



POSTMAN

POSTMAN



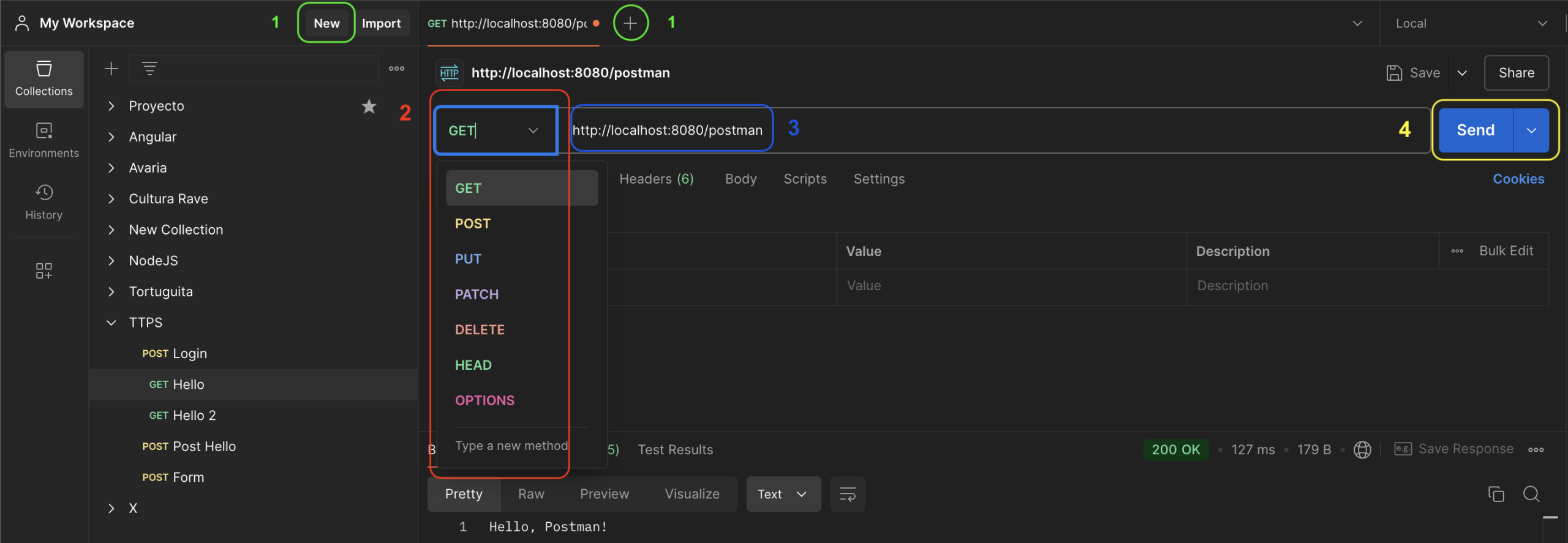
Postman es una herramienta que nos permite diseñar, probar y documentar nuestra API Rest.



Tiene una versión desktop para Windows, Linux o Mac y una versión web pero que no cuenta con la totalidad de las funciones.



Se puede descargar en <https://www.postman.com/downloads/>. Desde las últimas versiones es necesario tener una cuenta e iniciar sesión para utilizar la mayoría de las funcionalidades.



HTTP Request

- Se puede crear y ejecutar rápidamente una request.
- Se debe especificar el método HTTP y el endpoint.

HTTP Request – GET – Query Params

The screenshot shows the Postman interface for a GET request. The URL bar contains `{{TTPS}} /postman?firstName=Lourdes`. The 'Params' tab is selected, showing a table of query parameters. The 'Send' button is highlighted in blue, and the 'Send and Download' button is visible below it. The 'Body' tab is selected at the bottom, showing the response 'Hello, Postman!'.

GET `{{TTPS}} /postman?firstName=Lourdes`

Params • Authorization Headers (6) Body Scripts Settings

Query Params

<input type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	firstName	Lourdes			
<input type="checkbox"/>	lastName	Valli	mi apellido		
	Key	Value	Description		

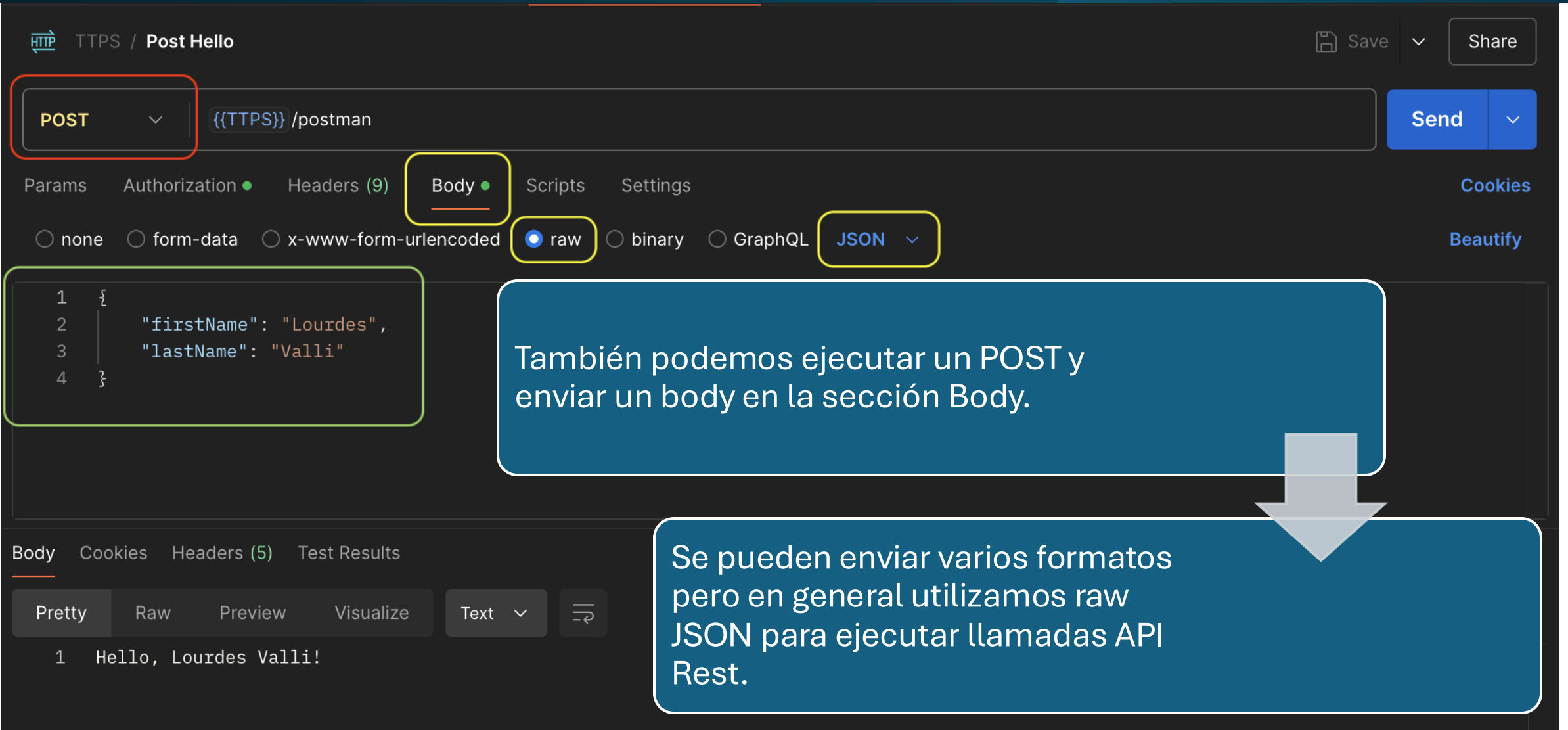
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text ▼

1 Hello, Postman!

- Para agregar query parameters a una request, se pueden escribir directamente en la URL o definir en la pestaña Params.
- Se puede agregar una descripción o deshabilitarlos para ciertas ejecuciones.
- Al ejecutar la request podemos especificarle a Postman que queremos descargar el resultado.

HTTP Request – POST – Body



The screenshot shows the Postman interface for a POST request. The URL is `{{TTPS}} /postman`. The request type is **POST**. The **Body** tab is selected, and the format is **raw** with **JSON** selected. The body content is a JSON object:

```
{  "firstName": "Lourdes",  "lastName": "Valli"}
```

. The response is displayed in the bottom panel, showing `Hello, Lourdes Valli!`. Annotations include a red box around the **POST** method, a yellow box around the **Body** tab and **raw** format, and a green box around the JSON body content.

También podemos ejecutar un POST y enviar un body en la sección Body.

Se pueden enviar varios formatos pero en general utilizamos raw JSON para ejecutar llamadas API Rest.

HTTP Request – POST – Form

The screenshot shows the Postman interface for a POST request to the endpoint `[[TTPS]] /postman/form`. The request body is configured as `form-data` with three fields: `firstName` (text, value: Lourdes), `lastName` (text, value: Valli), and `avatar` (file, value: archivo.jpg). The response is a 200 OK status with a response time of 3.08 s and a size of 32.86 KB. The response body is displayed in JSON format, showing the received data structure.

POST `[[TTPS]] /postman/form` Send

Params Authorization Headers (8) **Body** Scripts Settings Cookies

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

	Key		Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	firstName	Text	Lourdes			
<input checked="" type="checkbox"/>	lastName	Text	Valli			
<input checked="" type="checkbox"/>	avatar	File	archivo.jpg			
	Key	Text	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 3.08 s • 32.86 KB • Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "firstName": "Lourdes",
3   "lastName": "Valli",
4   "avatar": {
5     "contentType": "image/jpeg",
6     "name": "avatar",
7     "bytes": "/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAAGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwckDcpLDxNDQ0Hyc5PTgyPC4zN
8     "empty": false,
9     "size": 24928,
10    "inputStream"
11  }
```

- Algunas veces podemos querer testear un endpoint que recibe un formulario, Postman tiene una opción que nos permite enviar los campos en este formato.

My Workspace

New

Import

GET http://localhost:

POST Form

POST Post Hello

Local

TTPS

POST Login

Local

Collections

Environments

History

Avaria

users

Carpeta users que pertenece a la colección Avaria

GET findAllUsers

GET findUserById

GET findUserServicesByUserId

POST createUser

PUT updateUser

DEL removeUserById

DEL removeAllUsers

POST login

POST token

GET findUserEventsByUserId

POST forgotPassword

services

events

Cultura Rave

TTPS

Save

Run

Fork

0

Share

Overview

Authorization

Scripts

Variables

Runs

These variables are specific to this collection and its requests. Learn more about [collection variables](#)

Filter variables

	Variable	Initial value	Current value	
<input checked="" type="checkbox"/>	collectionToken		ey123456	
<input checked="" type="checkbox"/>	otherVar	unValor	unValor	
	Add new variable			

Collections

- La utilización de colecciones es útil cuando tenemos varios endpoints que probar.
- Podemos guardar, nombrar y agrupar las distintas requests.
- En una colección se pueden definir variables, métodos de autenticación y scripts.

My Workspace

NewImport

Collections

Environments

History

Globals

Local

Production

Local

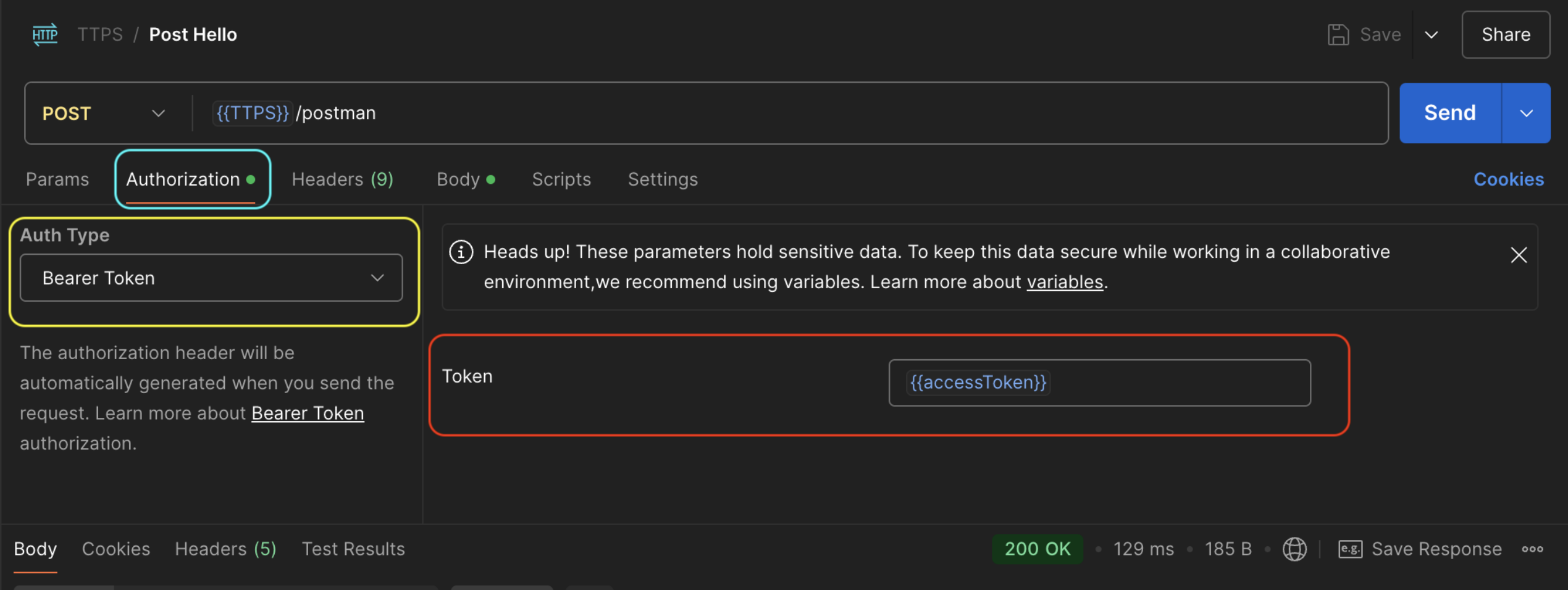
SaveFork0Share

Filter variables

	Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/>	URL	default	http://localhost:5000	http://localhost:5000
<input checked="" type="checkbox"/>	AVARIA_URL	default	http://localhost:8080/avaria	http://localhost:8080/avaria
<input checked="" type="checkbox"/>	CR_token	secret	
<input checked="" type="checkbox"/>	CR_URL	default	http://localhost:8080	http://localhost:8080
<input checked="" type="checkbox"/>	accessToken	secret	
<input checked="" type="checkbox"/>	TTPS	default	http://localhost:8080	http://localhost:8080
	Add new variable			

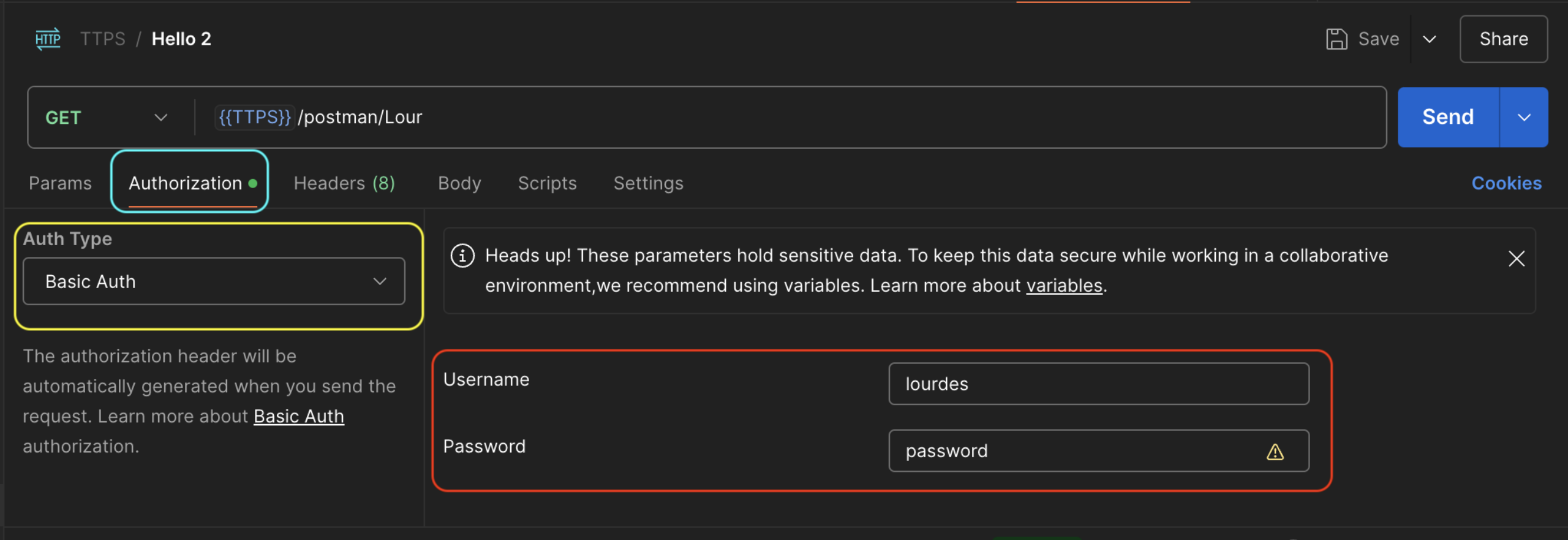
Environments

- Se pueden definir environments para guardar variables que varían por ambiente, por ejemplo cuando queremos ejecutar request localmente o contra un servidor externo.
- Las variables se pueden definir como secret para que nos oculte los valores.



Authentication – Bearer Token

- Podemos cargar un método de autenticación.
- Postman nos ofrece muchas opciones pero una muy utilizada es el Bearer token.
- Para utilizar una variable definida en el environment o en la colección actual, escribimos el nombre entre {{ }}.



Authentication – Basic Auth

- Otro método que también puede ser bastante común también es la autenticación básica por medio de usuario y contraseña.

The screenshot shows the Postman interface for a POST request to `https://postman/login`. The **Scripts** tab is active, displaying the following JavaScript code:

```
1 var jsonData = pm.response.json();
2 pm.environment.set("accessToken", jsonData.token);
3 pm.collectionVariables.set("collectionToken", jsonData.token);
4 pm.globals.set("globalToken", jsonData.token);
```

Below the script, the **Body** tab shows the response in JSON format:

```
1 {
2   "token": "eyJ23456"
3 }
```

At the top right, there are buttons for **Save** and **Share**. On the right side, there are tabs for **Params**, **Authorization**, **Headers (7)**, **Body**, **Scripts** (selected), and **Settings**. Below these are **Pre-request** and **Post-response** script sections. On the far right, there is a **Cookies** tab and a **Send** button. At the bottom right, the status bar shows **200 OK**, **123 ms**, **184 B**, and a **Save Response** button.

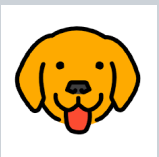
Scripts

- Otra funcionalidad muy útil son los scripts. Nos permiten ejecutar código javascript antes o después de la ejecución de una request.
- En general utilizamos el lenguaje de consulta JsonPath para acceder por ejemplo a valores dentro de la respuesta y utilizarlos más tarde.
- De esta forma podemos obtener un token de una respuesta y guardarlo en una variable.
- Se utiliza **pm.environment**, **pm.collectionVariables** y **pm.globals** para acceder a las variables definidas en un ambiente, en la colección actual o globales respectivamente.

Alternativas



Insomnia: <https://insomnia.rest/>



Bruno: <https://www.usebruno.com/>

¿Preguntas?