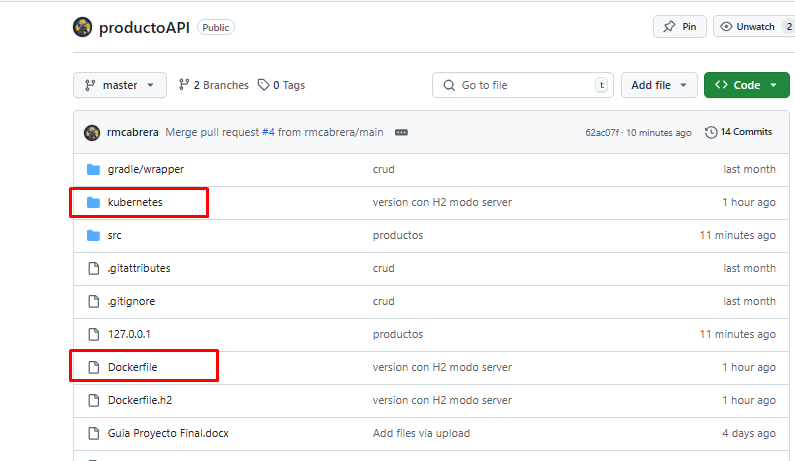
**GUÍA DE DESPLIEGUE DE UNA APLICACIÓN SPRING BOOT EN KUBERNETES CON DOCKER Y KILLERCODA CON UN SERVIDOR DE BASE DE DATOS H2**

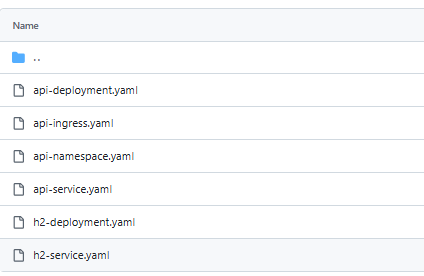
Exponer una aplicación Spring Boot en Kubernetes utilizando NodePort, incluyendo el acceso a múltiples réplicas. Para el despliegue usaremos la plataforma Killercoda. Se configurará un Ingress para exponer la aplicación.

1. **Repositorio del proyecto:**

[**https://github.com/rmcabrera/productoAPI**](https://github.com/rmcabrera/productoAPI)



En la carpeta kubernetes se encuentras los archivos de configuración yaml que se usaran para el despliegue de los contenedores. Asimismo, el archivo Dockerfile contiene la configuración para crear la imagen.



1. **Creación de la imagen del proyecto y almacenamiento en Docker Registry**

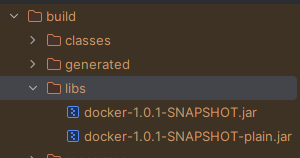
Clonamos el repositorio.

git clone https://github.com/rmcabrera/productoAPI

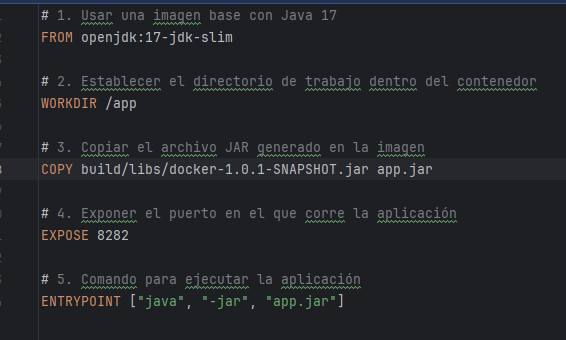
Generamos el archivo jar del proyecto.

gradlew build





Verificamos el archivo Dockerfile



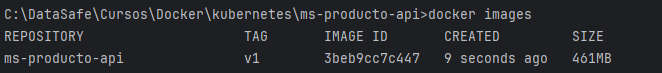
Generamos la imagen con el siguiente comando:

docker build -t ms-producto-api:v2 .

docker images

Resultado esperado





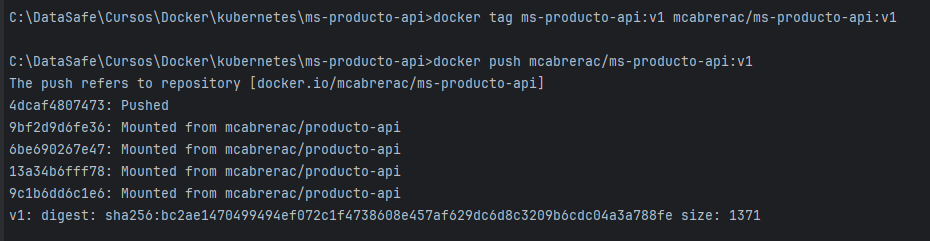
Iniciamos sesión con nuestra cuenta para iniciar el registro en Docker Registry, y subimos la imagen:

docker login

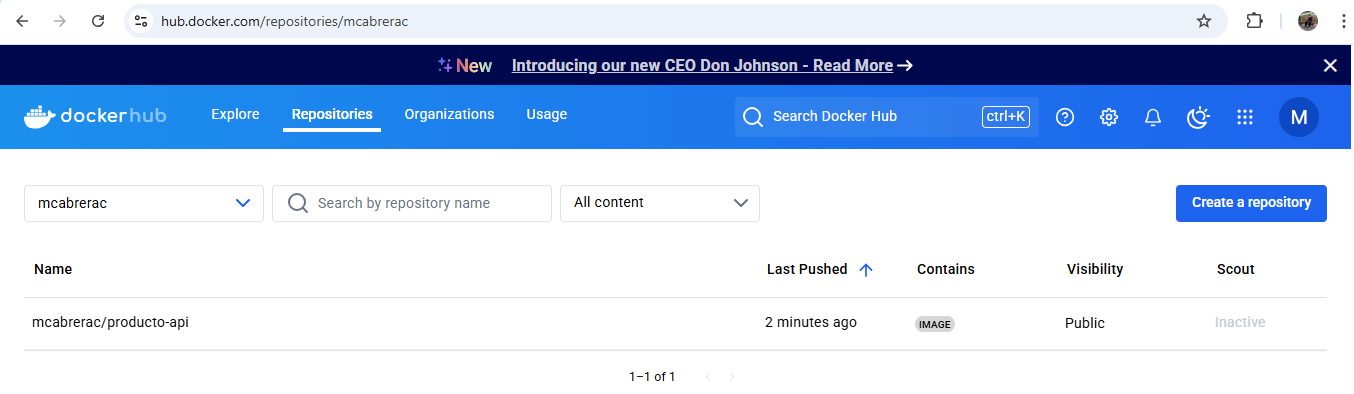
docker tag ms-producto-api:v1 mcabrerac/ms-producto-api:v1

docker push mcabrerac/ms-producto-api:v1

Resultado esperado:



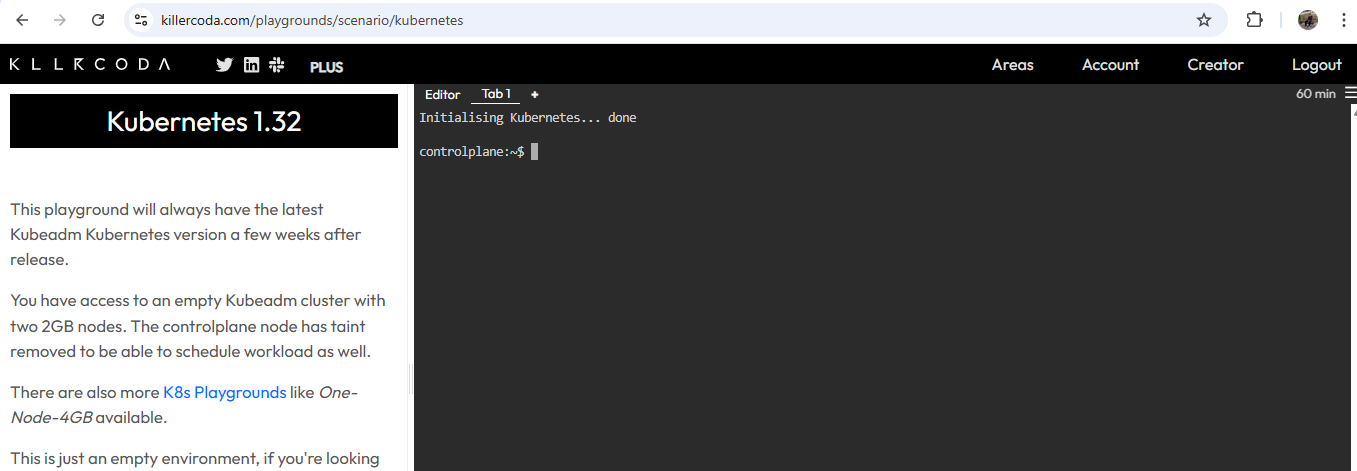
<https://hub.docker.com/r/mcabrerac/ms-producto-api>



1. **Desplegar el Proyecto en Killercoda**

Acceder a Killercoda y clonar el repositorio de GitHub:

<https://killercoda.com/playgrounds/scenario/kubernetes>

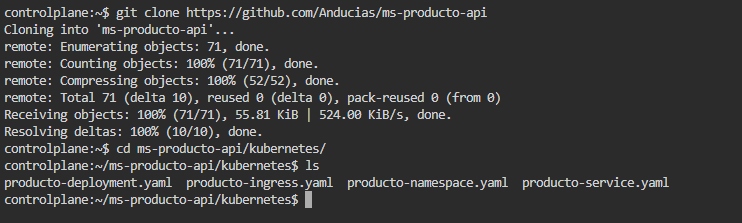


git clone https://github.com/rmcabrera/productoAPI

cd productoAPI/kubernetes/

ls

Resultado esperado:



**Debe contener:**

* api-namespace.yaml
* h2-deployment.yaml
* h2-service.yaml
* api-deployment.yaml
* api-service.yaml
* api-ingress.yaml

1. **Crear el Namespace, Deployment y el Service**

kubectl apply -f api-namespace.yaml

kubectl apply -f h2-deployment.yaml

kubectl apply -f h2-service.yaml

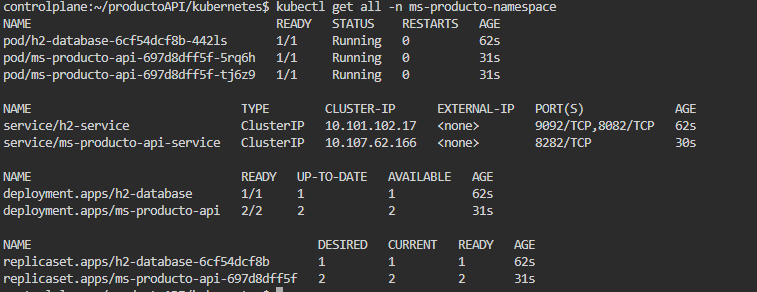
kubectl apply -f api-deployment.yaml

kubectl apply -f api-service.yaml

Verificar que todo se haya creado correctamente:

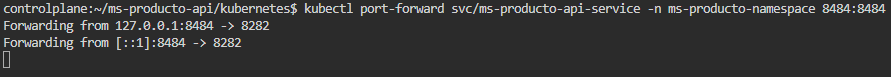
kubectl get all -n ms-producto-namespace

En donde se evidencia que se han generado correctamente el despliegue.



1. **Validar el Acceso Interno al Servicio**

kubectl port-forward svc/ms-producto-api-service -n ms-producto-namespace 8484:8484



Abrimos otra ventana de comandos y probamos:

curl -X GET http://<IP\_SERVICIO>:8484/api/productos

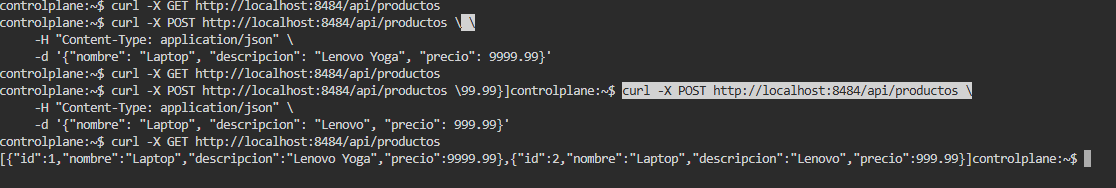
Debe devolver un JSON vacío o con los productos existentes.

Prueba de inserción:

curl -X POST http://<IP\_SERVICIO>:8484/api/productos \

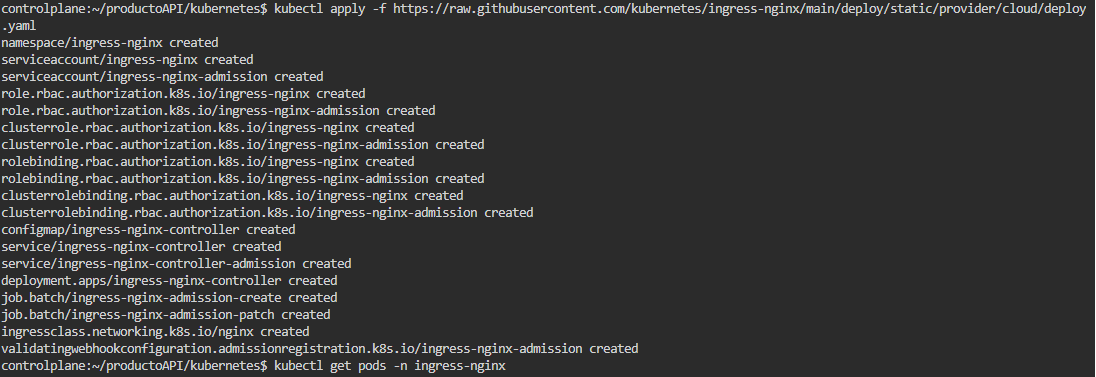
-H "Content-Type: application/json" \

-d '{"nombre": "Laptop", "descripcion": "Lenovo", "precio": 999.99}'



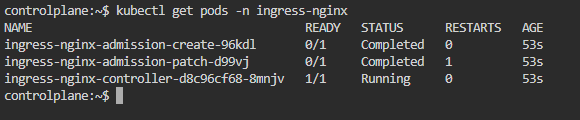
1. **Instalar el NGINX Ingress Controller**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml



Verifica que los pods del controlador estén activos y en Ejecución:

kubectl get pods -n ingress-nginx



1. **Aplicar el Ingress.yaml**

Desplegamos el archivo api-ingress.yaml

kubectl apply -f api-ingress.yaml

Verificamos que se creó correctamente:

kubectl get ingress -n ms-producto-namespace

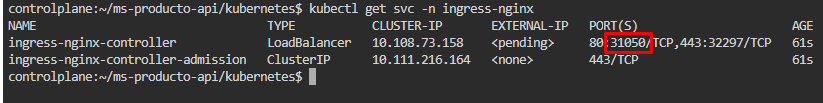


1. **Pruebas de GET, POST, PUT, DELETE**

Para realizar las pruebas debemos obtener el puerto del ingress

kubectl get svc -n ingress-nginx

Anotamos el puerto indicado y reemplazamos en los comandos curl



***GET***

curl -H "Host: ms-producto.server" <http://localhost:31050/api/productos>

***POST***

curl -X POST -H "Host: ms-producto.server" -H "Content-Type: application/json" \

-d '{"nombre": "Laptop", "descripcion": "Lenovo Yoga", "precio": 1000}' \

[http://localhost: 31050/api/productos](http://localhost:%2031050/api/productos)

***PUT***

curl -X PUT -H "Host: ms-producto.server" -H "Content-Type: application/json" \

-d '{"nombre": "Laptop Pro", "descripcion": "Lenovo Yoga 2024", "precio": 1200}' \

<http://localhost:31050/api/productos/1>

***DELETE***

curl -X DELETE -H "Host: ms-producto.server" <http://localhost:31050/api/productos/1>

1. **CONFIGURACION OPCIONAL INSTALAR NGROK**

Descargar e instala ngrok a fin de poder acceder externamente desde Postman al servicio desplegado en Killercoda.

wget https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-stable-linux-amd64.zip

unzip ngrok-stable-linux-amd64.zip

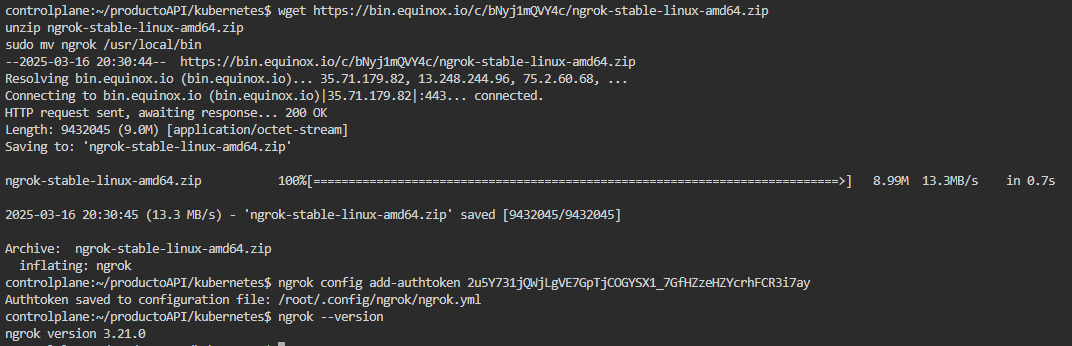
sudo mv ngrok /usr/local/bin

Autenticar con cuenta de ngrok:

ngrok config add-authtoken 2u5Y731jQWjLgVE7GpTjCOGYSX1\_7GfHZzeHZYcrhFCR3i7ay

Verificar la instalación:

ngrok --version



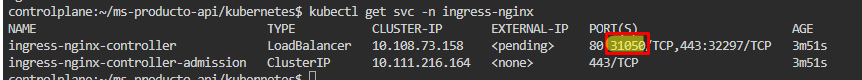
**Identificar el puerto del Ingress Controller**

Necesitamos encontrar el puerto que usa el Ingress Controller para exponer las solicitudes externas.

Ejecutar el siguiente comando:

kubectl get svc -n ingress-nginx

Debe verse algo como:



En este caso el puerto de HTTP es 31050

**Exponer el Ingress Externamente con ngrok**

ngrok http <puerto> --host-header=ms-producto.server

curl -X GET \

https://500c-212-2-243-224.ngrok-free.app/api/productos

curl -X POST -H "Content-Type: application/json" \

-d '{"nombre": "Producto 1", "precio": 100}' \

https://500c-212-2-243-224.ngrok-free.app/api/productos

curl -X PUT -H "Content-Type: application/json" \

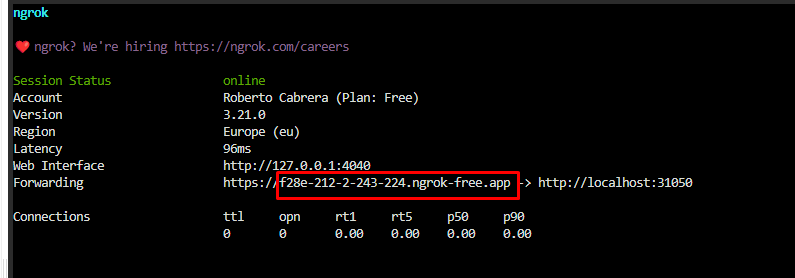
-d '{"nombre": "Producto Actualizado", "precio": 150}' \

https://500c-212-2-243-224.ngrok-free.app/api/productos/1

curl -X DELETE \

https://500c-212-2-243-224.ngrok-free.app/api/productos/1

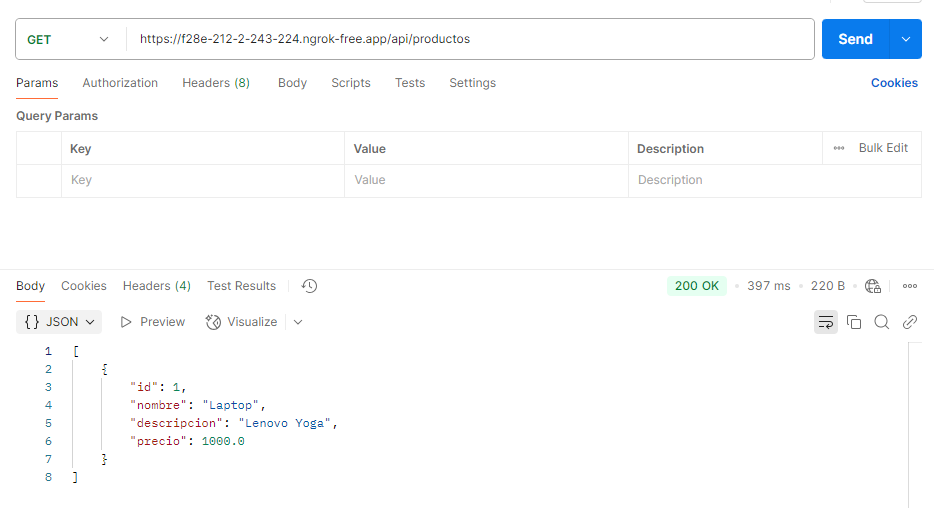
Se mostrará la url que usaremos desde postman



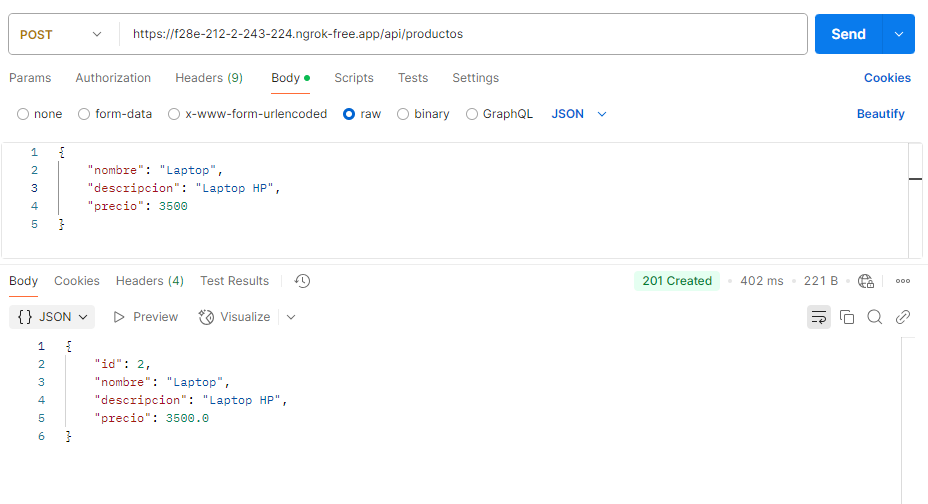
Copiamos la url generada, en este caso: f28e-212-2-243-224.ngrok-free.app

**Probar el Acceso Externo desde Postman**

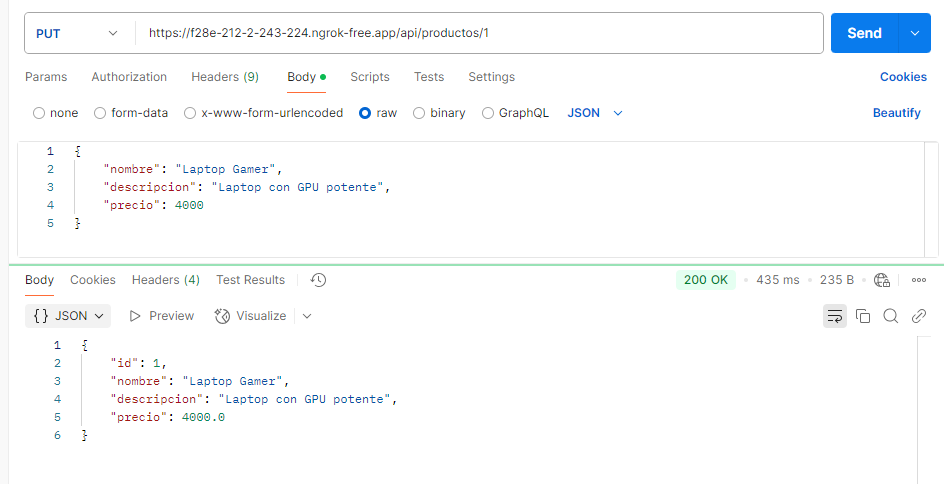
***Listar***



***Crear***

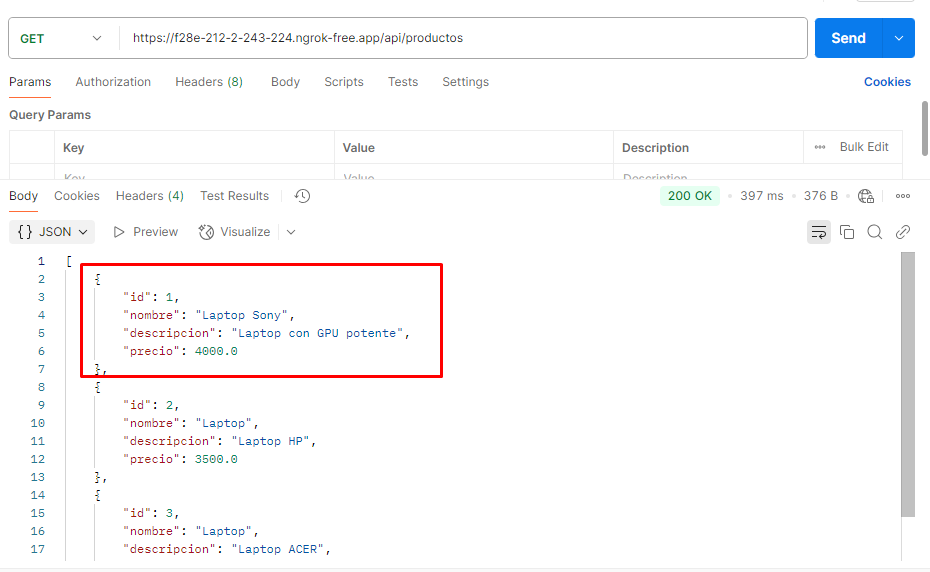


***Actualizar***



***Eliminar***

Listamos los productos actuales, y mediante el id eliminaremos el seleccionado en la imagen.



Ejecutamos la eliminación y al volver a consultar, el producto no se mostrará en la lista.

