

# EXAMEN PRÁCTICO - MÓDULO 03

PowerShell.

```
PS C:\Users\Tardes\Desktop\productos-api> docker-compose up --build -d
[+] Building 1.0s (11/11) FINISHED
=> [api internal] load build definition from Dockerfile
=> => transferring dockerfile: 159B
=> [api internal] load metadata for docker.io/library/node:20
=> [api internal] load .dockerignore
=> => transferring context: 2B
=> [api 1/5] FROM docker.io/library/node:20@sha256:bcf90f85634194bc51e92f8add1221c7fdeeff94b7f1ff360aeaa7498086d641
=> => resolving docker.io/library/node:20@sha256:bcf90f85634194bc51e92f8add1221c7fdeeff94b7f1ff360aeaa7498086d641
=> [api internal] load build context
=> => transferring context: 1.70kB
=> CACHED [api 2/5] WORKDIR /app
=> CACHED [api 3/5] COPY package*.json ./
=> CACHED [api 4/5] RUN npm install
=> [api 5/5] COPY . .
=> [api] exporting to image
=> => exporting layers
=> => exporting manifest sha256:ff6f555a3e9534a616b02d3a7a5cf2152ef5a25372bb39e0ba612639f12f3274
=> => exporting config sha256:4bbdf6798b3136850577f6b087de796f64810db999a2763e502cc8a7a4c465a
=> => exporting attestation manifest sha256:4a8b0e697f05efaeafa4218f77bb2d20c6d6d78c0dce13c9a267ba0cce338bbe
=> => exporting manifest list sha256:71fd1575829bc2d7b2c1a38c317f53baa74e3d0c504826c3f46b96e203d3f0d9
=> => naming to docker.io/library/productos-api-api:latest
=> => unpacking to docker.io/library/productos-api-api:latest
=> [api] resolving provenance for metadata file
[+] Running 3/3
✔ api
✔ Container productos-api-mongo-1 Started
✔ Container productos-api-api-1 Started
PS C:\Users\Tardes\Desktop\productos-api> docker-compose logs -f api
api-1 | Servidor en puerto 4000
api-1 | Conectado a MongoDB
```

Docker.

The screenshot shows the Docker Desktop application interface. The top bar includes the Docker logo, the name 'docker desktop PERSONAL', a search bar, and a 'Sign in' button. Below the top bar, a sidebar on the left lists various Docker components: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'productos-api' service, which is currently 'Up'. It shows the service's name, image, and a 'View configurations' button. Below the service status, a log viewer displays the output of the 'api' service, showing messages about connection acceptance and session management. The bottom status bar indicates the engine is running with 1.27 GB RAM and 0.00% CPU usage.

## Prueba get antes de añadir productos.

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/products/`. The response is a 200 OK status, indicating a successful GET operation. The response body is empty, which is typical for a GET request that returns a list of items without including the items themselves in the response body.

Key	Value	Description
Key	Value	Description

```
1 {}
```

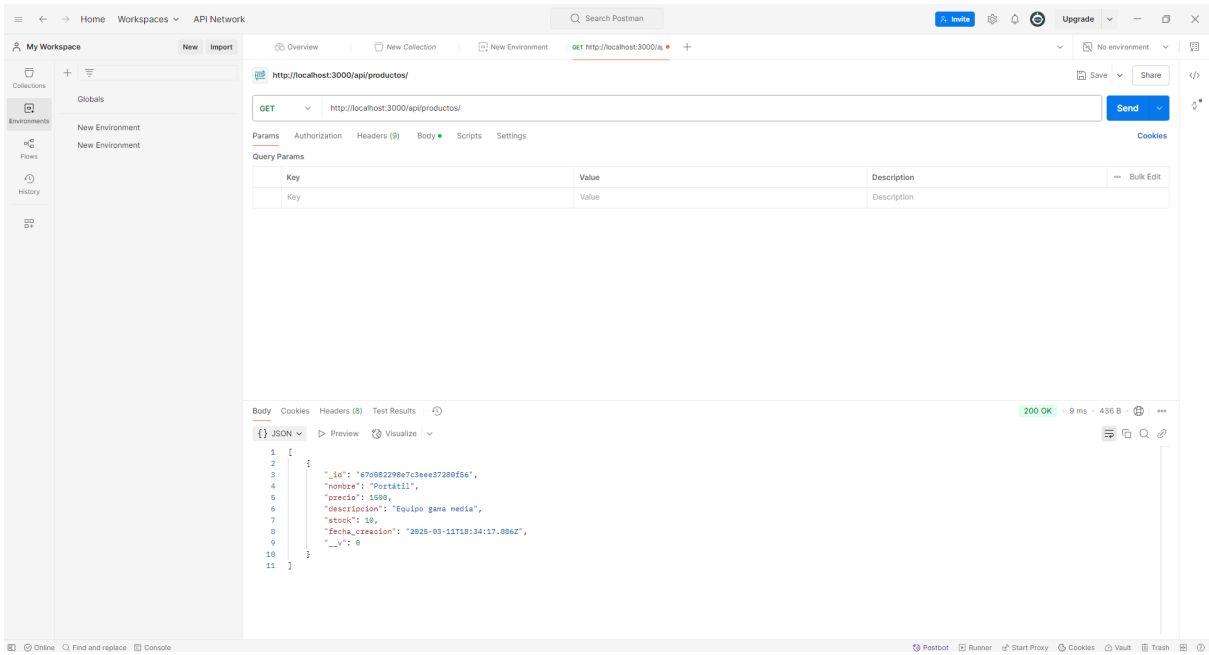
## Primer post.

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/products/`. The request body is a JSON object representing a product. The response is a 201 Created status, indicating that a new product has been successfully added to the database.

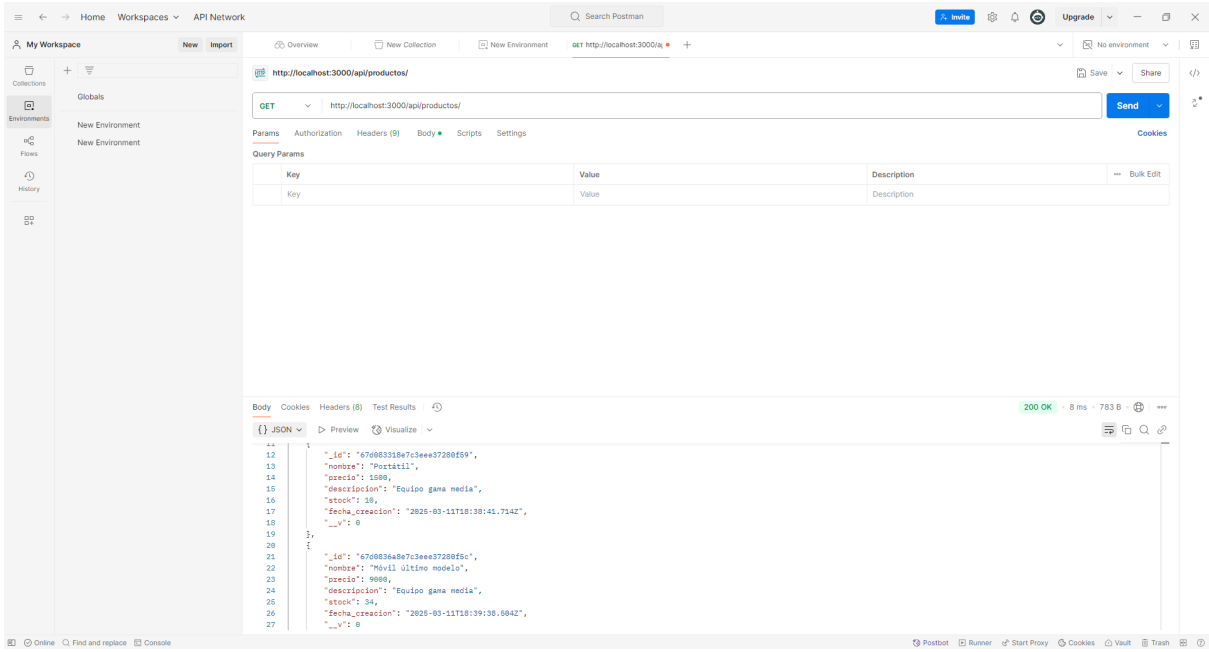
```
1 {
2   "nombre": "Portátil",
3   "precio": 1599,
4   "descripcion": "Equipo gama media",
5   "stock": 10
6 }
```

```
1 {
2   "nombre": "Portátil",
3   "precio": 1599,
4   "descripcion": "Equipo gama media",
5   "stock": 10,
6   "_id": "67d682298c7c3ee07288f564",
7   "fecha_creacion": "2025-09-11T18:34:17.886Z",
8   "_v": 0
9 }
```

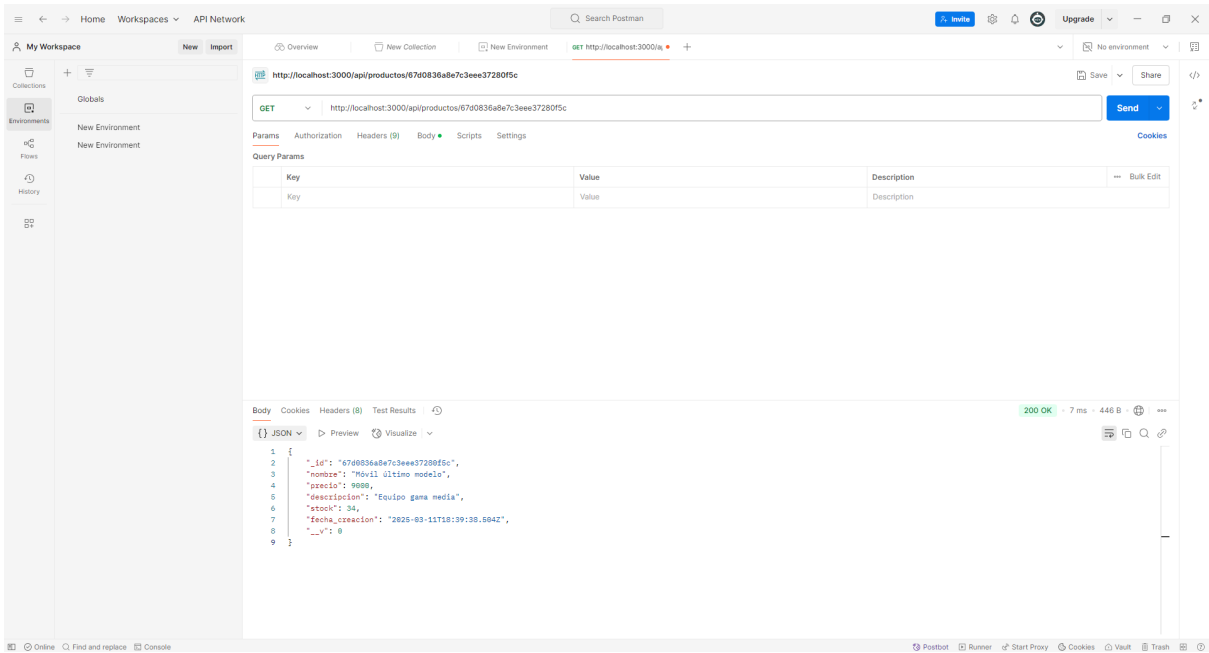
# Get tras añadir el producto.



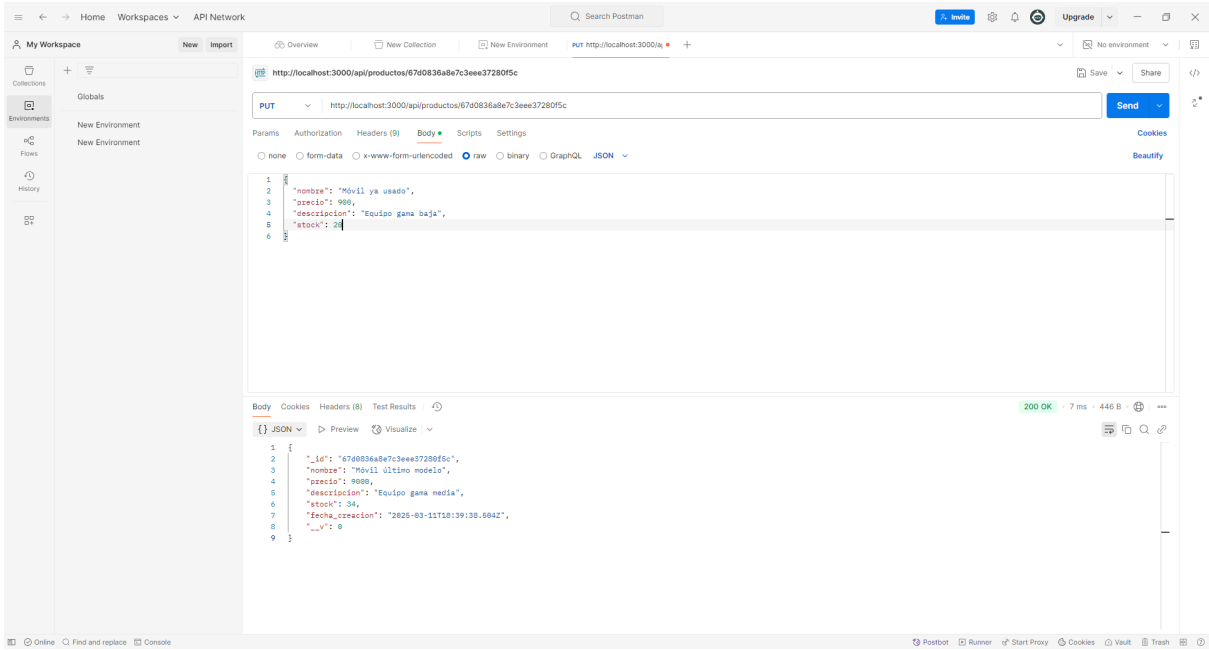
# Get tras añadir un segundo producto.



# Get con id.



# Put.



## Put resuelto (se actualiza correctamente).

The screenshot shows the Postman interface with a PUT request to `http://localhost:3000/api/productos/67d0836a9e7c3ee37280f5c`. The request body is a JSON object:

```
1 {
2   "nombre": "Movil ya usado",
3   "precio": 999,
4   "descripcion": "Equipo gama baja",
5   "stock": 20
6 }
```

The response is a 200 OK status with a response time of 11 ms and a body size of 438 B. The response body is a JSON object:

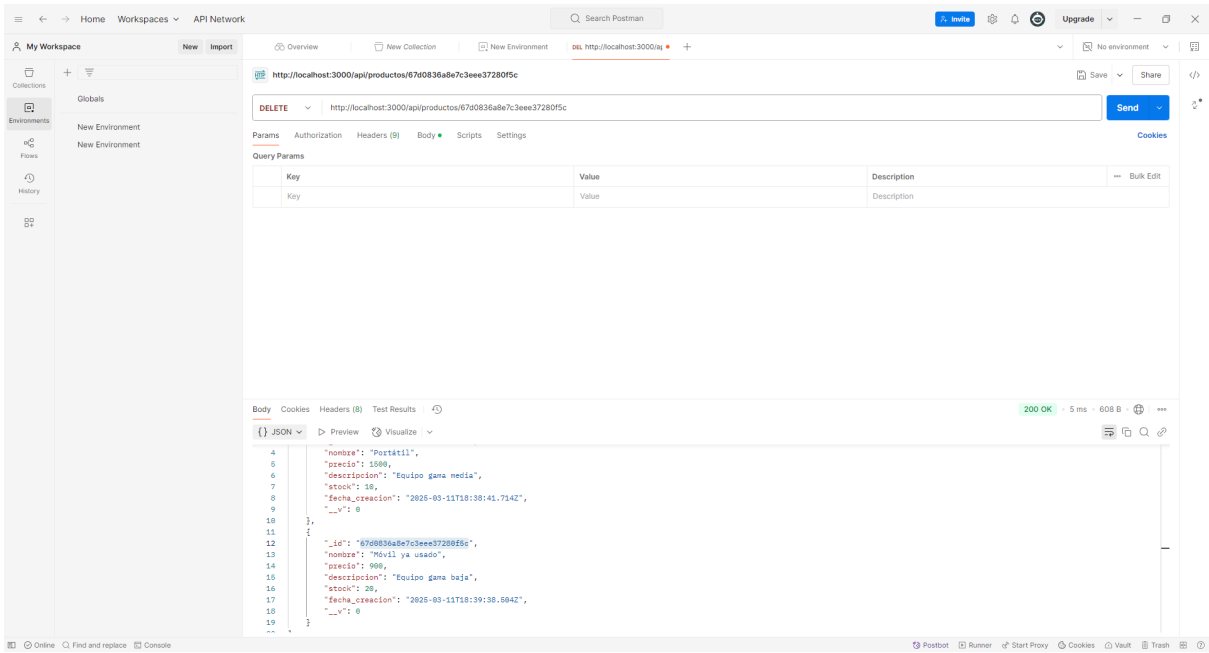
```
1 {
2   "_id": "67d0836a9e7c3ee37280f5c",
3   "nombre": "Movil ya usado",
4   "precio": 999,
5   "descripcion": "Equipo gama baja",
6   "stock": 20,
7   "fecha_creacion": "2025-03-11T18:39:38.584Z",
8   "_v": 0
9 }
```

## Get de los productos tras la modificación.

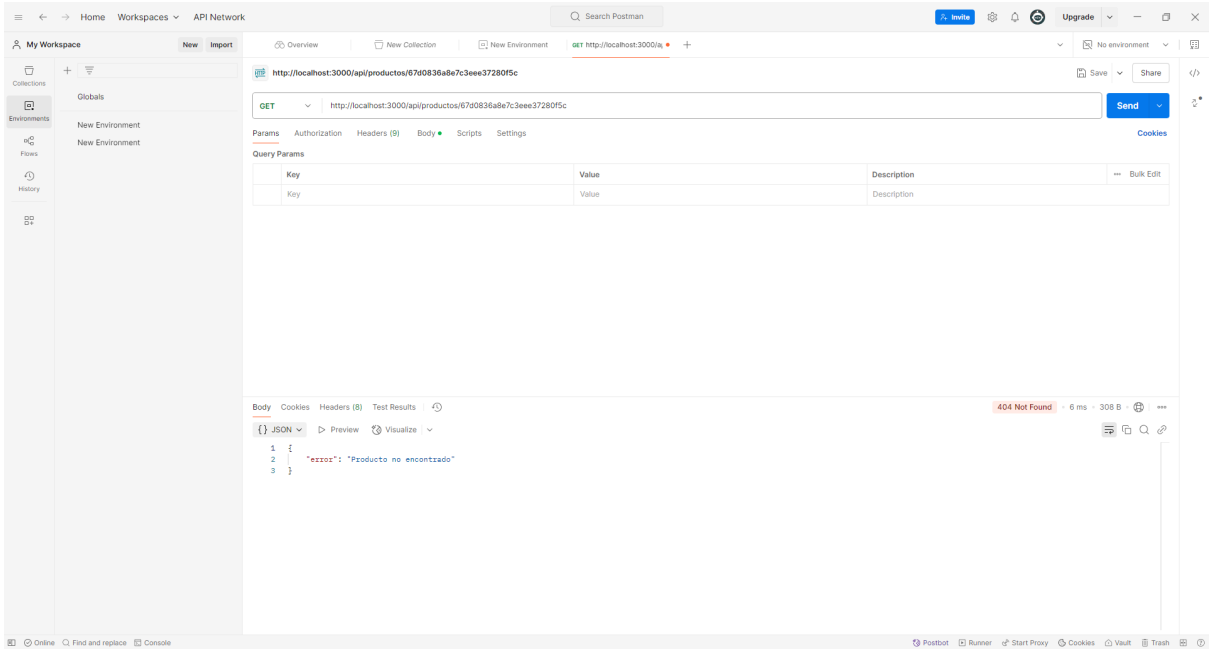
The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/productos/`. The response is a 200 OK status with a response time of 7 ms and a body size of 775 B. The response body is a JSON array of two product objects:

```
11 [
12   {
13     "_id": "67d0836a9e7c3ee37280f59",
14     "nombre": "Portatil",
15     "precio": 1599,
16     "descripcion": "Equipo gama media",
17     "stock": 10,
18     "fecha_creacion": "2025-03-11T18:38:41.754Z",
19     "_v": 0
20   },
21   {
22     "_id": "67d0836a9e7c3ee37280f5c",
23     "nombre": "Movil ya usado",
24     "precio": 999,
25     "descripcion": "Equipo gama baja",
26     "stock": 20,
27     "fecha_creacion": "2025-03-11T18:39:38.584Z",
28     "_v": 0
29   }
30 ]
```

Delete.



Confirmación de que el producto ha sido eliminado.



Último get para mostrar que sólo queda un producto tras el delete.

