

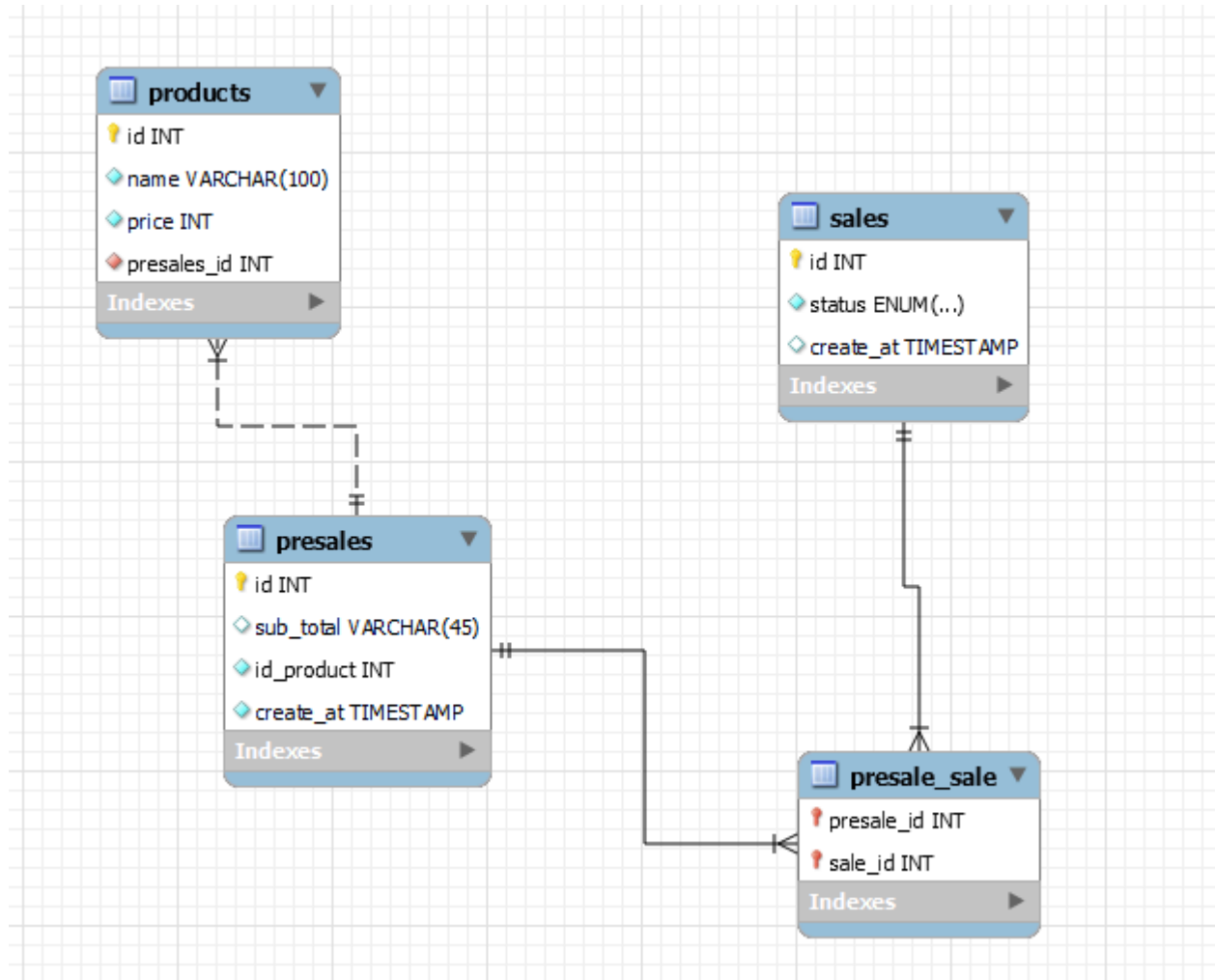
Identificación módulo	
Aplicación	API
Nombre	Learning microservices
Versión	1
Fecha de elaboración	27/05/2024
Elaborado por	Michael Andrés Ardila Rotavista

Objetivo
<p>Desarrollar una API gateway para la gestión de venta de productos, que centralizara la información y enrutara las direcciones hacia los microservicios, esta API contendrá unos LOGS que se almacenara en una base de datos no SQL para brindar el respectivo control o actualizaciones a la API.</p> <p>Se deben crear dos microservicios, uno para la gestión de los registros de los productos (crear, actualizar) etc, y otro para la gestión de las ventas del producto como (crear venta, cancelar venta).</p>

Información código fuente
<ul style="list-style-type: none"> - El código fuente del proyecto se encuentra desarrollado en: - Backend: Laravel 11, PHP 8.2, Docker - DB: MySQL - La ruta de acceso al proyecto + http://sio.localhost/gestion-documental/documental/documentos - El proyecto cuenta con un sistema de control de versiones en Git donde almacena los cambios realizados por medio de los commits, y estos se pueden visualizar a través del repositorio de GitHub, cuya dirección es: https://github.com/MaiLear/learning-microservice

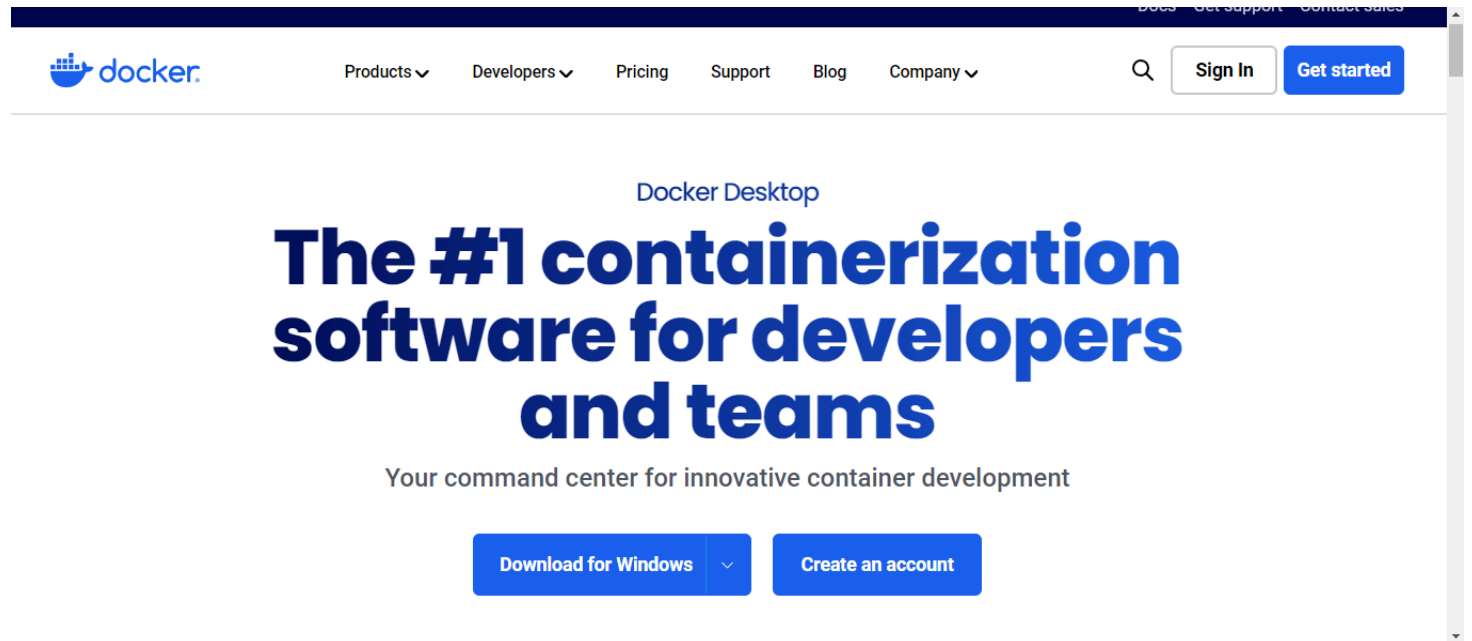
Modelo de datos

- Diagrama del modelo entidad relación con las tablas que se utiliza para almacenar los datos del la API

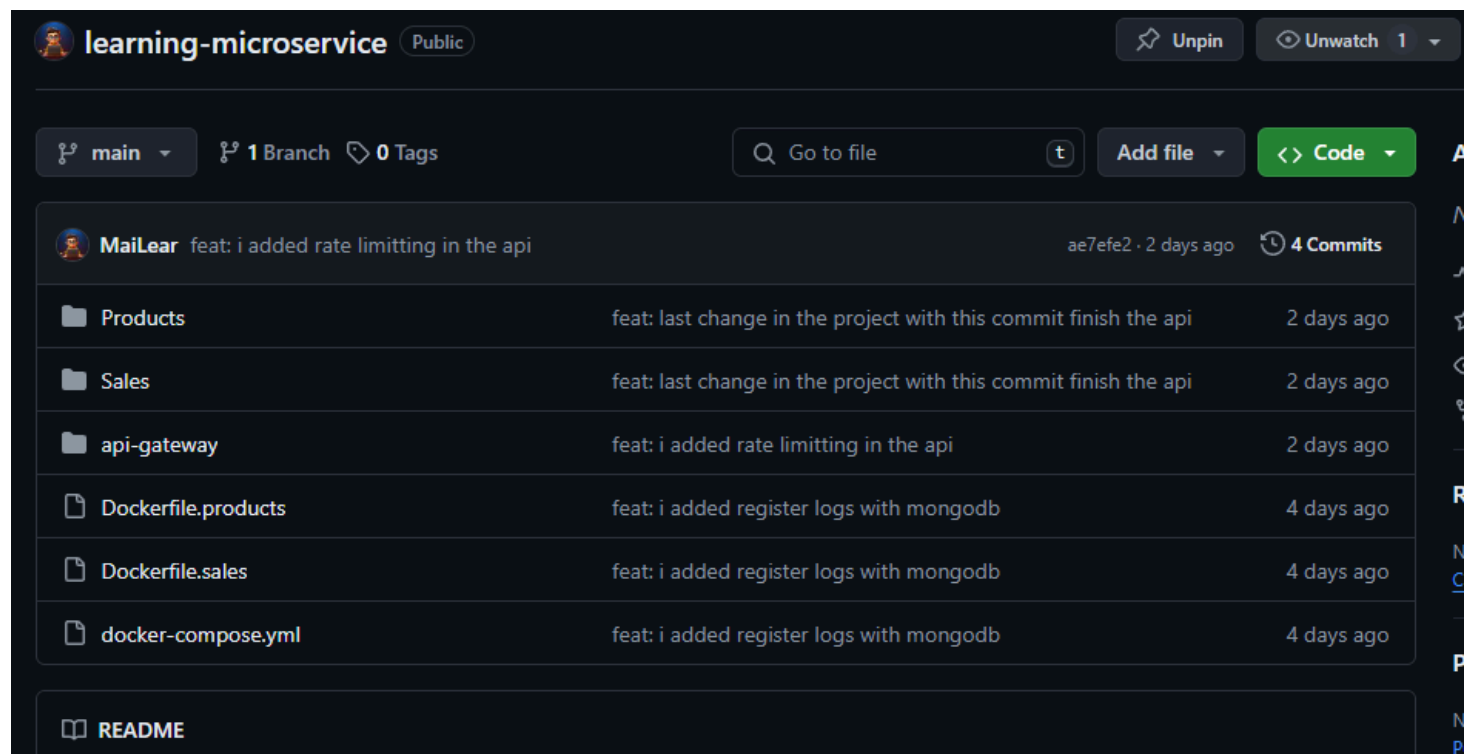


Como usar la API

Primero dirígete a <https://www.docker.com/products/docker-desktop/> y descarga la aplicación Docker Desktop:

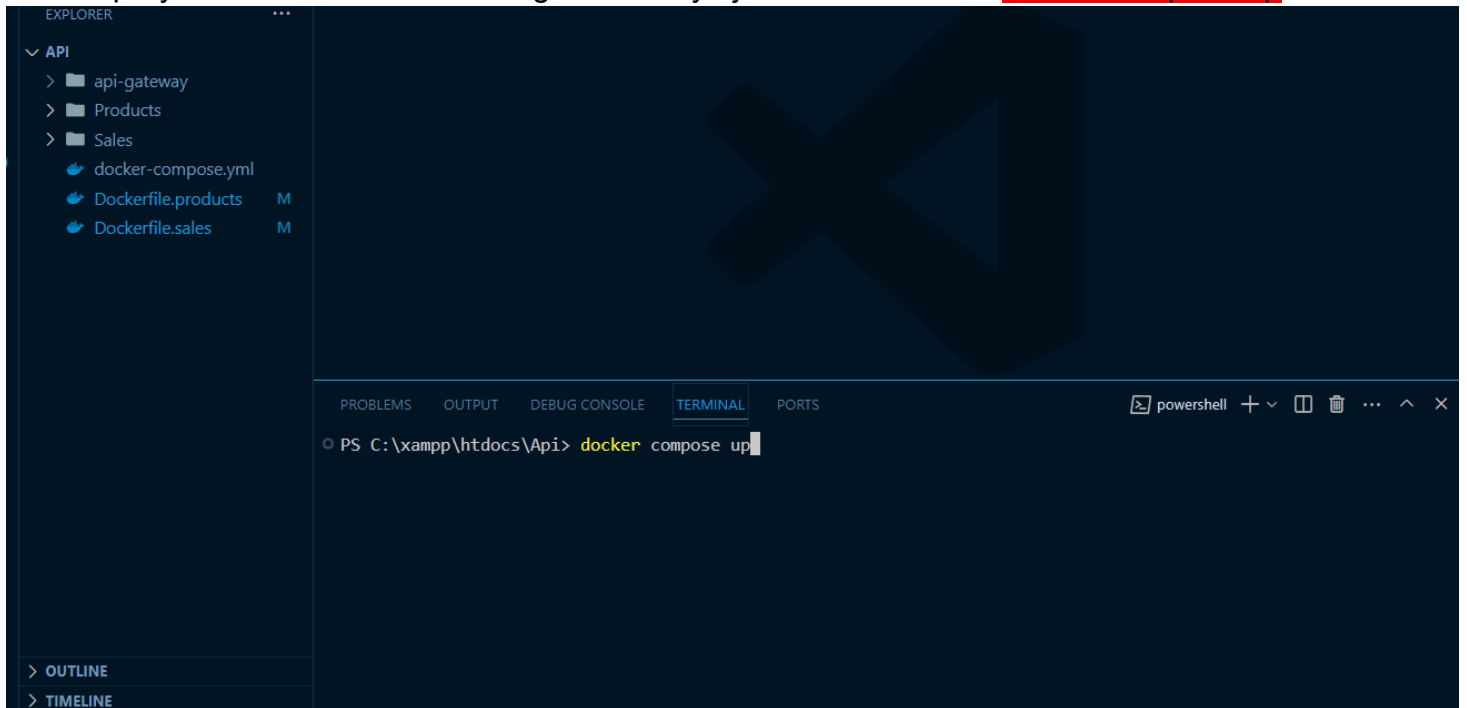


Luego descarga el repositorio de la API en <https://github.com/MaiLear/learning-microservice>:

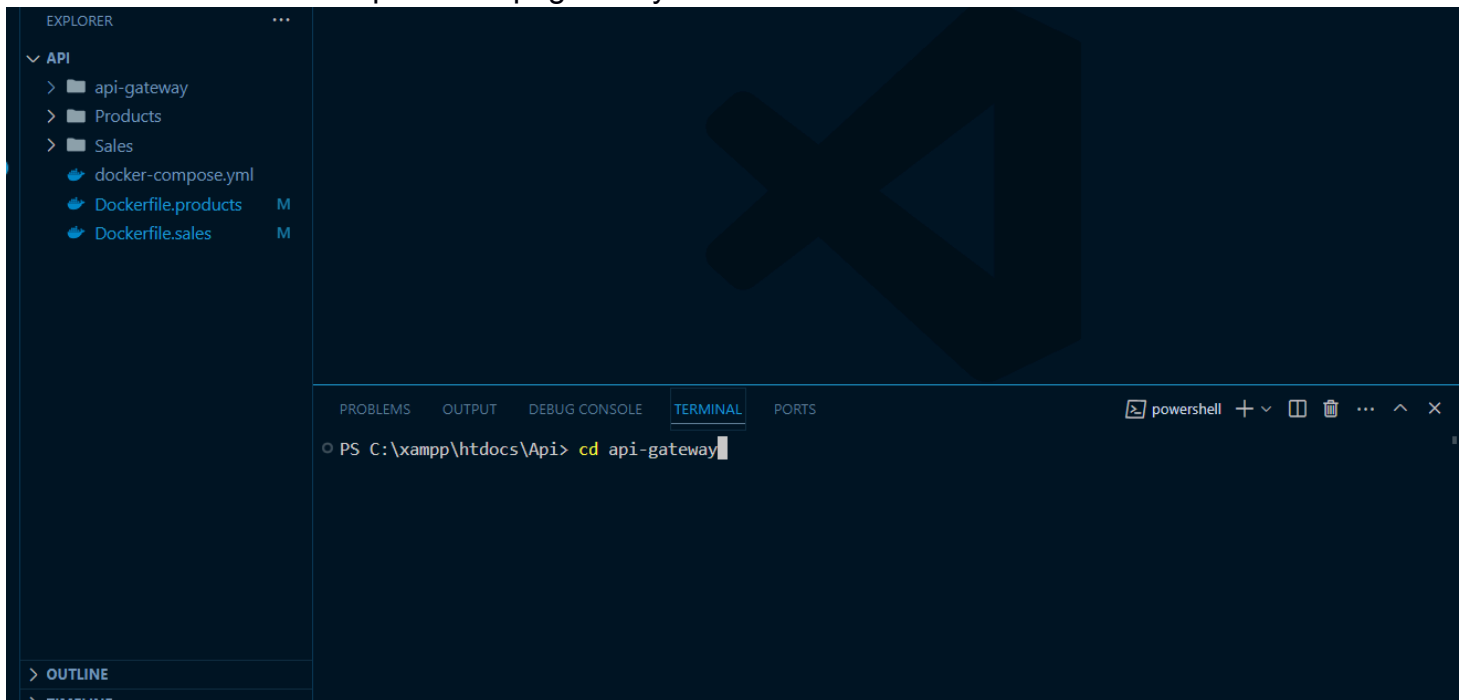


Como usar la API

Abre el proyecto con tu editor de código favorito y ejecuta el comando **docker compose up**



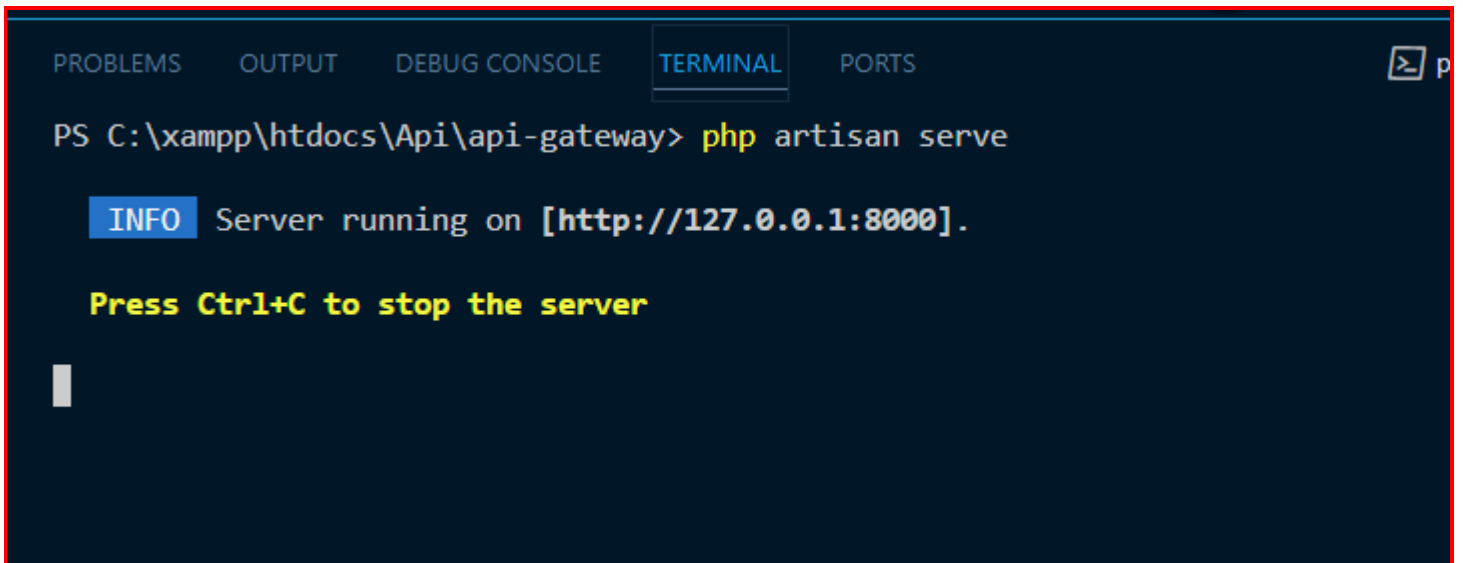
Por último accede a la carpeta del api-gateway en la consola



y ejecuta el comando
php artisan serve

Como usar la API

Puedes acceder a la página principal de la API presionando el link que aparece



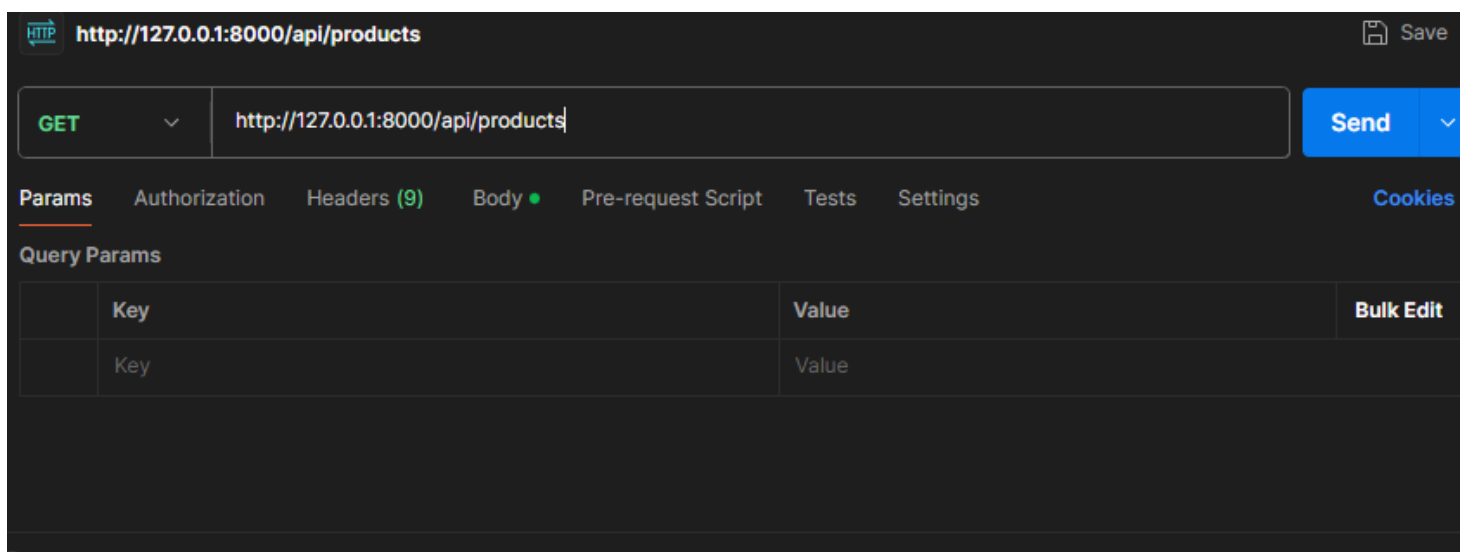
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\xampp\htdocs\Api\api-gateway> php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

Lo siguiente son las formas en las que puedes interactuar con el microservicio de Products y Sales. En este ejemplo utilizo postman para enviar solicitudes a la API, pero lo puedes hacerlo desde un formulario en HTML.

Products

GET

Obtiene una lista de todos los productos



Como usar la API

POST

Si harás la prueba desde un formulario debes crear dos inputs con las propiedades **name** y **value** el primero debe tener el siguiente valor **name="name" value="cualquier texto"** y el segundo **name="price" value="4500 o cualquier numero"**.

The screenshot shows the Postman interface for a POST request to `http://127.0.0.1:8000/api/products`. The request is configured with the following details:

- Method:** POST
- URL:** `http://127.0.0.1:8000/api/products`
- Body Type:** form-data
- Body Data:**

Key	Value
name	chocolate
price	10000

PUT

Si harás la prueba desde un formulario debes crear tres inputs con las propiedades **name** y **value** el primero debe tener el siguiente valor **name="name" value="cualquier texto"**, el segundo **name="price" value="45 o cualquier numero"** y el tercero debe contener **name="method" value="PUT"**.

The screenshot shows the Postman interface for a POST request to `http://127.0.0.1:8000/api/products/2`. The request is configured with the following details:

- Method:** POST
- URL:** `http://127.0.0.1:8000/api/products/2`
- Body Type:** form-data
- Body Data:**

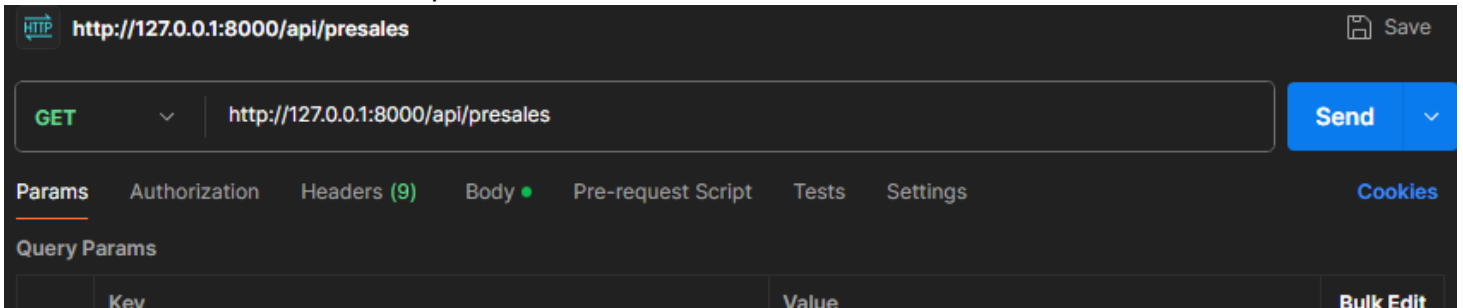
Key	Value
name	chocolate
price	10000
_method	PUT

Como usar la API

PRESALES

GET

Obtiene una lista de todas las pre ventas

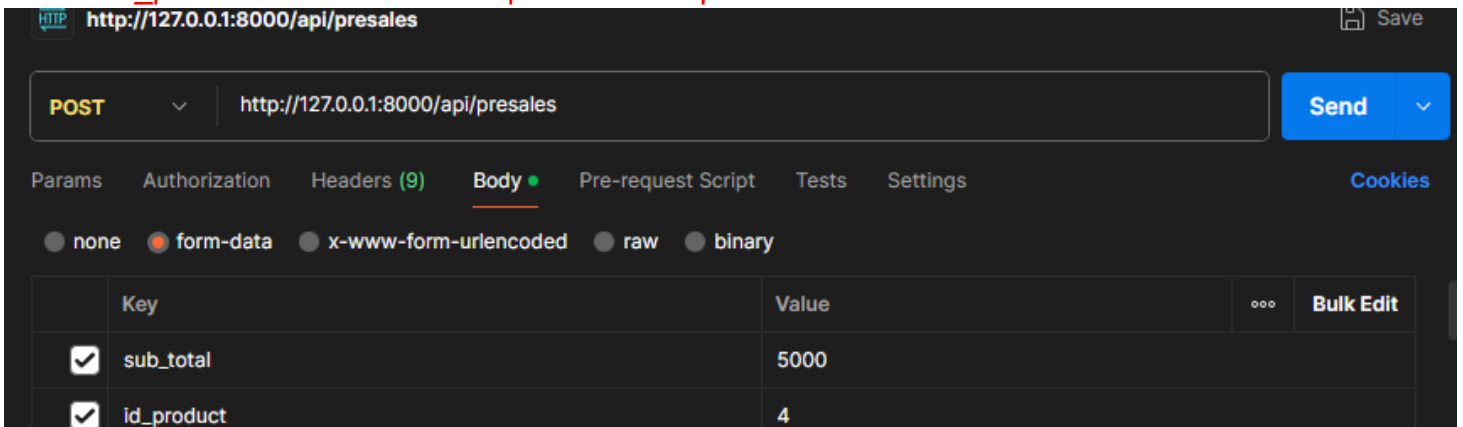


The screenshot shows a REST client interface with the following elements:

- URL bar: `http://127.0.0.1:8000/api/presales`
- Method dropdown: `GET`
- Send button: `Send`
- Tab bar: `Params`, `Authorization`, `Headers (9)`, `Body` (selected), `Pre-request Script`, `Tests`, `Settings`, `Cookies`
- Section: `Query Params`
- Table with 3 columns: `Key`, `Value`, `Bulk Edit`

POST

Si harás la prueba desde un formulario debes crear dos inputs con las propiedades `name` y `value` el primero debe tener el siguiente valor `name="sub_total" value="8 o cualquier número"` y el segundo `name="id_product" value="5 o cualquier id de un producto existente en la base de datos"`.



The screenshot shows a REST client interface with the following elements:

- URL bar: `http://127.0.0.1:8000/api/presales`
- Method dropdown: `POST`
- Send button: `Send`
- Tab bar: `Params`, `Authorization`, `Headers (9)`, `Body` (selected), `Pre-request Script`, `Tests`, `Settings`, `Cookies`
- Body type selection: `none`, `form-data` (selected), `x-www-form-urlencoded`, `raw`, `binary`
- Table with 4 columns: `Key`, `Value`, `...`, `Bulk Edit`

	Key	Value	...	Bulk Edit
<input checked="" type="checkbox"/>	sub_total	5000		
<input checked="" type="checkbox"/>	id_product	4		

Como usar la API

SALES

GET

Obtiene una lista de todas las ventas

The screenshot shows a REST client interface with the URL `http://127.0.0.1:8000/api/sales` and the method `GET`. The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, Settings, and Cookies. The Params tab is active, showing a table for Query Params.

	Key	Value	Bulk Edit
	Key	Value	

POST

Si harás la prueba desde un formulario debes crear un input con las propiedades `name` y `value` con el siguiente valor `name="status"` `value="Vendido o No vendido"`

The screenshot shows a REST client interface with the URL `http://127.0.0.1:8000/api/sales` and the method `POST`. The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, Settings, and Cookies. The Body tab is active, showing the `form-data` option selected. A table for form data is visible.

<input checked="" type="checkbox"/>	status	No vendido
-------------------------------------	--------	------------

Response

PUT

Si harás la prueba desde un formulario debes crear dos inputs con las propiedades `name` y `value` el primero debe tener el siguiente valor `name="status"` `value="Vendido o No vendido"`, y el segundo debe contener `name="method"` `value="PUT"`.

Como usar la API

POST

http://127.0.0.1:8000/api/sales/50

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

	Key	Value	...	Bulk Edit
<input checked="" type="checkbox"/>	status	No vendido		