

Manual de Creación y Prueba de API REST para el Módulo Languages en Laravel

¿Que es una api?(concepto)

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que diferentes aplicaciones de software se comuniquen entre sí, intercambien datos y funciones de manera estandarizada. Funciona como un "puente" o un "contrato" entre dos sistemas, donde una aplicación envía una solicitud a la API y ésta devuelve una respuesta con la información o acción solicitada.

Ejemplo sencillo:

Así como un mesero lleva tu pedido a la cocina y te trae la comida sin que tú entres a cocinar, una API lleva tu solicitud a un sistema y te devuelve la respuesta sin que tengas que entrar al código interno.

¿Para qué sirve?

- **Conectar sistemas:** Permite que distintas aplicaciones compartan datos y funciones.
- **Reutilizar funcionalidades:** No tienes que crear todo desde cero; puedes usar APIs ya hechas (ej: Google Maps, pasarelas de pago).
- **Automatizar procesos:** Tu sistema puede intercambiar datos automáticamente con otros.
- **Escalabilidad:** Facilita integrar servicios nuevos sin reescribir todo el software.

Ejemplo:

Cuando en una app de viajes ves un mapa de Google, esa app está consumiendo la API de Google Maps para mostrar información de ubicación.

¿Cómo consumo una API?

Para “consumir” una API significa usarla desde tu aplicación. Normalmente se hace enviando peticiones HTTP (GET, POST, PUT, DELETE) a las URLs que la API ofrece.

- GET → obtener datos.
- POST → enviar/crear datos.
- PUT/PATCH → actualizar datos(Patch solo actualiza una parte y PUT actualiza todo).
- DELETE → borrar datos.

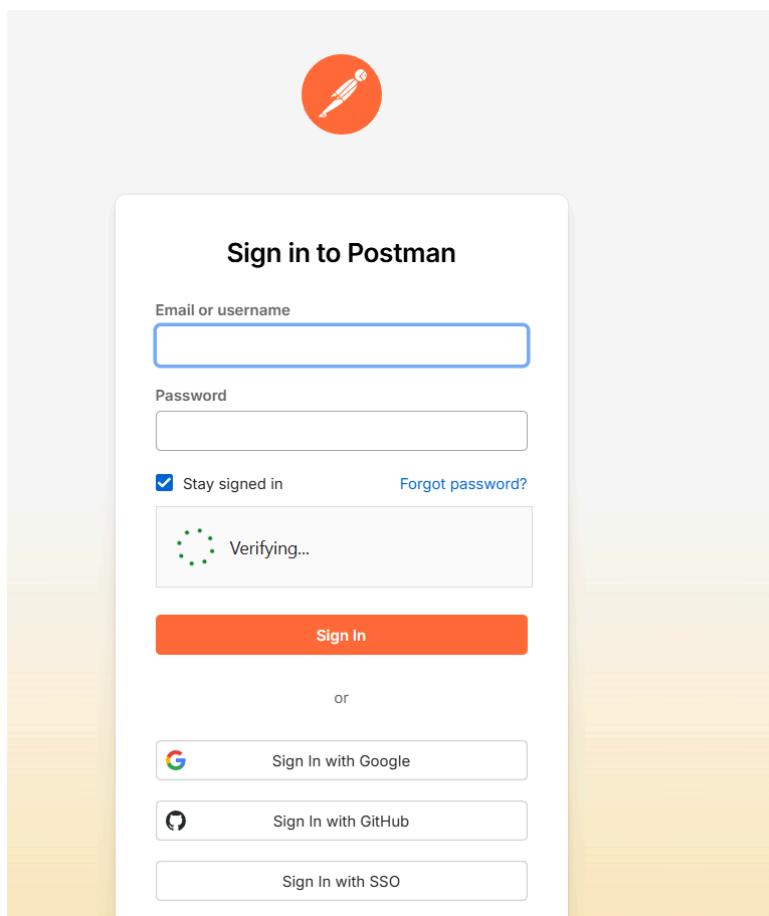
Paso 1:

Descargamos el programa postman desde el siguiente url:

<https://dl.pstmn.io/download/latest/win64>

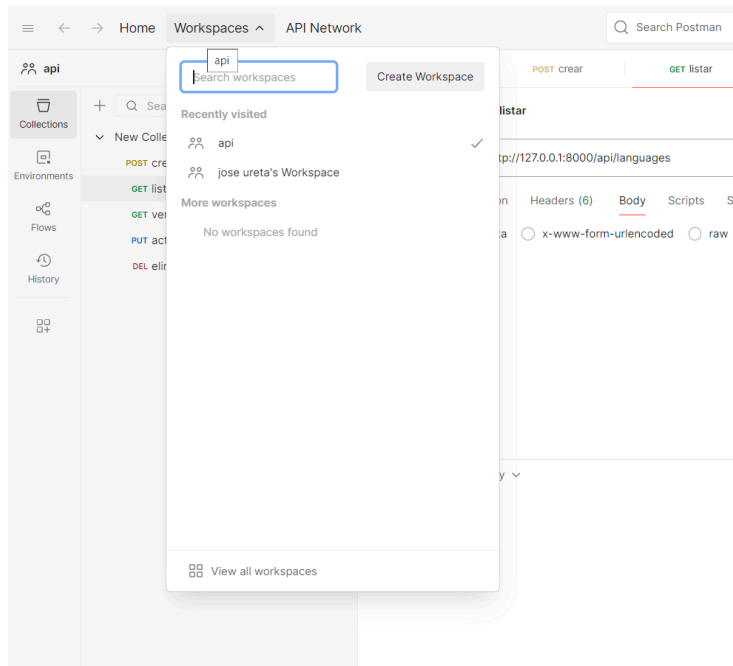
Paso 2:

ahora nos registramos en el postman con nuestro gmail:



The image shows the Postman sign-in interface. At the top is the Postman logo (an orange circle with a white rocket icon). Below it is a white card with the title "Sign in to Postman". The card contains the following elements: a text input field for "Email or username", a text input field for "Password", a checkbox labeled "Stay signed in" which is checked, and a link "Forgot password?". Below these is a "Verifying..." status bar with a green circular progress indicator. At the bottom of the card is an orange "Sign In" button. Below the card, the word "or" is centered. Underneath "or" are three buttons: "Sign In with Google" (with the Google logo), "Sign In with GitHub" (with the GitHub logo), and "Sign In with SSO".

Paso 3: le daremos create workspace para crear nuestro espacio de trabajo



Paso 4: Ahora seleccionamos blank workspace y después le asignaremos un nombre y creamos.

Create your workspace

Get the most out of your workspace with a template.

☒ Blank workspace

Explore our templates

- ☐ API development
- ☐ API testing
- ☐ Incident response
- ☐ Cloud infrastructure management
- ☐ API demos
- ☐ API security
- ☐ Partner Collaboration
- ☐ Public API Network

Cancel

Next

Create your workspace

Name

paco

Select workspace type

- ☒ Internal
Build and test APIs within your team
- ☐ Partner [TRY FOR FREE](#)
Share APIs securely with trusted partners
- ☐ Public
Make APIs accessible to the world

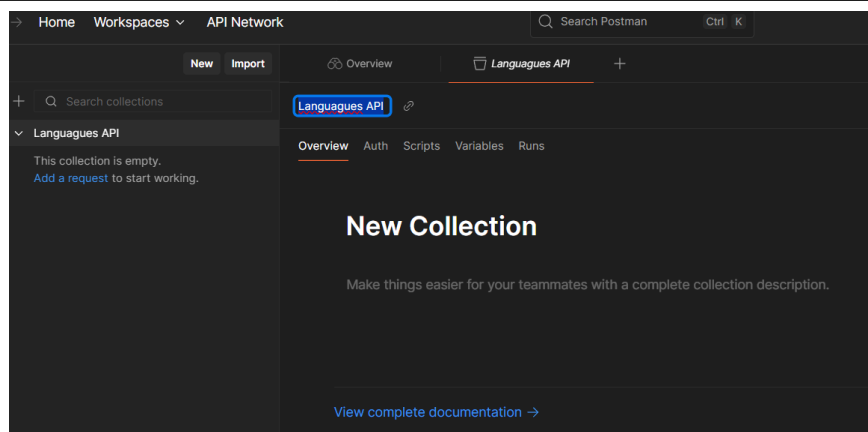
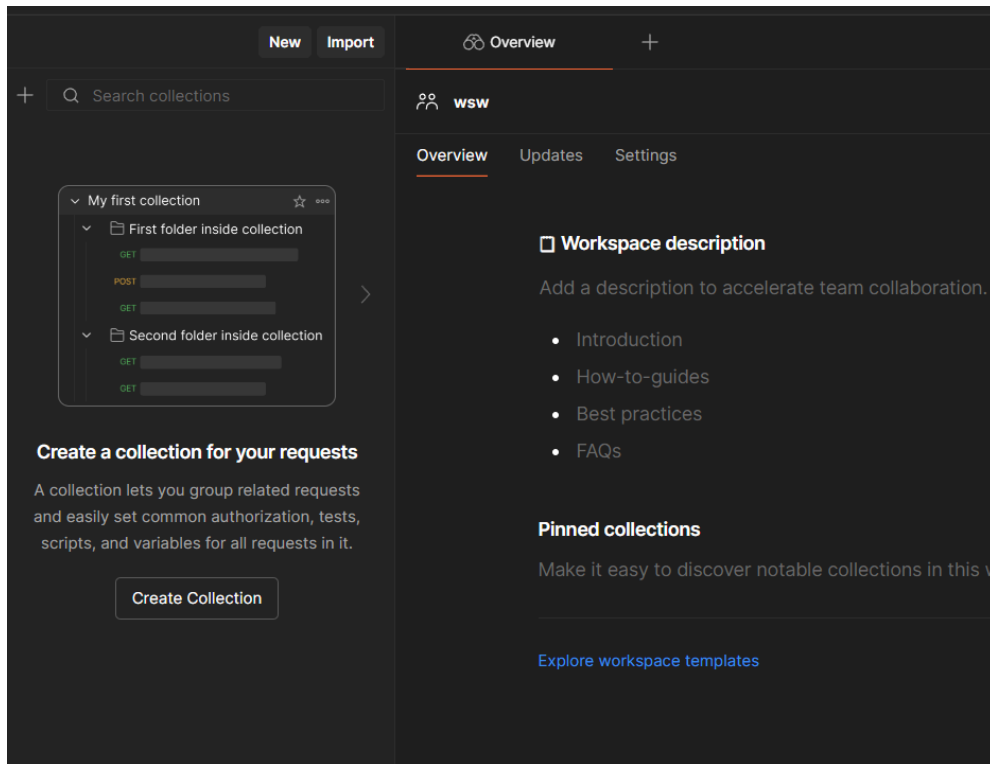
Manage access

- ☒ Everyone in jose ureta's Team
1 member
- ☐ Only you and invited people

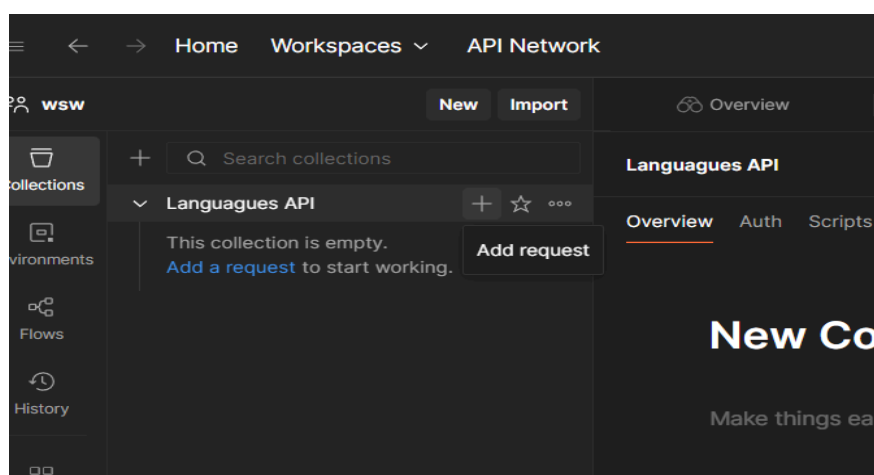
Back

Create

5. Ahora crearemos la colección, y le asignaremos un nombre, ejemplo: Languages API



Paso 6: ahora le daremos en el signo + para añadir un request:

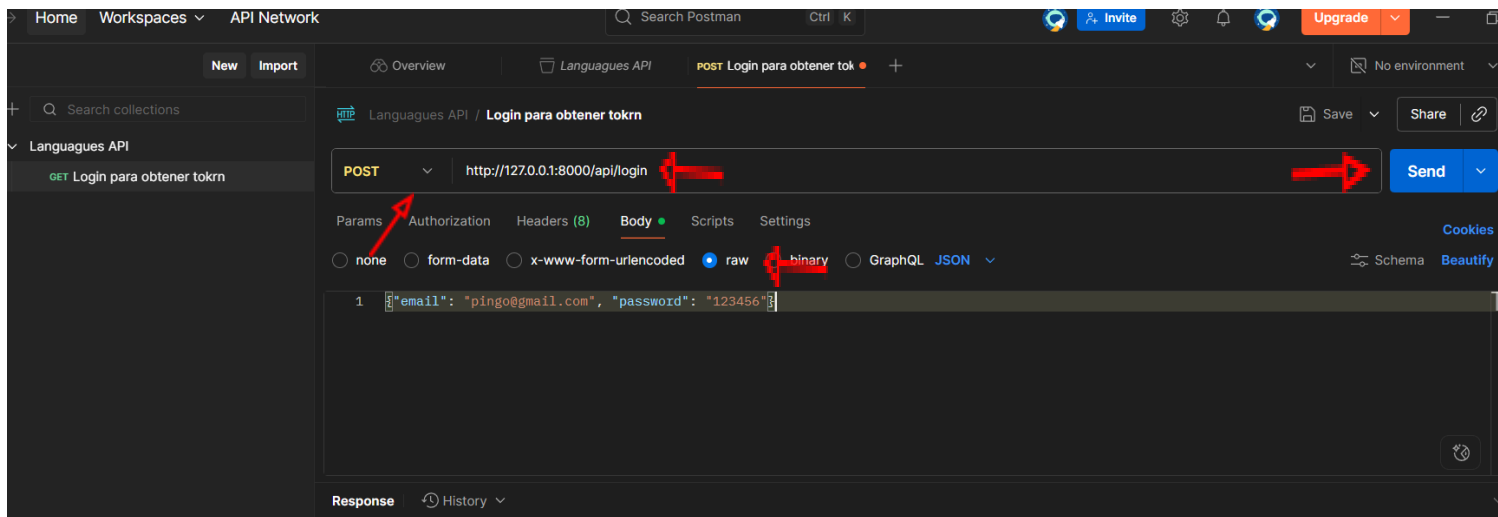


7. REQUEST LOGIN

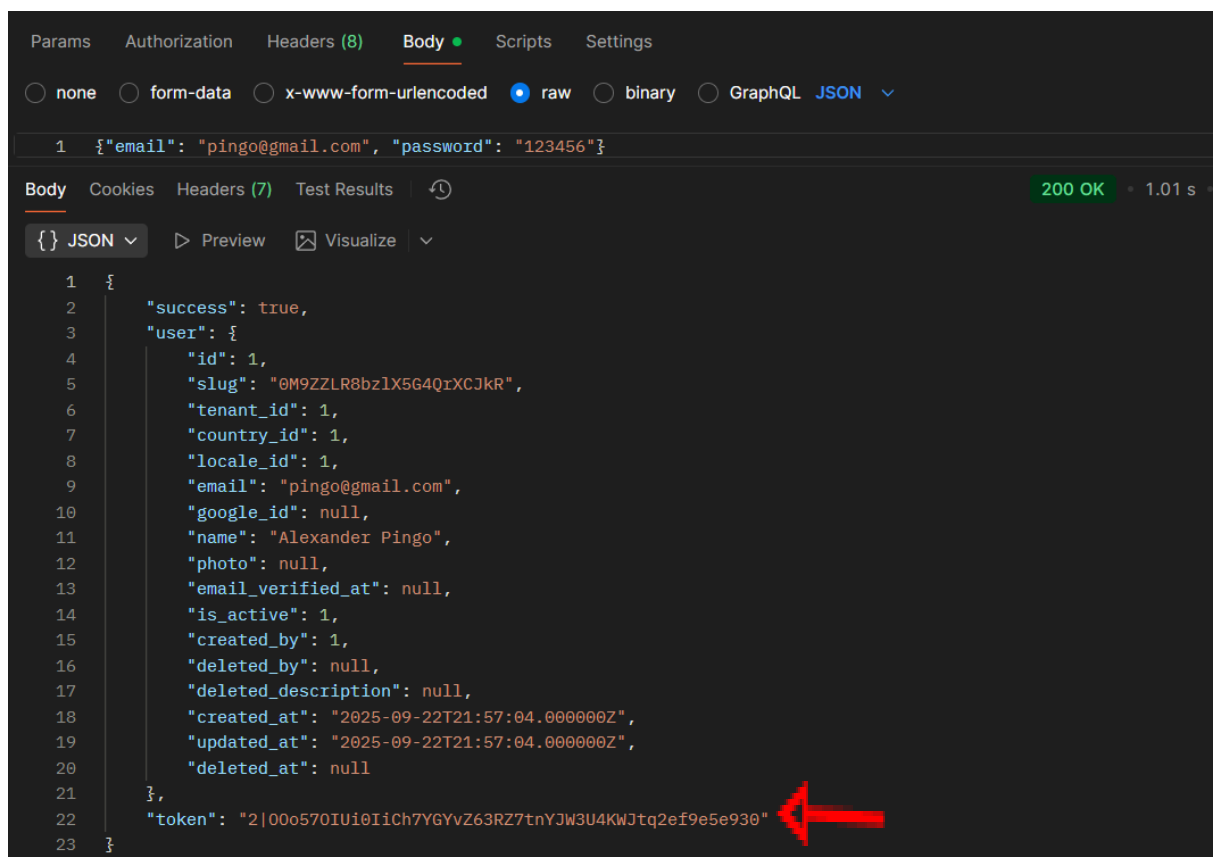
Nombramos el request a **Login para obtener api**, seleccionamos **POST** (sirve para crear un nuevo recurso o información) y en la URL podremos:

<http://127.0.0.1:8000/api/login>.

Ahora seleccionamos Body y raw, ahí es donde pondremos las credenciales del usuario con el que iniciaste sesión en el sistema y después le damos en send



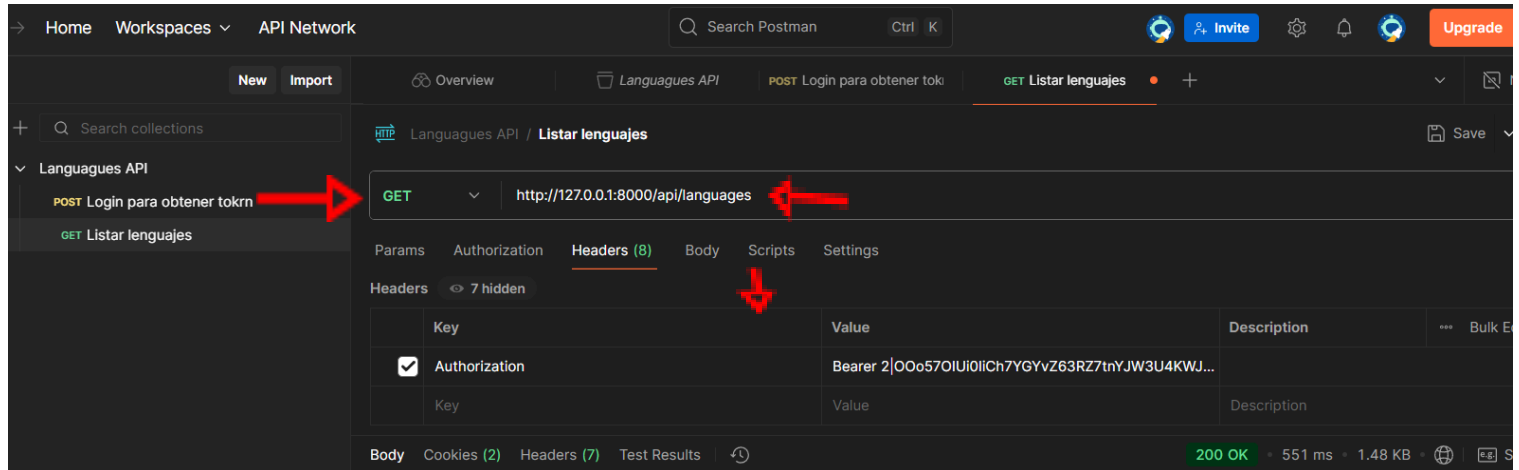
Nos tendría que salir esto junto con el token ya creado, le damos en save (como en todos los request) y seguimos con los demás request.



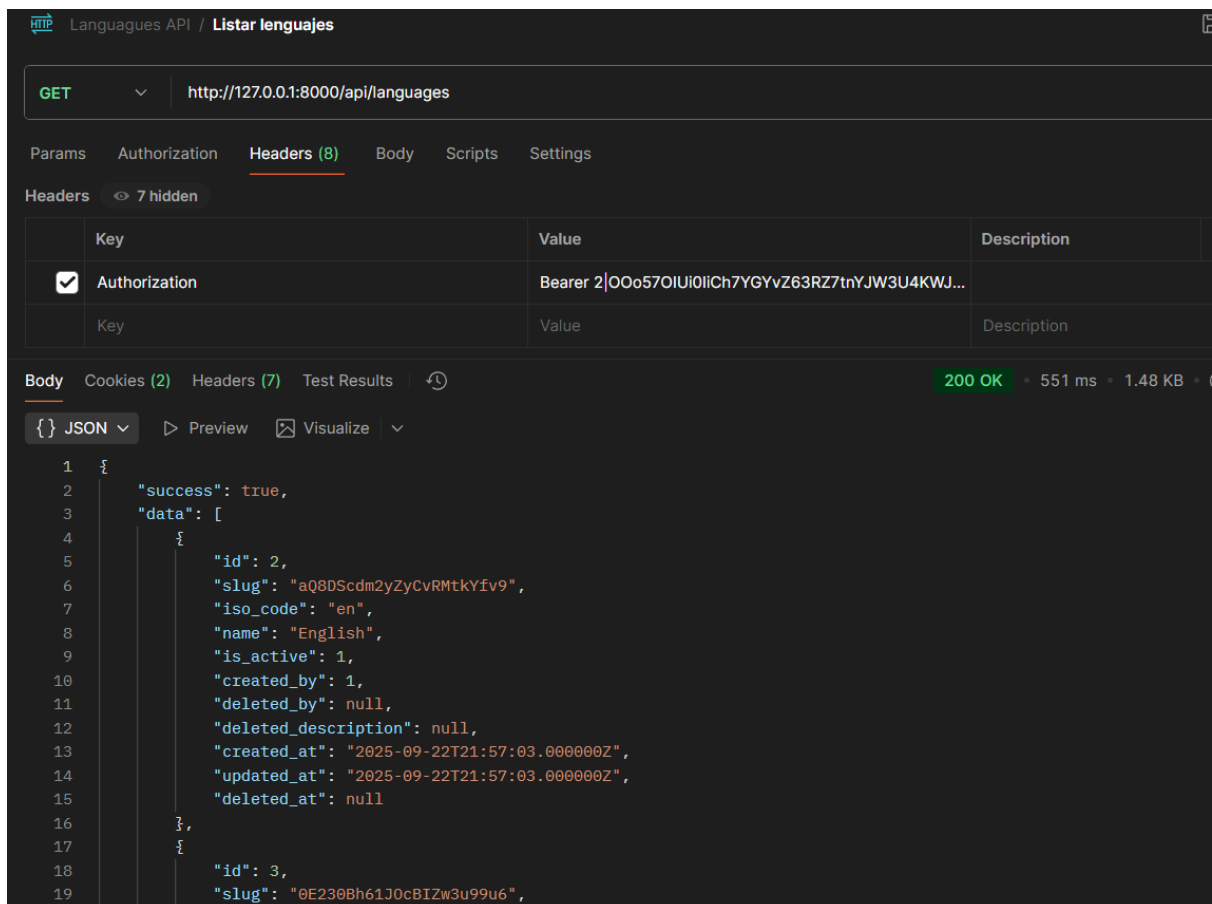
6. REQUEST LISTA LENGUAJES

Como en el request anterior nombramos el request, pero ahora con **GET** (sirve para pedir información) y con la URL: <http://127.0.0.1:8000/api/languages>

Ahora en **Headers** pondremos: **Key: Authorization** y **Value: Bearer (token generado)**



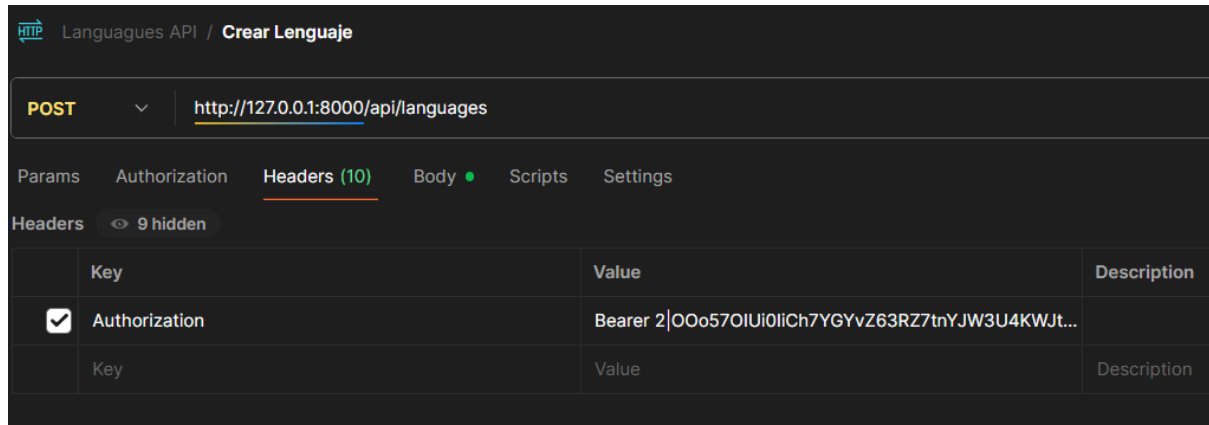
le damos send y tendría que salir la lista de los lenguajes registrados.



7. REQUEST CREAR LENGUAJE

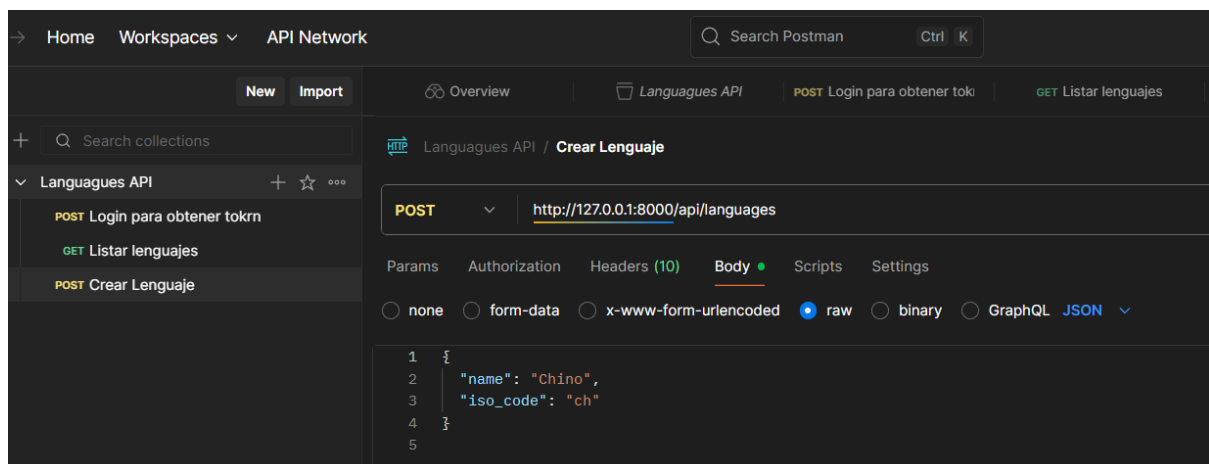
Añadimos y nombramos el request, seleccionamos **POST**(sirve para crear información) y en la URL: <http://127.0.0.1:8000/api/languages>

Ahora en **Headers** pondremos: **Key: Authorization** y **Value: Bearer (token generado)**



En **Body** seleccionamos **raw** y pegaremos el siguiente código, donde pondremos el nombre y su iso code del lenguaje a crear

```
{  
  
  "name": "Chino",  
  
  "iso_code": "ch"  
}
```



send y tendría que salir:

HTTP Languages API / **Crear Lenguaje**

POST http://127.0.0.1:8000/api/languages

Params Authorization Headers (10) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```
1 {
2   "name": "Chino",
3   "iso_code": "ch"
4 }
5
```

Body Cookies (2) Headers (7) Test Results 201 Cr

{ } JSON Preview Visualize

```
1 {
2   "success": true,
3   "message": "Created successfully!",
4   "data": {
5     "name": "Chino",
6     "iso_code": "ch",
7     "created_by": 1,
8     "slug": "OsPDCI2Z6FwA1mJjuGgxmg",
9     "updated_at": "2025-09-23T03:43:54.000000Z",
10    "created_at": "2025-09-23T03:43:54.000000Z",
11    "id": 9
12  }
13 }
```

verificamos:

5	Chino	Activo	
---	-------	--------	--

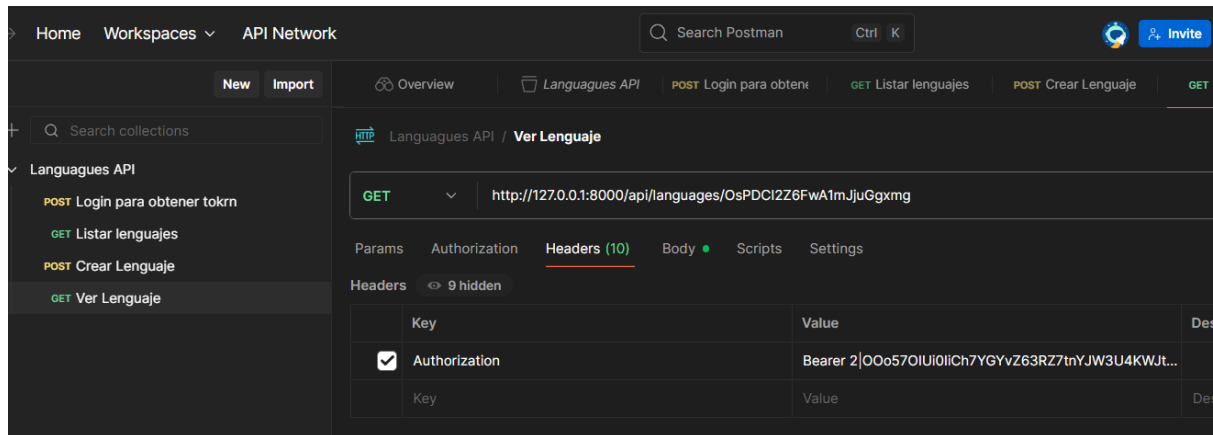
8. REQUEST VER LENGUAJE

Añadimos y nombramos el request, seleccionamos **GET**(sirve para pedir información) y en la URL: <http://127.0.0.1:8000/api/languages/>(pegar slug a consultar de tu tabla)

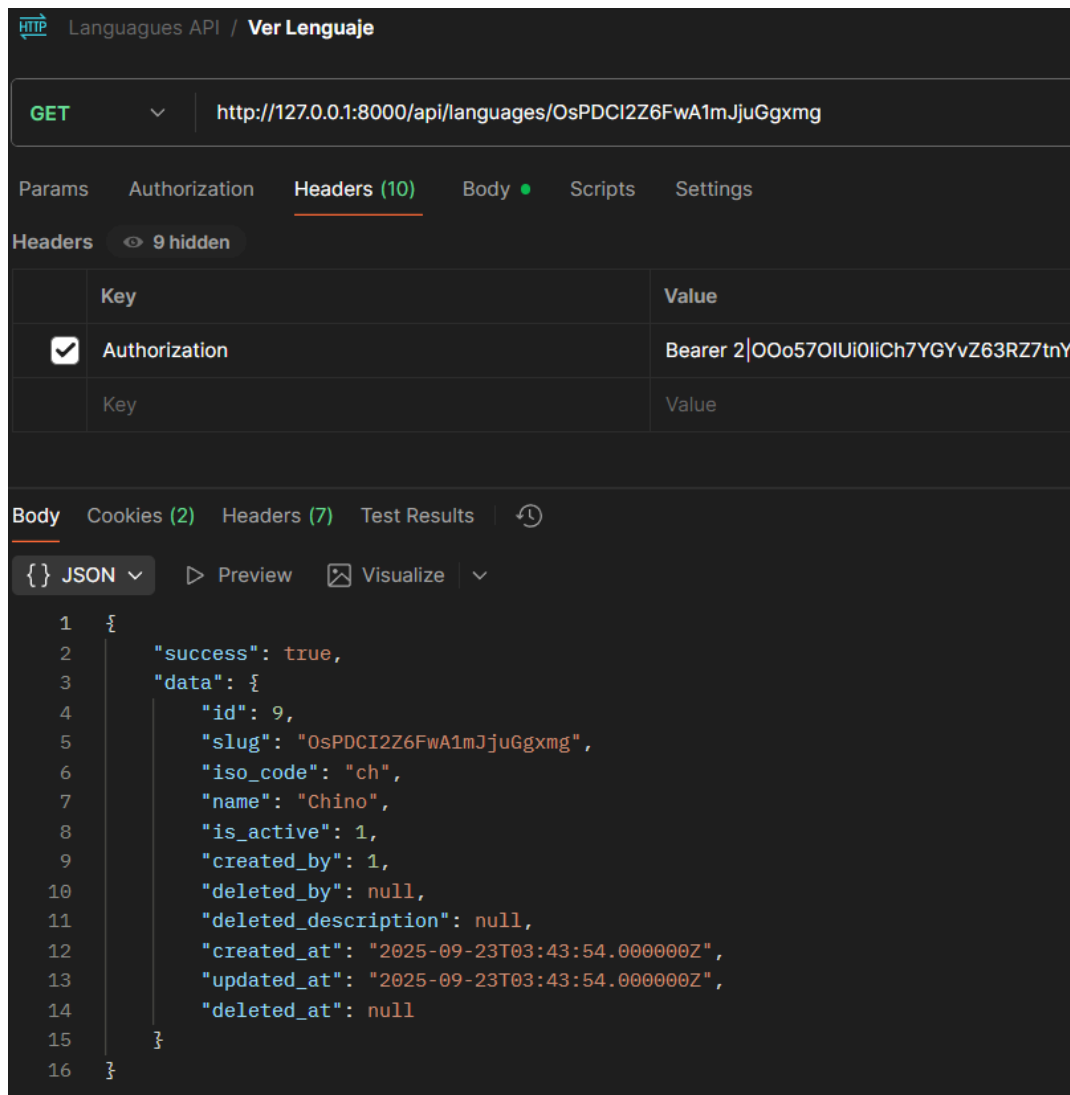
blog_db.languages >> Sigüientes > Mostrar todo > Ordenación

#	id	slug	iso_code	name	is_a
1	1	jnPzZiQEWNyX9PqFzPZzYU	jp	Deutsch	
2	2	aQ8DScdm2yZyCvRMtkYfv9	en	English	
3	3	0E230Bh61JOcBIZw3u99u6	pt	Português	
4	4	yRdI6nXlD2AJwZcMzLjR6P	fr	French	
5	5	sP9T7LV1F7Ic2b8WYRwREE	de	German	
6	6	igx0RaUCSuIQGHn7v0vioa	jp	japanese	
7	7	yGfjJYlcZ5OQE1apf06syb	az	azteco	
8	8	dYR6FdazZlzn8fG4wokBbJ	ai	aymara	
9	9	OsPDCI2Z6FwA1mJjuGgxmg	ch	Chino	

Ahora en **Headers** pondremos: **Key: Authorization** y **Value: Bearer (token generado)**



send y tiene que dar solo información de del slug consultado:



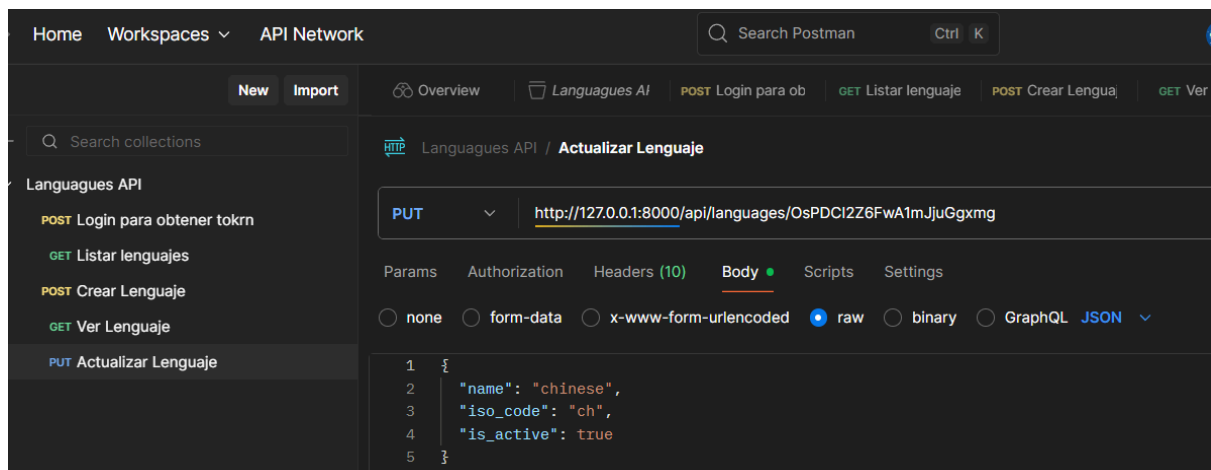
10. REQUEST ACTUALIZAR LENGUAJE

Añadimos y nombramos el request, seleccionamos **PUT**(sirve para actualizar información) y en la URL: <http://127.0.0.1:8000/api/languages/>(pegar slug a actualizar de tu tabla)

blog_db.languages		» Siguietes		Mostrar todo	▼ Ordenación		
#	id	slug	iso_code	name	is_a		
1	1	jnPzZiQEWNyX9PqFzPZzYU	jp	Deutsch			
2	2	aQ8DScdm2yZyCvRMtkYfv9	en	English			
3	3	0E230Bh61JOcBIZw3u99u6	pt	Português			
4	4	yRdI6nXID2AJwZcMzLjR6P	fr	French			
5	5	sP9T7LV1F7Ic2b8WYRwREE	de	German			
6	6	igx0RaUCSuIQGHn7v0vioa	jp	japanese			
7	7	yGfjJYlcZ5OQE1apf06syb	az	azteco			
8	8	dYR6FdazZlzn8fG4wokBbJ	ai	aymara			
9	9	OsPDCI2Z6FwA1mJjuGgxmg	ch	Chino			

Ahora en **Headers** pondremos: **Key: Authorization** y **Value: Bearer (token generado)** y en **Body** seleccionamos **raw** y ponemos el siguiente codigo (donde actualizaremos el nombre e iso_code):

```
{  
  
  "name": "chinese",  
  
  "iso_code": "ch",  
  
  "is_active": true  
}
```






send y tendría que salir la confirmación de actualización de lenguaje:

```
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 {
2   "name": "chinese",
3   "iso_code": "ch",
4   "is_active": true
5 }
```

```
body Cookies (2) Headers (7) Test Results
{ } JSON Preview Visualize
1 {
2   "success": true,
3   "message": "Updated successfully!",
4   "data": {
5     "id": 9,
6     "slug": "OsPDCI2Z6FwA1mJjuGgxmg",
7     "iso_code": "ch",
8     "name": "chinese",
9     "is_active": true,
10    "created_by": 1,
11    "deleted_by": null,
12    "deleted_description": null,
13    "created_at": "2025-09-23T03:43:54.000000Z",
14    "updated_at": "2025-09-23T04:14:26.000000Z",
15    "deleted_at": null
16  }
17 }
```

Verificar:

5	chinese	Activo	  
---	---------	--------	---

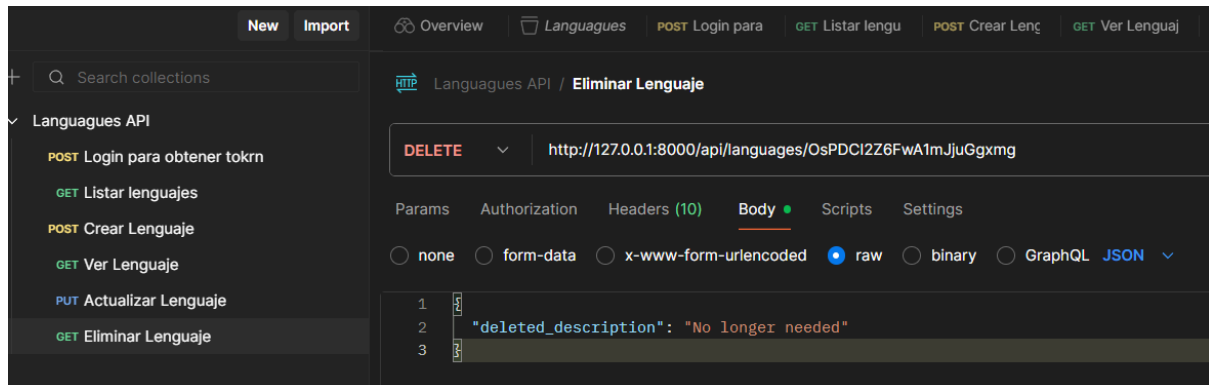
11. REQUEST ELIMINAR LENGUAJE

Añadimos y nombramos el request, seleccionamos **DELETE**(sirve para eliminar información) y en la URL: <http://127.0.0.1:8000/api/languages/>(pegar slug a eliminar de tu tabla)

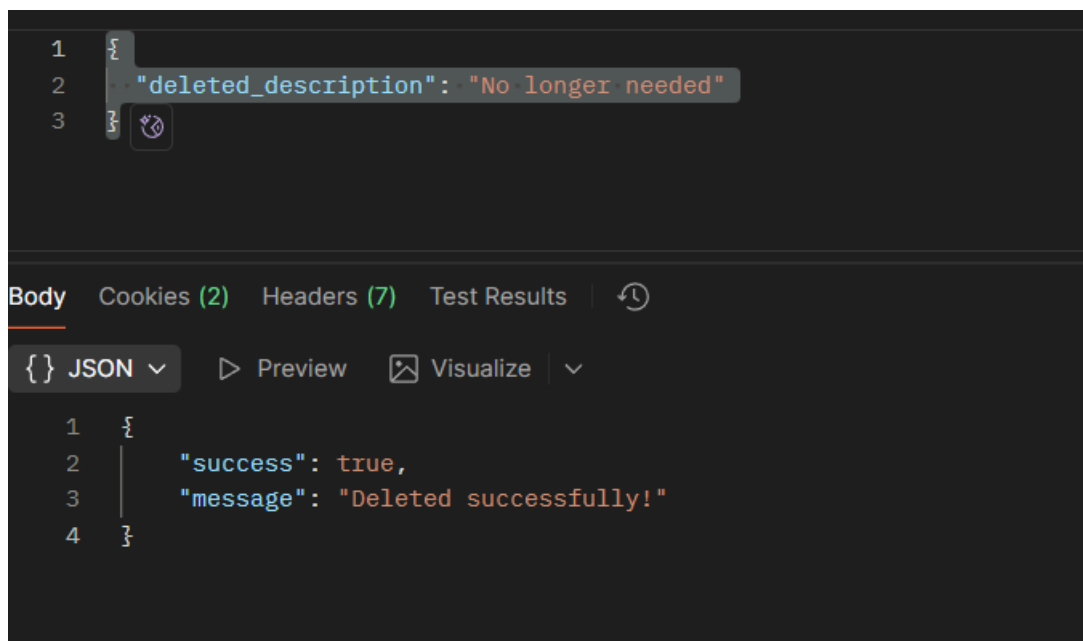
blog_db.languages		»» Siguientes		Mostrar todo		Ordenación	
#	id	slug	iso_code	name	is_a		
1	1	jnPzZiQEWNyX9PqFzPZzYU	jp	Deutsch			
2	2	aQ8DScdm2yZyCvRMtkYfv9	en	English			
3	3	0E2308h61JOcBIZw3u99u6	pt	Português			
4	4	yRdI6nXID2AJwZcMzLjR6P	fr	French			
5	5	sP9T7LV1F7Ic2b8WYRwREE	de	German			
6	6	igx0RaUCSuIQGHn7v0vioa	jp	japanese			
7	7	yGfjJYlcZ5OQE1apf06syb	az	azteco			
8	8	dYR6FdazZIZn8fG4wok8BbJ	ai	aymara			
9	9	OsPDCI2Z6FwA1mJjuGgxmg	ch	Chino			

Ahora en **Headers** pondremos: **Key: Authorization** y **Value: Bearer (token generado)** y en **Body** seleccionamos **raw** y ponemos el siguiente código:

```
{  
  
  "deleted_description": "No longer needed"  
  
}
```



por último al dar en **send** tendría que dar la confirmación:



Verificamos:

