Evaluación

Requerimiento (incluye validaciones, regresar código de error y que funcione correctamente).	Valor	Realizado (Si/No)
1. GET /products - Filtro por parámetros de búsqueda.	15	
2. POST /products/cart	15	
3. GET /products/:id	10	
4. Middleware de autenticar	10	
5. POST /admin/products	10	
6. PUT /admin/products/:id	15	
7. DELETE /admin/products:id	10	
8. GET /, /home, /shopping_cart	10	
9. Código estructurado y modularizado - Archivo de router global - Archivo de router para productos - Archivo de router para endpoints de administrador de productos	10	
10. Manejo de archivos	10	

Conclusiones

La práctica se trató de implementar una API RESTful para un sistema e-commerce, usando rutas, middlewares, validaciones, persistencia en archivos y demás, la modulación de controladores y rutas fue lo que me facilitó y ayudo a finalmente lograr mantener la estabilidad del código. Esta práctica me hizo ver la importancia del manejo correcto de estados HTTP, autenticaciones y actualización de datos en aplicaciones web.

Algunos de los retos fueron el manejo de errores (de manera correcta claro) y para mí el mayor encontrar la mejor estructura para ordenar el proyecto para poder mezclar el front-end y back-end para que todo funcionara en base a los requerimientos. Finalmente, gracias a esta práctica consigo ver un flujo de trabajo lo más similar al de proyectos profesionales en el desarrollo de APIs modernas lo cual agradezco ya que me gusta ver las cosas que se llega a ser capaz en comparación a algo cercano a lo profesional.

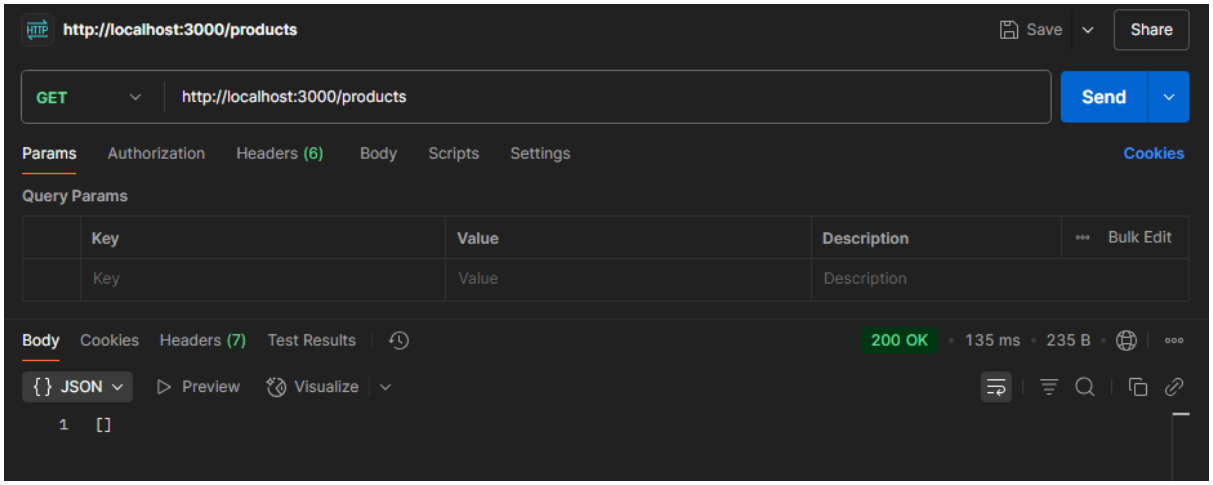
Repositorio: https://github.com/Angel751658/DASW.git

Evidencias

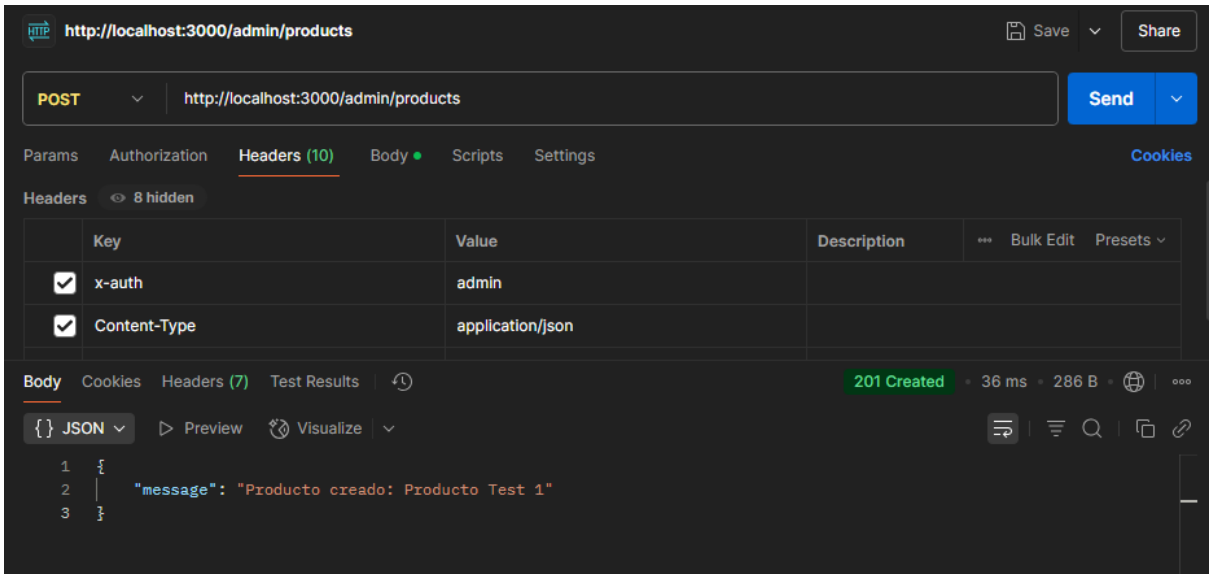
Inicio del servidor

```
C:\Users\Hp\OneDrive\Pro\html\github\DASW\Practica 3>nodemon server.js
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server running on http://localhost:3000
```

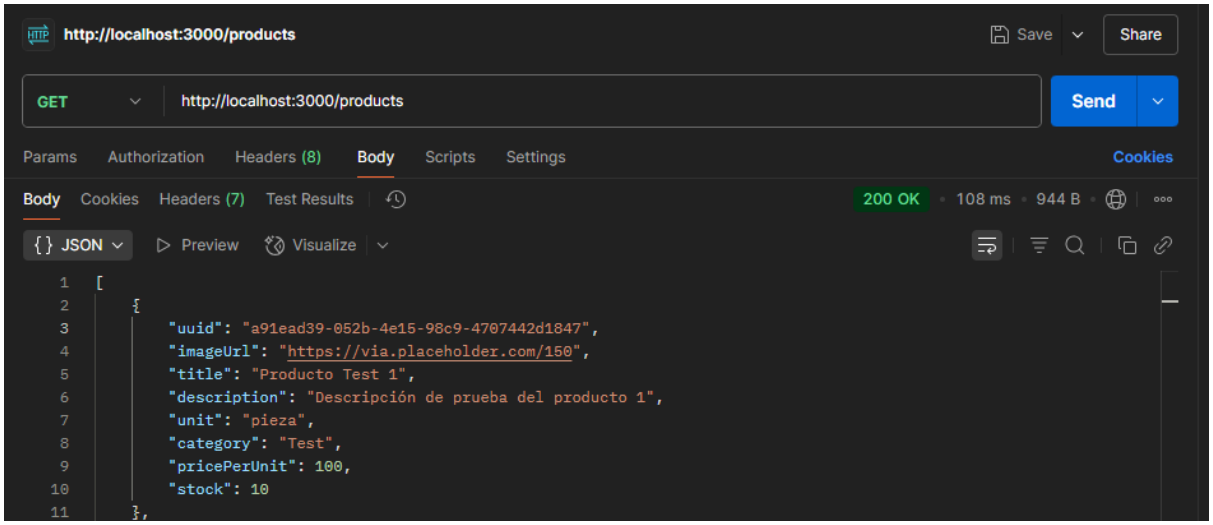
Listado de productos `GET /products`

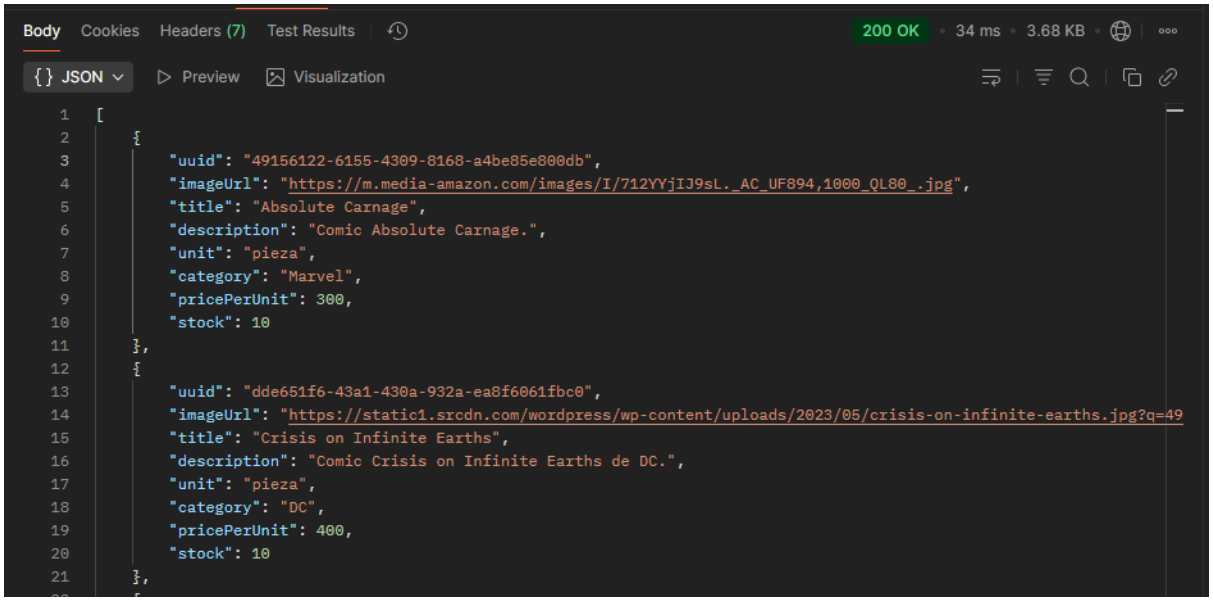


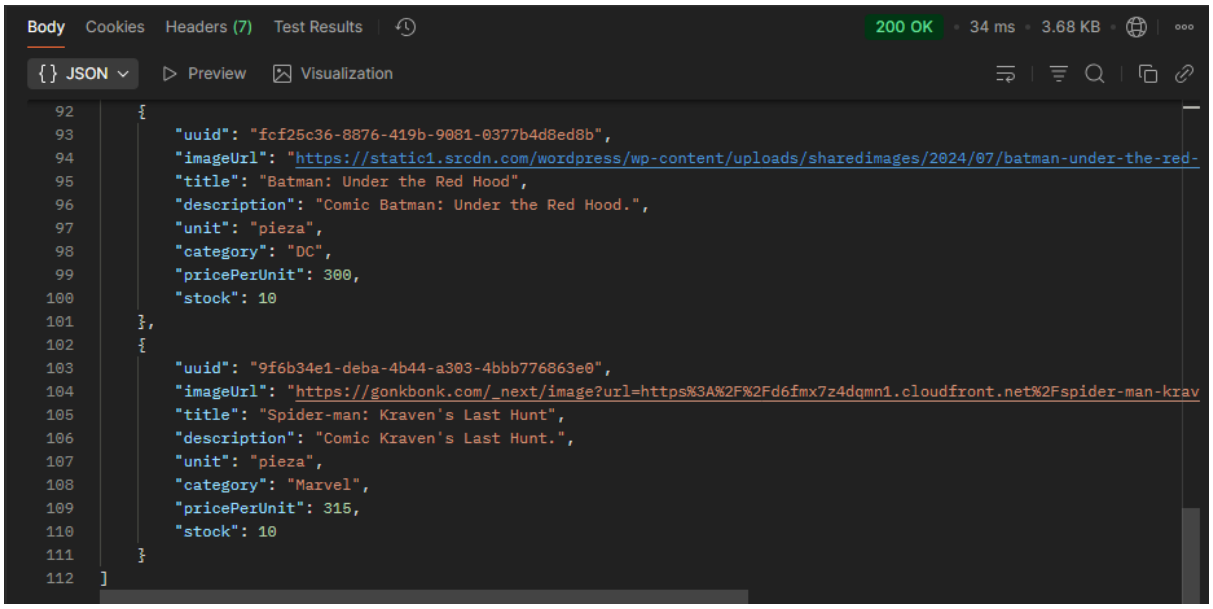
Creación de productos `POST /admin/products`



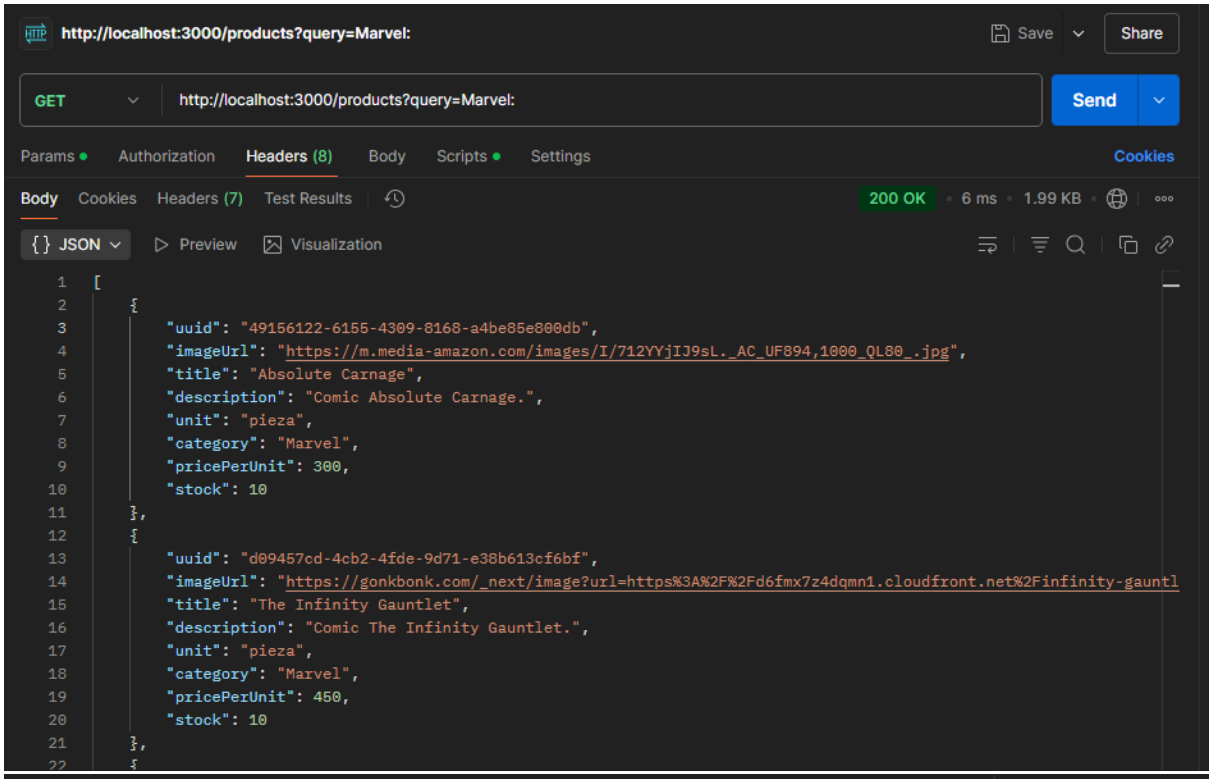
Validación de productos creados

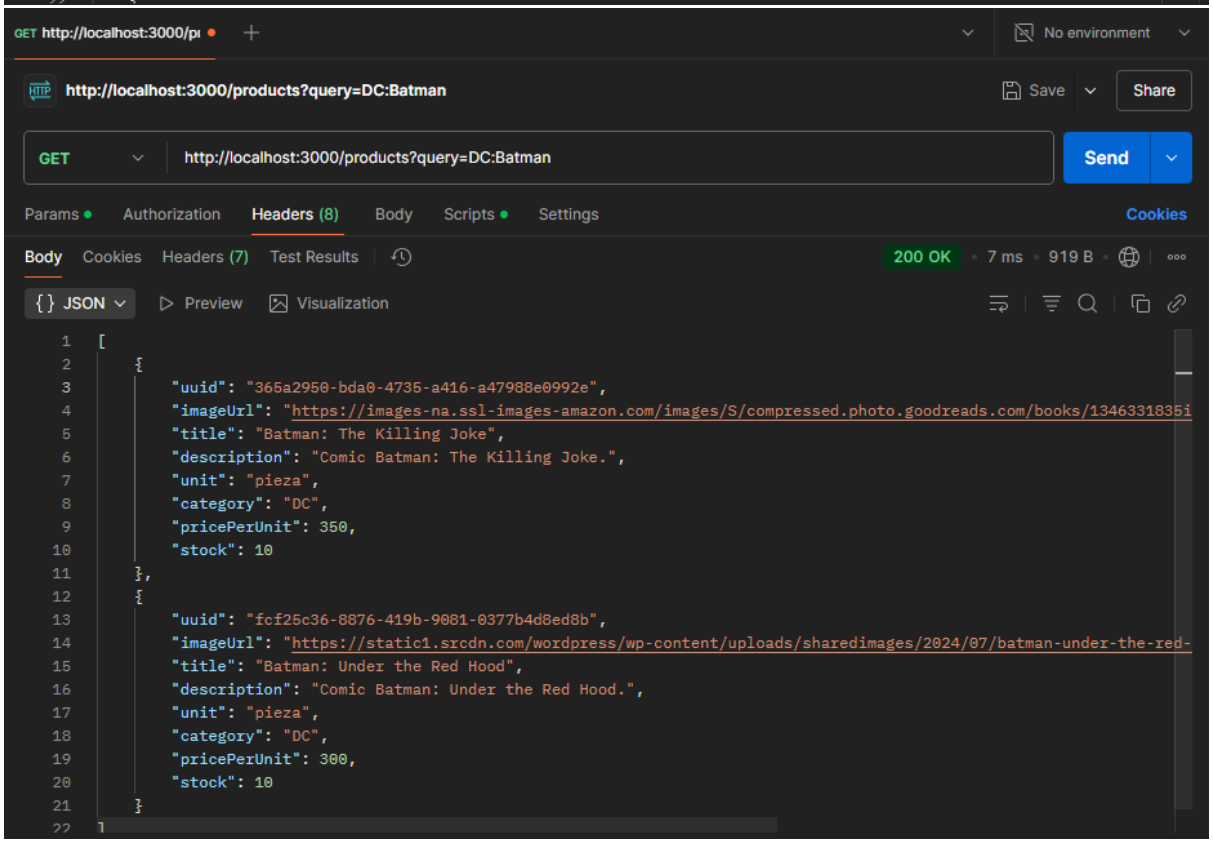




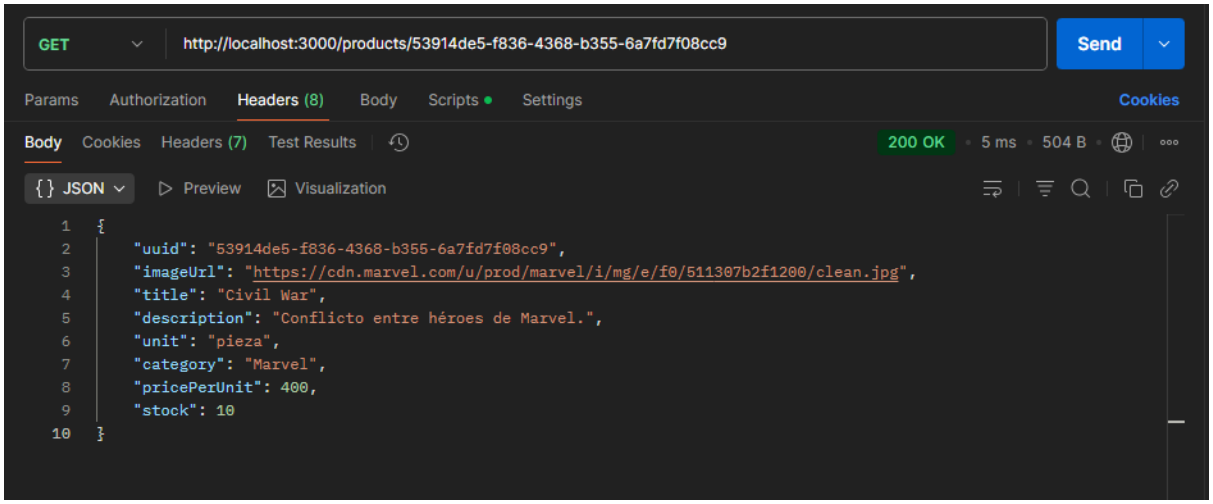


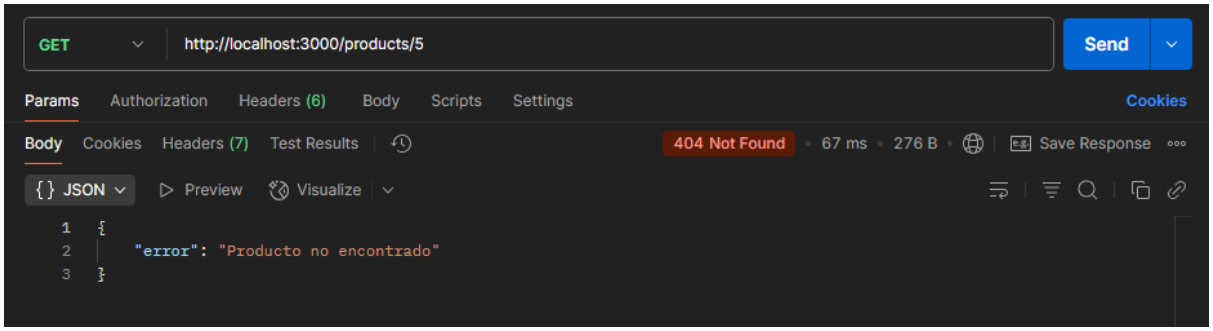
Filtro de productos `GET /products?query=Marvel:, /products?query=DC:Batman`



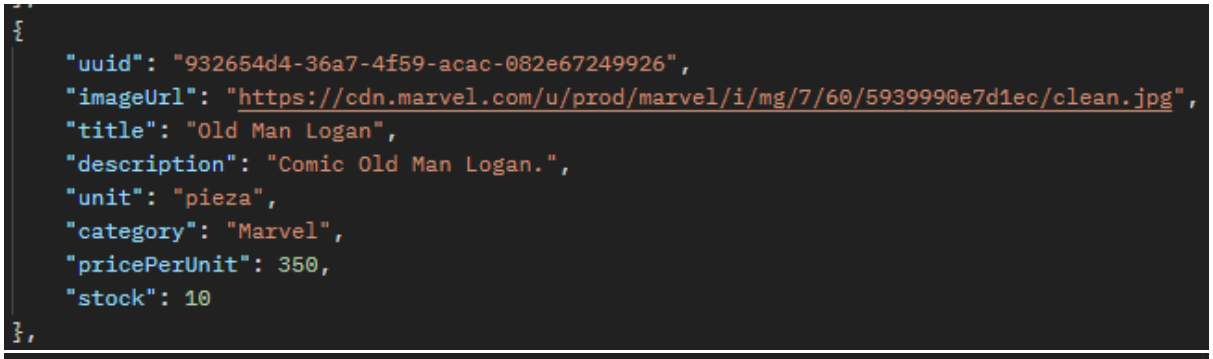


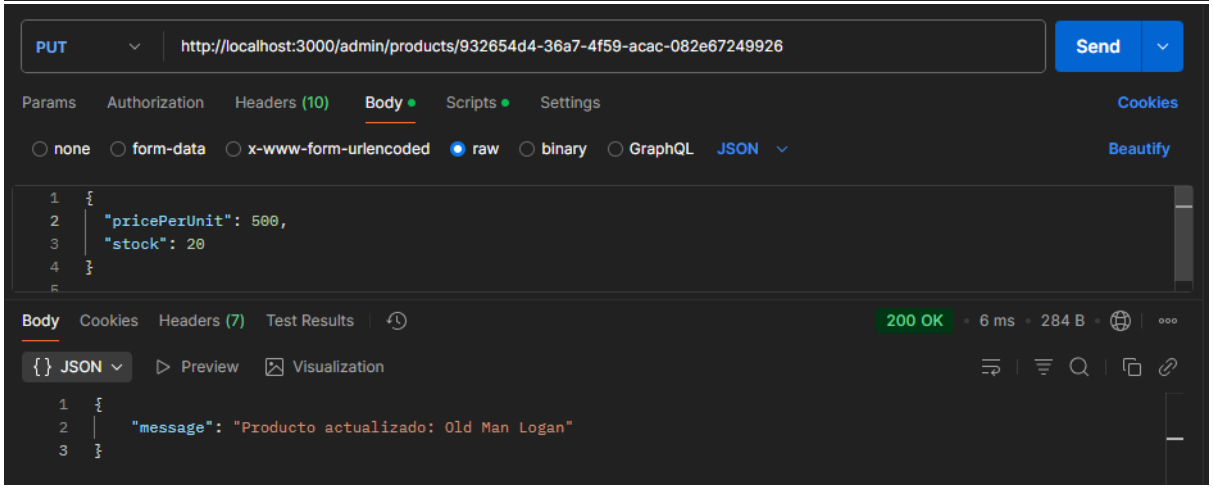
Buscar producto específico `GET /products/:id`

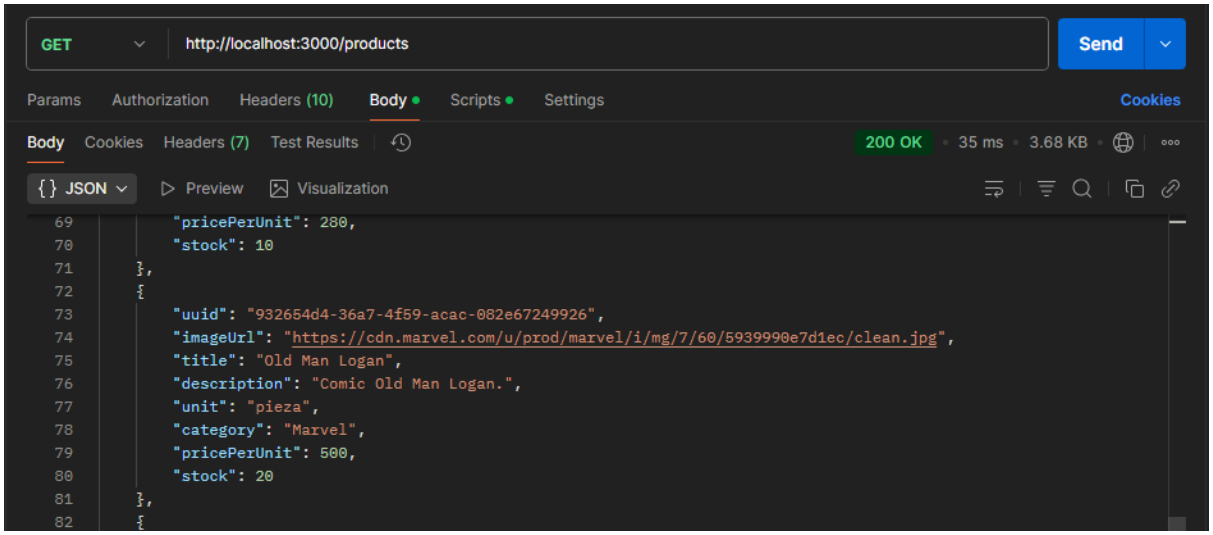




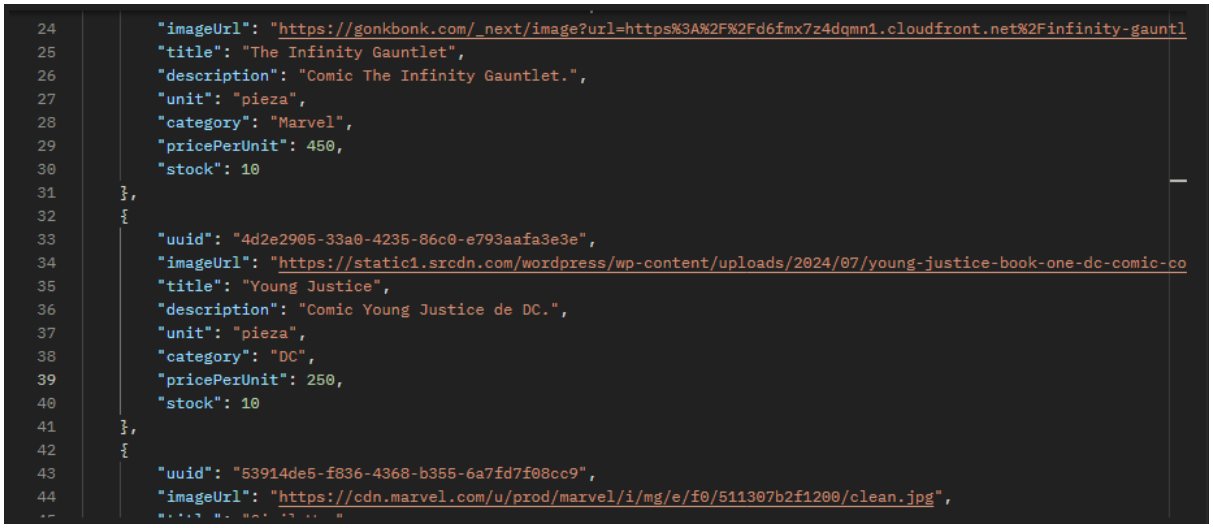
Actualizar producto `PUT /admin/products/:id`

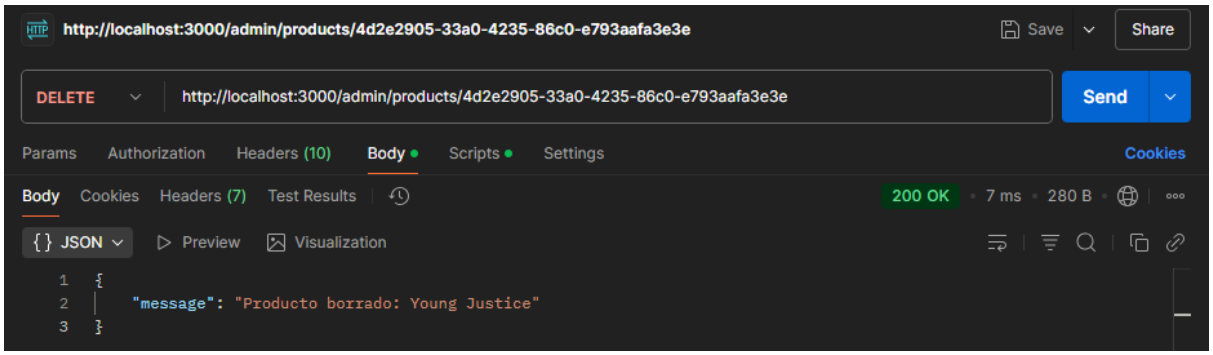






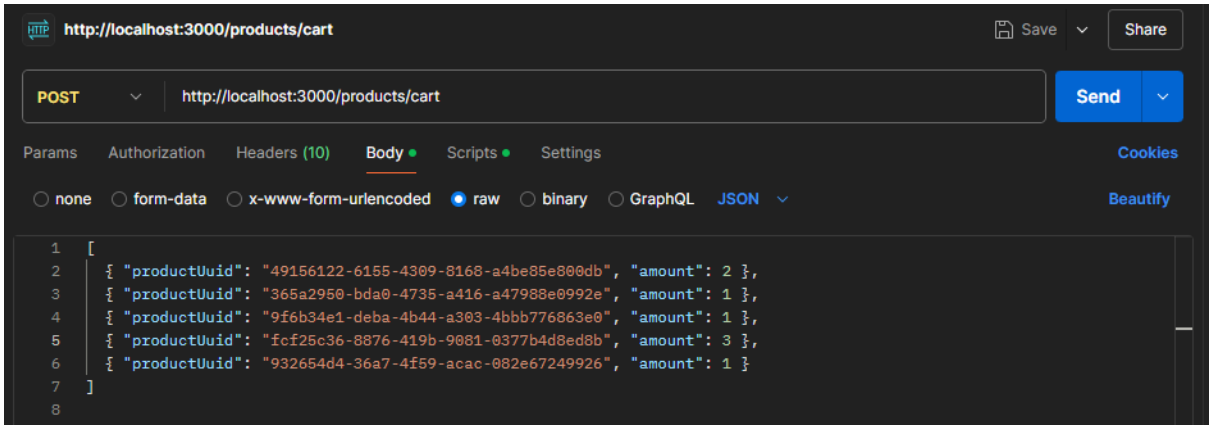
Borrar producto `DELETE /admin/products/:id`

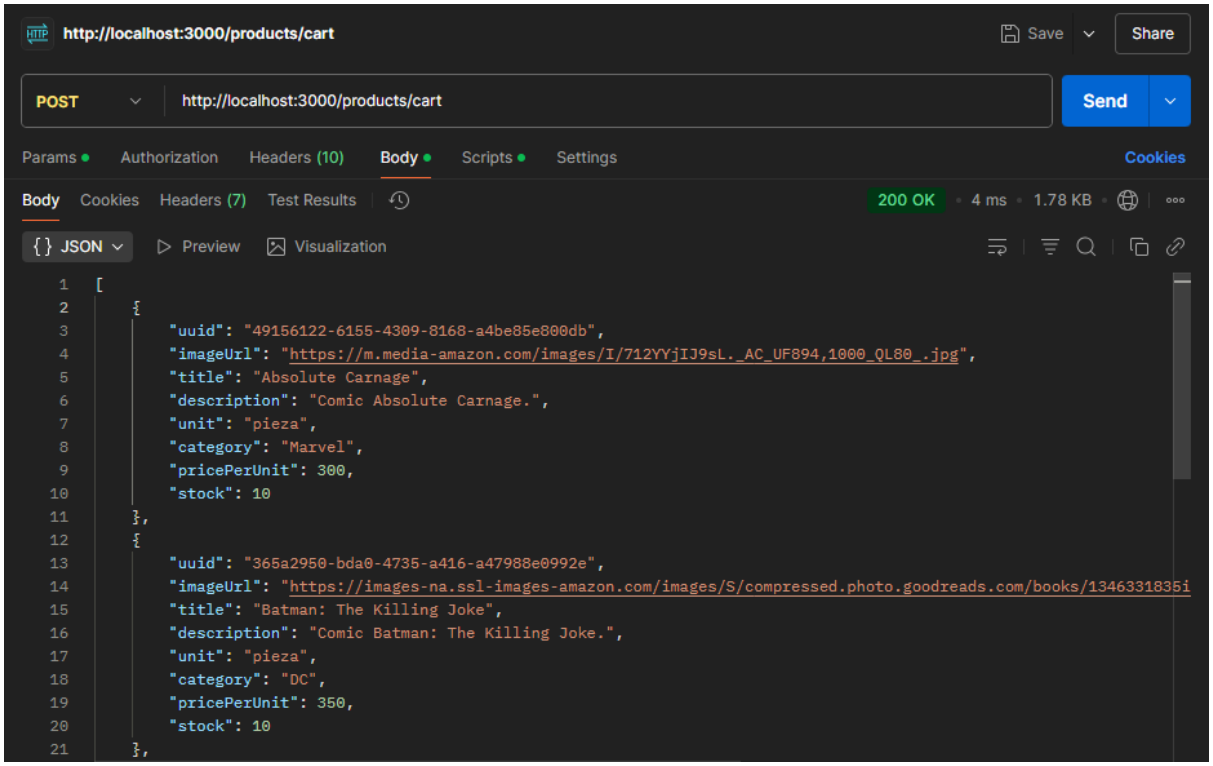


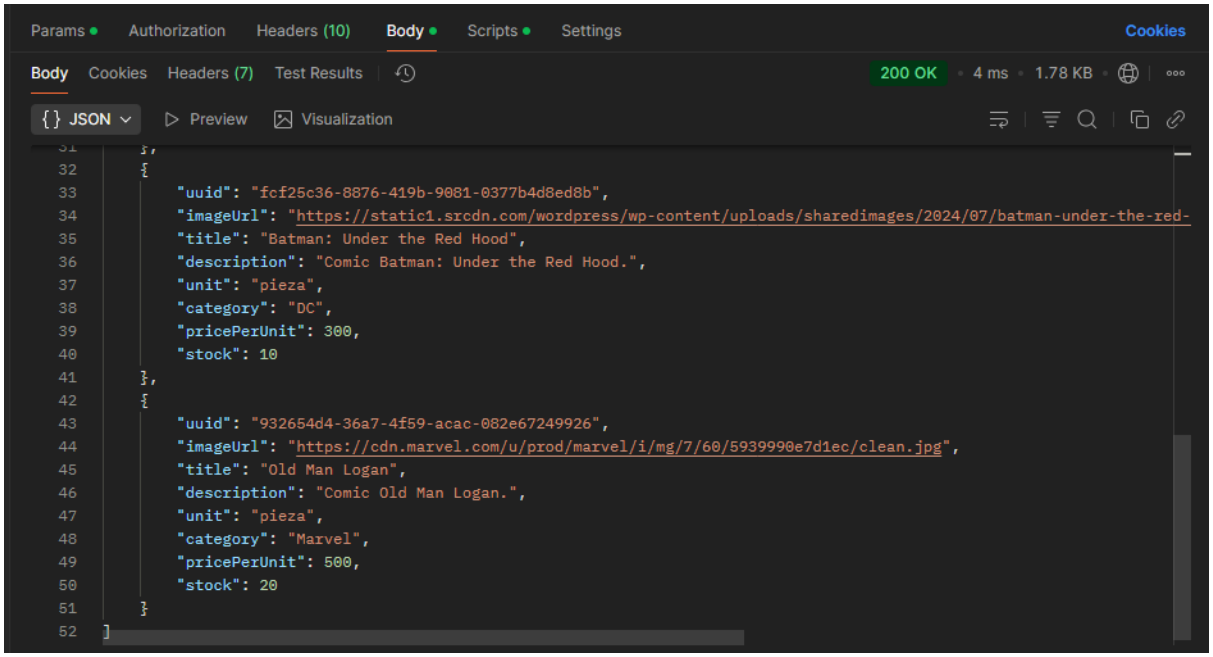


```
    {
        "uuid": "d09457cd-4cb2-4fde-9d71-e38b613cf6bf",
        "imageUrl": "https://gonkbonk.com/_next/image?url=https%3A%2F%2Fd6fmx7z4dqmn1.cloudfront.net%2Finfinity-gauntl
        "title": "The Infinity Gauntlet",
        "description": "Comic The Infinity Gauntlet.",
        "unit": "pieza",
        "category": "Marvel",
        "pricePerUnit": 450,
        "stock": 10
    },
    {
        "uuid": "53914de5-f836-4368-b355-6a7fd7f08cc9",
        "imageUrl": "https://cdn.marvel.com/u/prod/marvel/i/mg/e/f0/511307b2f1200/clean.jpg",
        "title": "Civil War",
        "description": "Conflicto entre héroes de Marvel.",
        "unit": "pieza",
        "category": "Marvel",
        "pricePerUnit": 400,
        "stock": 10
```

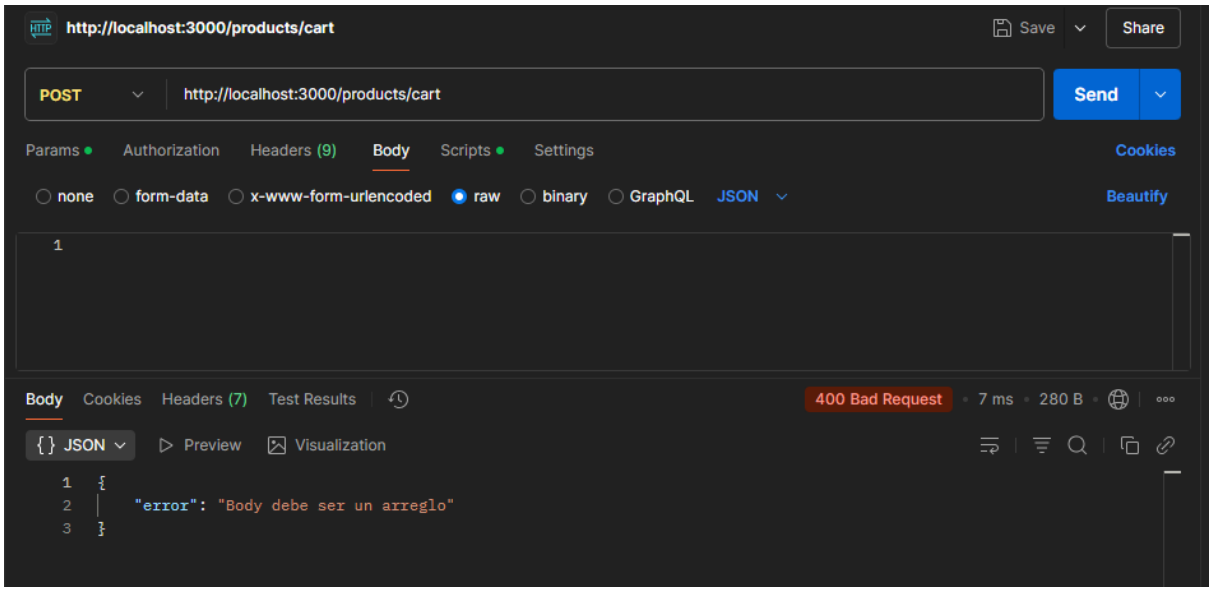

Operaciones de carrito `POST /products/cart`

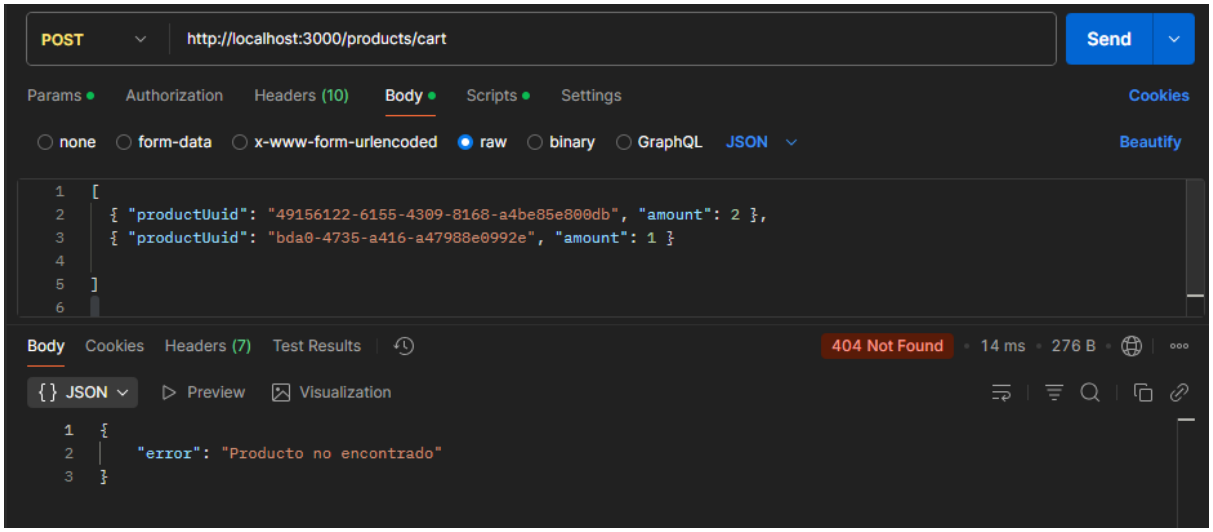




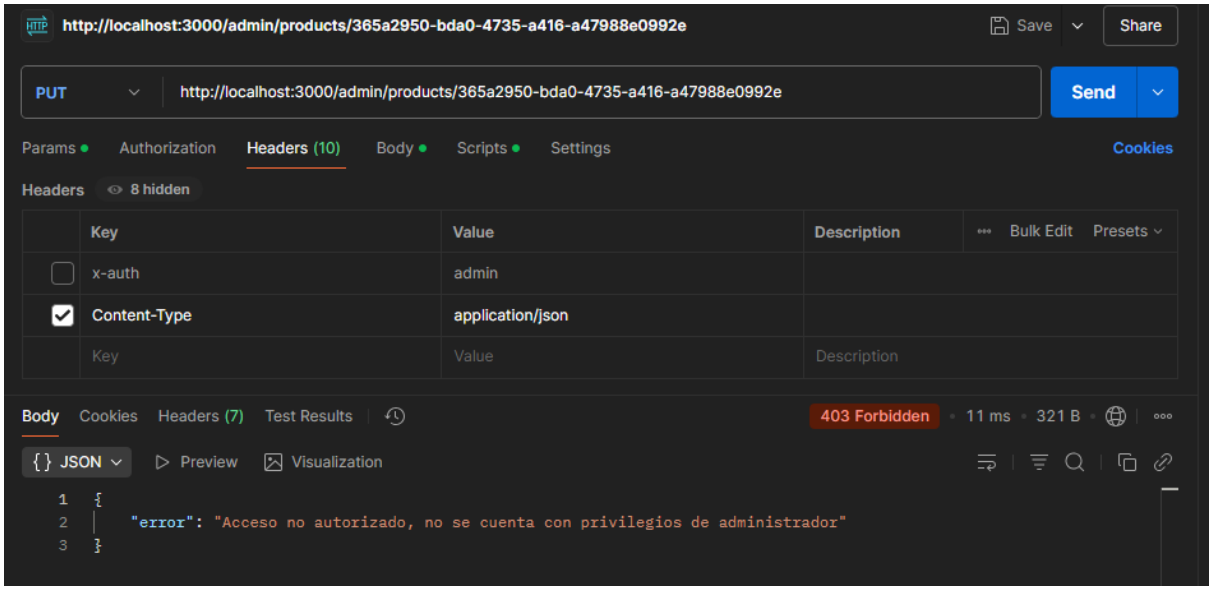


Error 400 al enviar body inválido y 404 al usar UUID inexistente.





Middleware de autenticación



Páginas estáticas