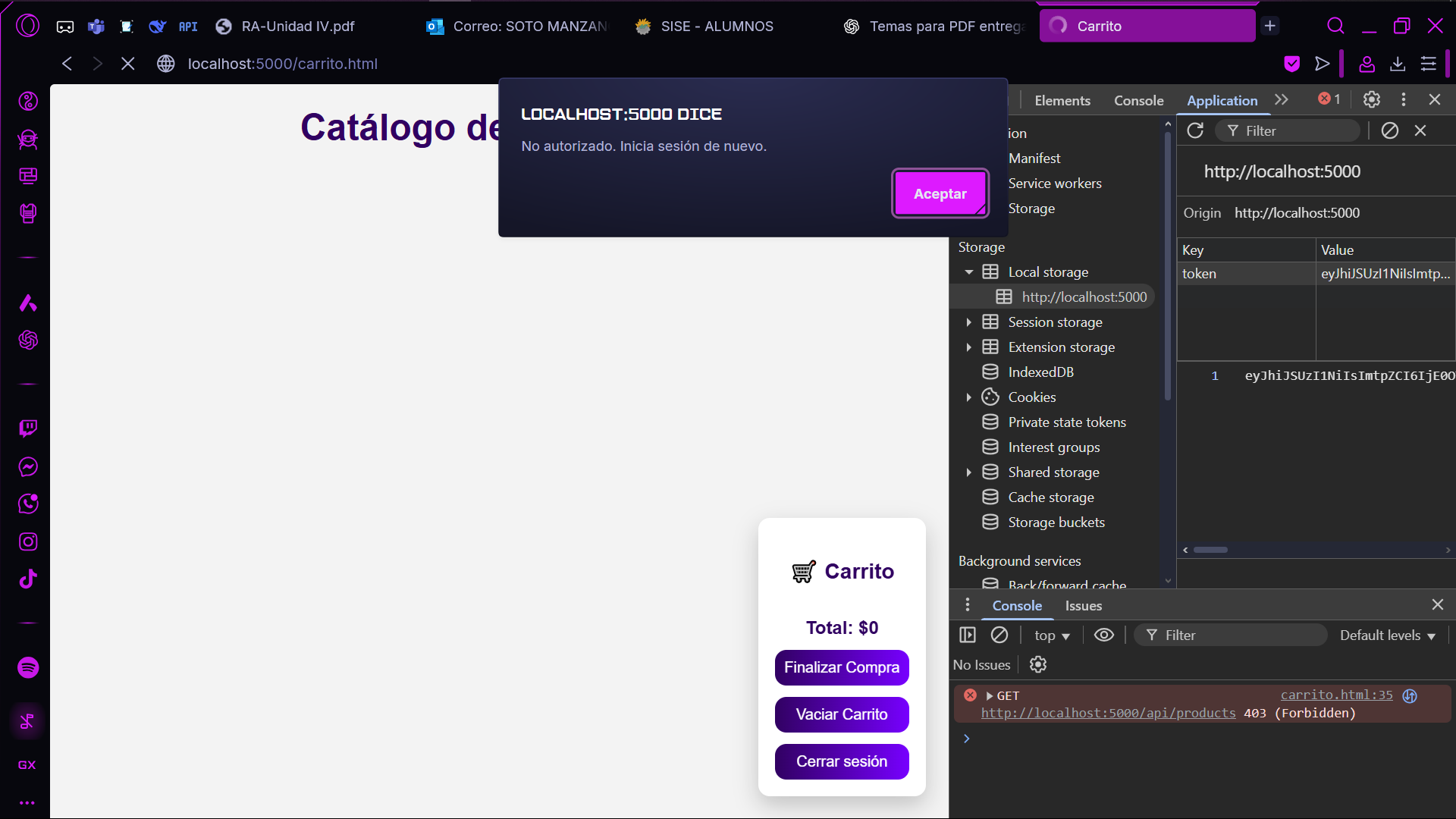
* **Acceso con token inválido:** retorna 403.



* **Flujo del carrito:**
  + Consultar carrito vacío → GET /api/carrito.
  + 
  + Agregar producto → POST /api/carrito/add.
  + 
  + Modificar cantidad → PUT /api/carrito/update/:id.
  + 
  + Eliminar producto → DELETE /api/carrito/remove/:id.
  + 
  + 
  + Vaciar carrito → DELETE /api/carrito/clear.
  + 
* **Casos de error:**
  + Agregar producto inexistente → 404.
  + 
  + Cantidad negativa → 400.
  + 
  + Stock insuficiente → 409.
  + 

**Manejo y Almacenamiento de Tokens**

* El token JWT o el ID Token de Google se guarda en localStorage del navegador.
* 
* Se envía en la cabecera Authorization: Bearer <token> para todas las peticiones.
* 
* Si el token no existe o es inválido, el sistema redirige a la página de login.

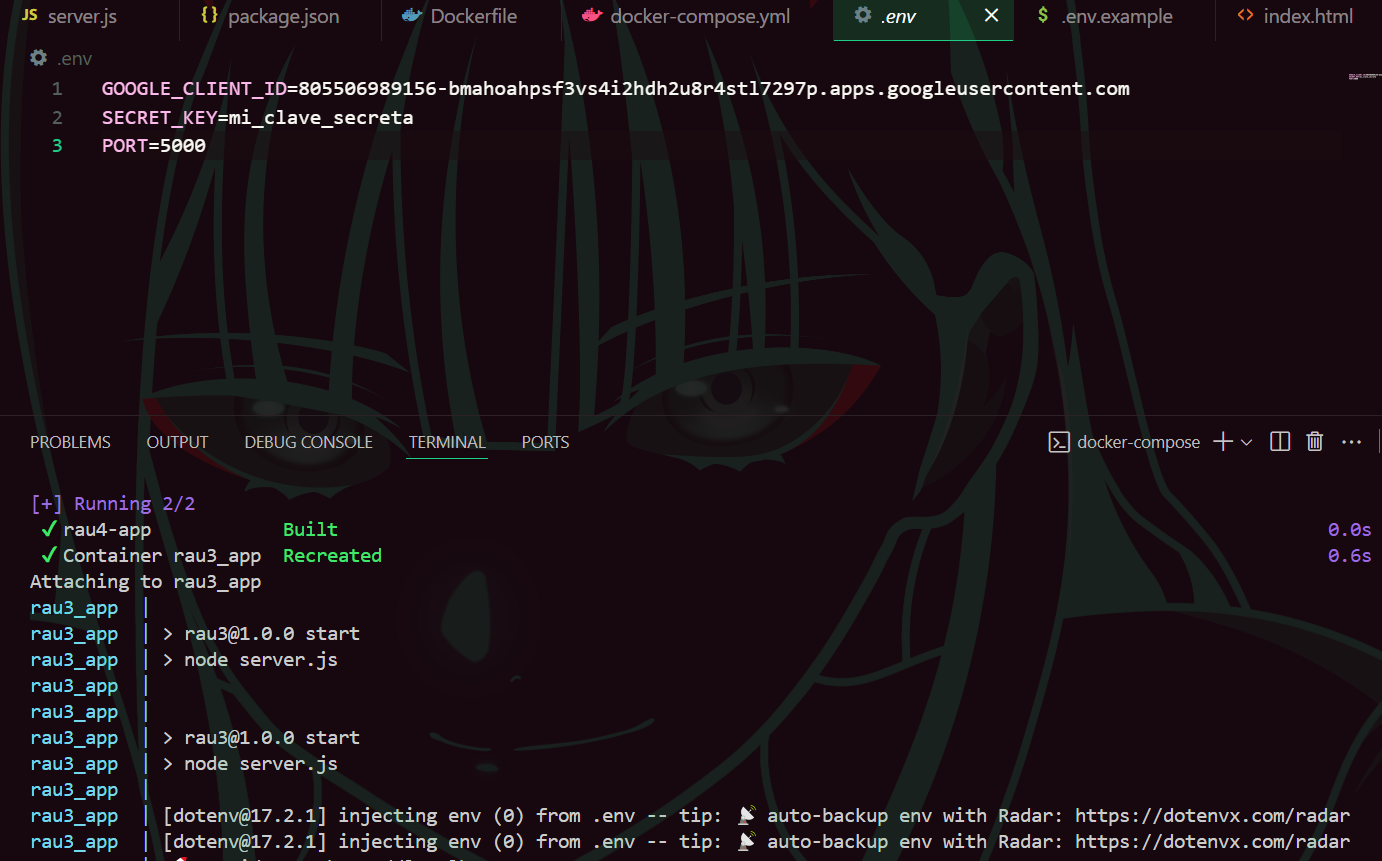


**Documentación y repositorio**

* **Código del contenedor:**
  + Dockerfile
  + docker-compose.yml
  + .env y .env.example
* **Archivo README.md:**
  + Pasos para construir y ejecutar el contenedor.
  + Explicación breve de variables de entorno.
  + Instrucciones de pruebas con navegador y Postman.

**Configuración de OAuth y JWT en el contenedor**

* **JWT:**
  + Se mantuvo la generación de tokens con jsonwebtoken.
  + La clave secreta se cargó desde el archivo .env.
* **OAuth (Google):**
  + El Client ID se leyó desde la variable GOOGLE\_CLIENT\_ID.
  + Se verificaron los tokens dentro del contenedor usando la librería oficial.



**Conclusiones**

* Se logró contenerizar correctamente la aplicación web del carrito con JWT y OAuth.
* El uso de Docker facilitó la portabilidad y la gestión de variables sensibles.
* Se realizaron pruebas completas con navegador y Postman sin errores.