

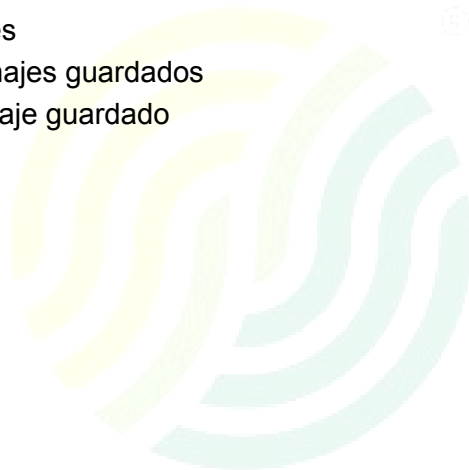


# **Documentación Prueba - El Jornalero**

**Miguel Hurtado Mesa**

# Índice

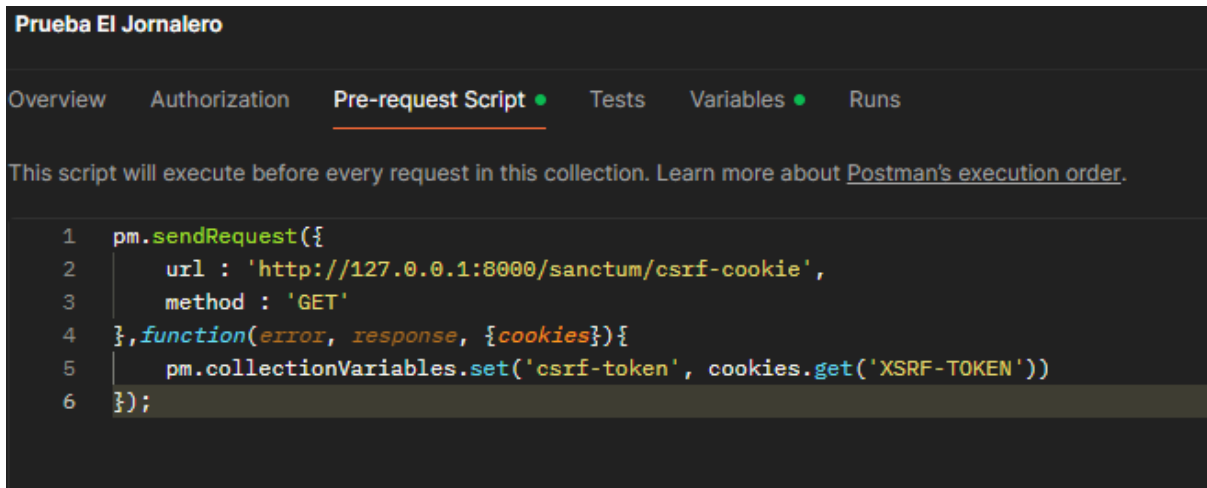
<b>Configuración de PostMan</b>	<b>3</b>
1. Configurar Pre-request Script	3
2. Headers en las peticiones	3
<b>Api en Laravel</b>	<b>4</b>
1. Autenticación	4
- Registro	4
- Login	4
- Obtener Usuario	5
- Logout	5
2. Obtener Personajes	6
- Obtener todos los personajes	6
-Ejemplo de paginación	6
-Ejemplo de filtro	7
- Obtener un personaje por su ID	8
3. Guardar Personajes	8
- Guardar Personajes	8
- Obtener los personajes guardados	9
- Eliminar un personaje guardado	9



# Configuración de PostMan

Para poder hacer las peticiones a nuestra Api de Laravel, primero crearemos una nueva colección en postman y deberemos hacer lo siguiente:

## 1. Configurar Pre-request Script



The screenshot shows the Postman interface with the 'Pre-request Script' tab selected. The script is a JavaScript function that sends a GET request to 'http://127.0.0.1:8000/sanctum/csrf-cookie' and then sets a collection variable 'csrf-token' to the value of the 'XSRF-TOKEN' cookie from the response.

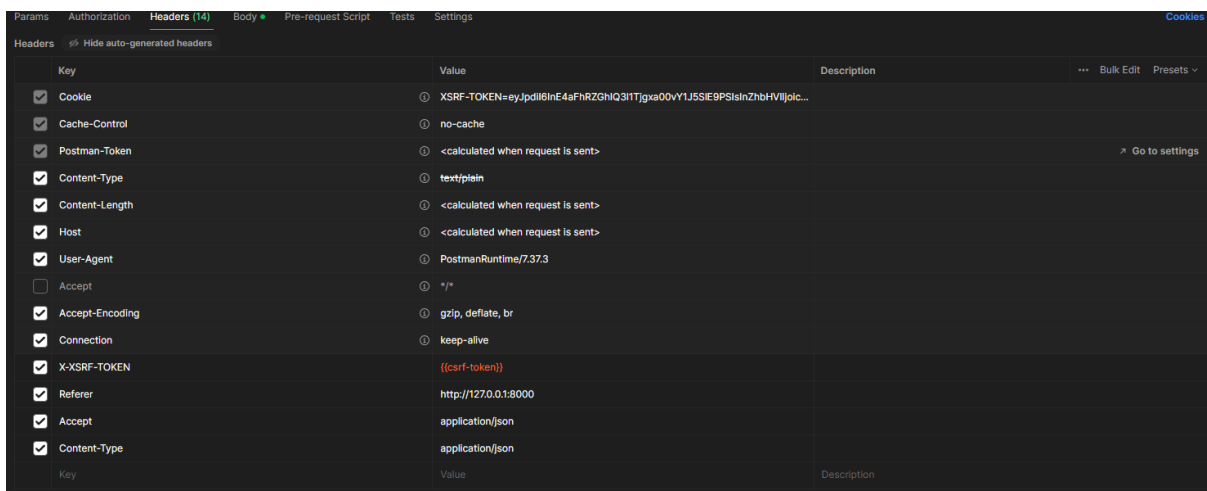
```
1 pm.sendRequest({
2   url : 'http://127.0.0.1:8000/sanctum/csrf-cookie',
3   method : 'GET'
4 },function(error, response, {cookies}){
5   pm.collectionVariables.set('csrf-token', cookies.get('XSRF-TOKEN'))
6 });
```

Este Script nos hace una petición para obtener el **Token csrf** y nos lo guarda en una variable de la colección llamada '**csrf-token**'.

## 2. Headers en las peticiones

Una vez configurado el Pre-request Script de nuestra colección deberemos crear una nueva petición.

Una vez creada, deberemos aplicar los siguientes **Headers**:



The screenshot shows the 'Headers' tab in Postman with a list of headers. The 'X-XSRF-TOKEN' header is highlighted in red, indicating it is the focus of the configuration.

Key	Value	Description
<input checked="" type="checkbox"/> Cookie	XSRF-TOKEN=eyJpdll6InE4aFhrZGhiQ3I1TjgwO0V1J5SIE9PSIsInZhbHVlIjoic...	
<input checked="" type="checkbox"/> Cache-Control	no-cache	
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Content-Type	text/plain	
<input checked="" type="checkbox"/> Content-Length	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.37.3	
<input type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> X-XSRF-TOKEN	{{csrf-token}}	
<input checked="" type="checkbox"/> Referer	http://127.0.0.1:8000	
<input checked="" type="checkbox"/> Accept	application/json	
<input checked="" type="checkbox"/> Content-Type	application/json	

# Api en Laravel

Una vez configurado PostMan podemos empezar a usar la **API en Laravel**.

## 1. Autenticación:

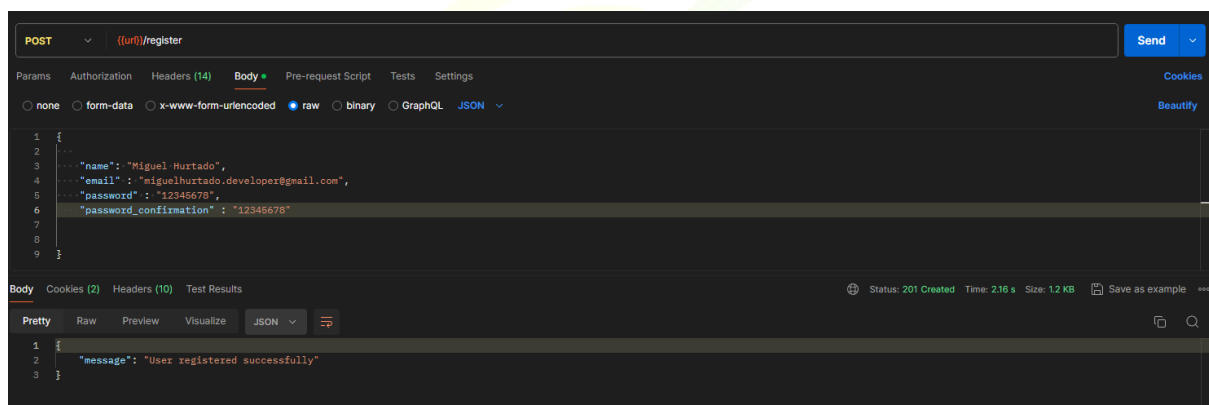
Para realizar la parte de **autenticación** de la API he utilizado el paquete **Breeze** para Laravel:

Documentación de Breeze: <https://laravel.com/docs/11.x/starter-kits#breeze-and-next>

### - Registro:

Lo primero que vamos a ver es el registro de un nuevo Usuario:

Url para el Registro: **POST** `{{url}}/register`

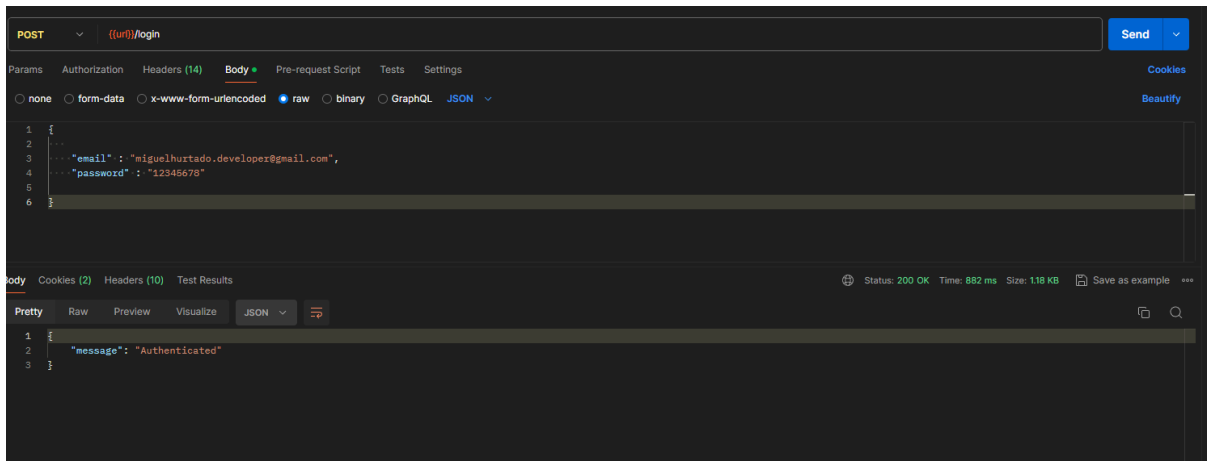


Al ejecutar la petición le pasamos los datos necesarios para el registro del usuario que son: Name, email, password y password\_confirmation. Y obtendremos la respuesta de que el usuario se ha registrado correctamente y automáticamente se iniciará sesión.

### - Login:

Una vez tengamos un usuario registrado, podremos Iniciar Sesión en nuestra API:

Url para el Login : **POST** `{{url}}/login`

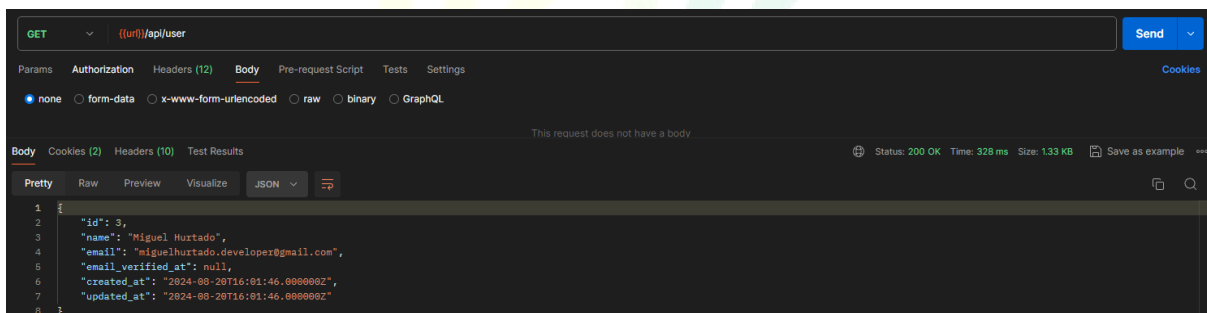


Cuando ejecutemos la petición de Login, debemos pasarle los parámetros email y password. Obtendremos la respuesta de que estamos autenticados.

## - Obtener Usuario:

Cuando tengamos la sesión iniciada, podremos ver los datos del usuario:

Url para obtener los datos del usuario : **GET** {{url}}/api/user

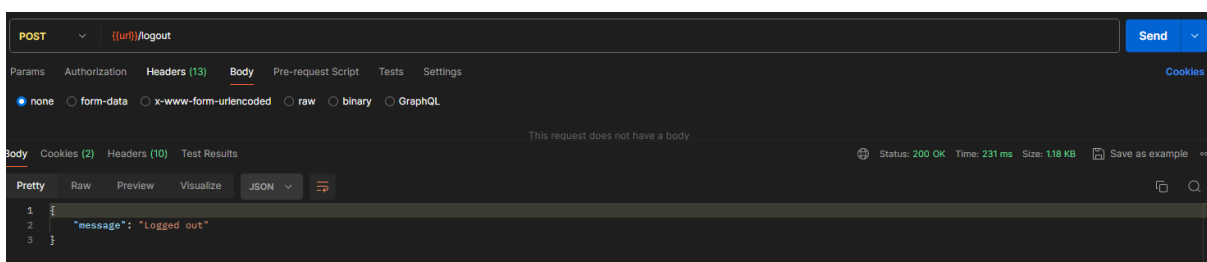


Al lanzar la petición nos devolverá los datos del usuario autenticado.

## - Logout:

Para cerrar nuestra sesión, tenemos la siguiente url:

Url para cerrar sesión: **POST** {{url}}/logout



Tras realizar la petición, obtendremos el mensaje de **Sesión Cerrada**.

## 2. Obtener Personajes:

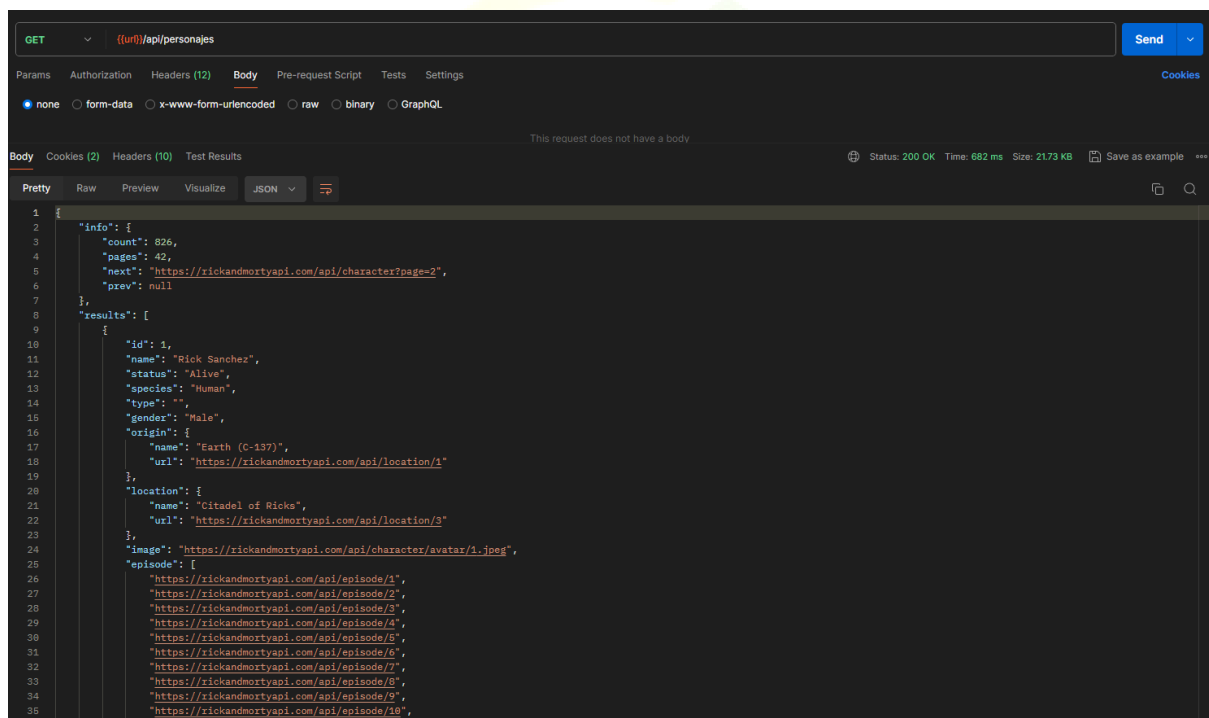
Con la API podremos obtener personajes de la serie Ricky & Morty usando una API externa:

Documentación de la API de Ricky & Morty: <https://rickandmortyapi.com/documentation/>

### - Obtener todos los personajes:

Podremos obtener todos los personajes en diferentes páginas, además de poder filtrarlos por: 'name', 'status', 'species', 'type', 'gender'.

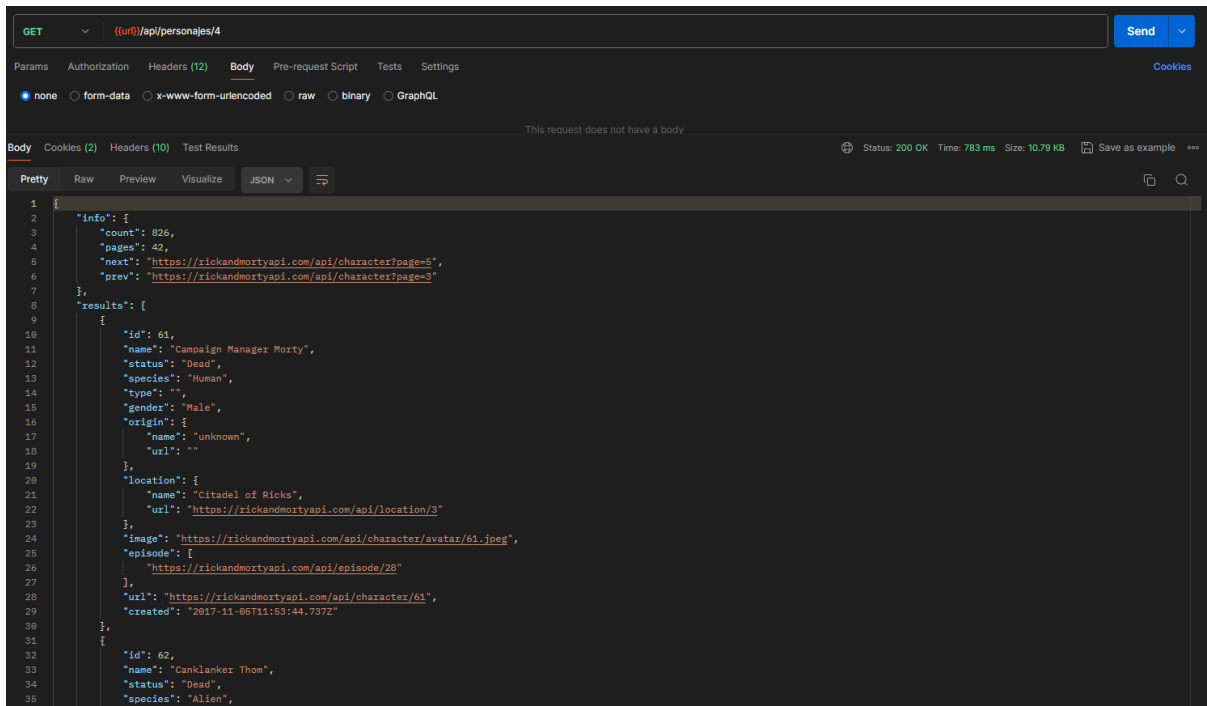
Url para obtener los personajes de la primera página: **GET** `{{url}}/api/personajes` o `{{url}}/api/personajes/1`



Obtendremos como respuesta todos los personajes de la primera página.

### -Ejemplo de paginación

Url para obtener los personajes de otra pagina: **GET** `{{url}}/api/personajes/{{página}}`



```

1 {
2   "info": {
3     "count": 826,
4     "pages": 42,
5     "next": "https://rickandmortyapi.com/api/character?page=5",
6     "prev": "https://rickandmortyapi.com/api/character?page=3"
7   },
8   "results": [
9     {
10      "id": 61,
11      "name": "Campaign Manager Morty",
12      "status": "Dead",
13      "species": "Human",
14      "type": "",
15      "gender": "Male",
16      "origin": {
17        "name": "unknown",
18        "url": ""
19      },
20      "location": {
21        "name": "Citadel of Ricks",
22        "url": "https://rickandmortyapi.com/api/location/3"
23      },
24      "image": "https://rickandmortyapi.com/api/character/avatar/61.jpeg",
25      "episode": [
26        "https://rickandmortyapi.com/api/episode/28"
27      ],
28      "url": "https://rickandmortyapi.com/api/character/61",
29      "created": "2017-11-05T11:03:44.737Z"
30    },
31    {
32      "id": 62,
33      "name": "Canklanker Thom",
34      "status": "Dead",
35      "species": "Alien",

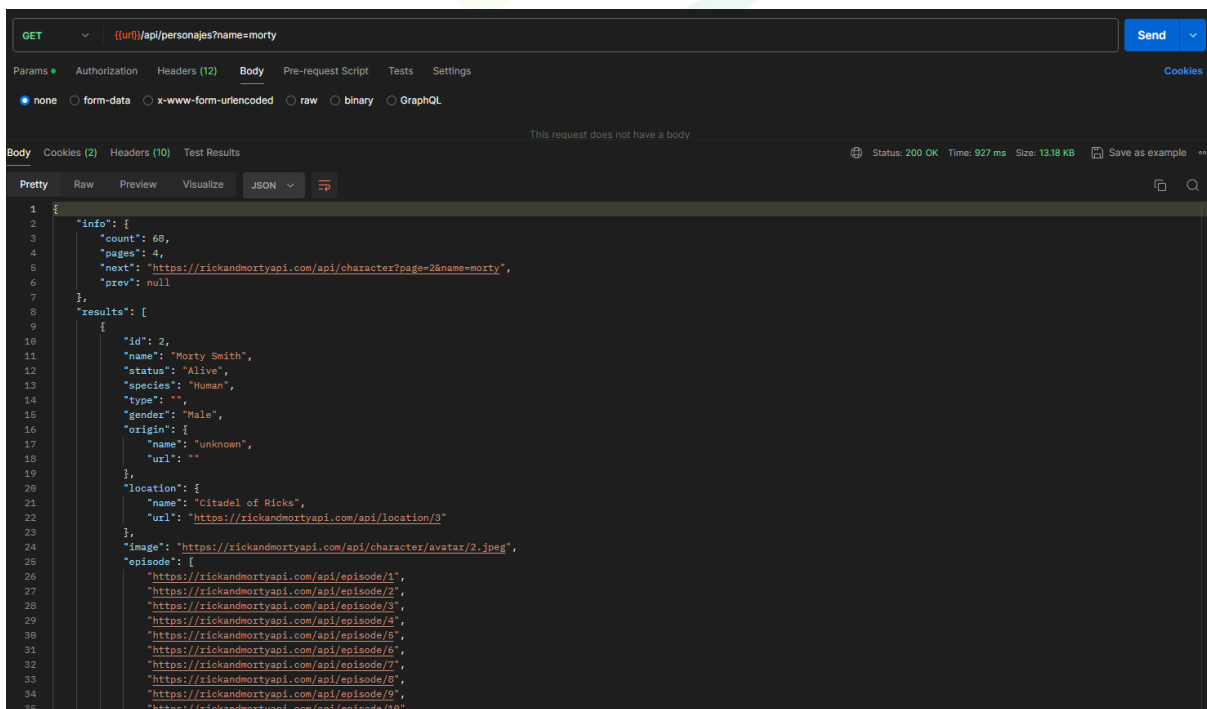
```

En esta ocasión la respuesta será los personajes de la página 4.

## -Ejemplo de filtro:

Podremos también filtrar los personajes por: 'name', 'status', 'species', 'type', 'gender'.

Url para obtener los personajes de otra pagina: **GET** `{{url}}/api/personajes?name=name` o `{{url}}/api/personajes/{página}?status=status`



```

1 {
2   "info": {
3     "count": 60,
4     "pages": 4,
5     "next": "https://rickandmortyapi.com/api/character?page=2&name=morty",
6     "prev": null
7   },
8   "results": [
9     {
10      "id": 2,
11      "name": "Morty Smith",
12      "status": "Alive",
13      "species": "Human",
14      "type": "",
15      "gender": "Male",
16      "origin": {
17        "name": "unknown",
18        "url": ""
19      },
20      "location": {
21        "name": "Citadel of Ricks",
22        "url": "https://rickandmortyapi.com/api/location/3"
23      },
24      "image": "https://rickandmortyapi.com/api/character/avatar/2.jpeg",
25      "episode": [
26        "https://rickandmortyapi.com/api/episode/1",
27        "https://rickandmortyapi.com/api/episode/2",
28        "https://rickandmortyapi.com/api/episode/3",
29        "https://rickandmortyapi.com/api/episode/4",
30        "https://rickandmortyapi.com/api/episode/5",
31        "https://rickandmortyapi.com/api/episode/6",
32        "https://rickandmortyapi.com/api/episode/7",
33        "https://rickandmortyapi.com/api/episode/8",
34        "https://rickandmortyapi.com/api/episode/9",
35        "https://rickandmortyapi.com/api/episode/10"

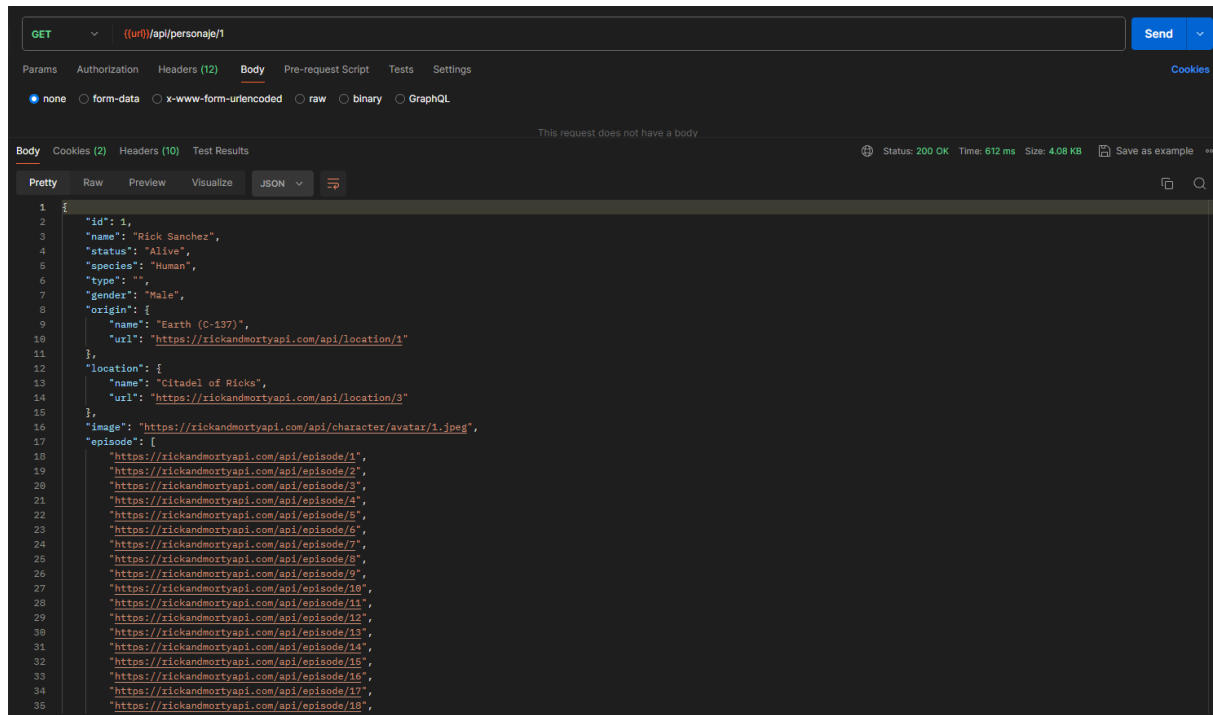
```

La respuesta de la petición será los personajes que su nombre contenga “morty”, los resultados también están paginados con lo cual si añadimos la página antes del atributo o atributos por los cuales queremos filtrar, obtendremos las demás páginas.

## - Obtener un personaje por su ID

Podemos obtener los detalles de un solo personaje mediante su ID:

Url para obtener un personaje por su ID: **GET** `{{url}}/api/personaje/{id}`



Obtenemos como respuesta los detalles del personaje cuyo id pasemos por la url.

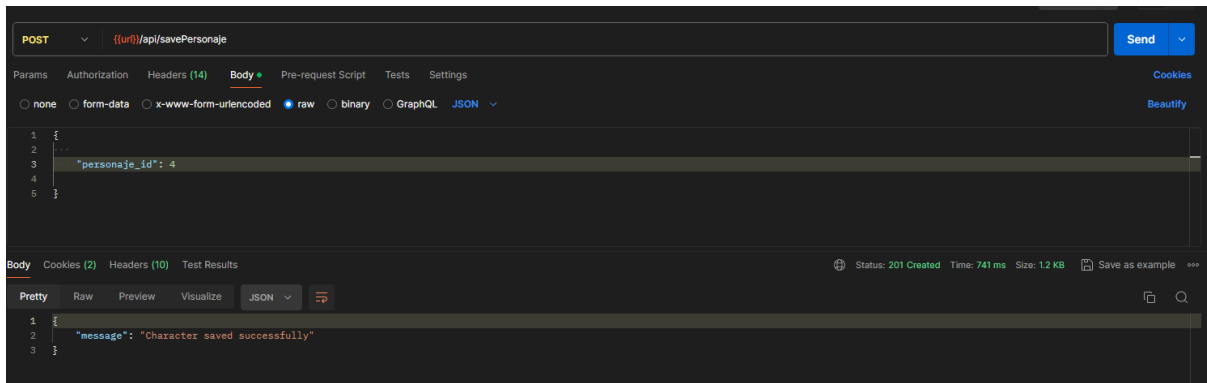
## 3. Guardar Personajes:

Si estamos **autenticados** en la API podremos guardar nuestros personajes favoritos.

### - Guardar Personajes:

Url para guardar un personaje: **POST** `{{url}}/api/savePersonaje`



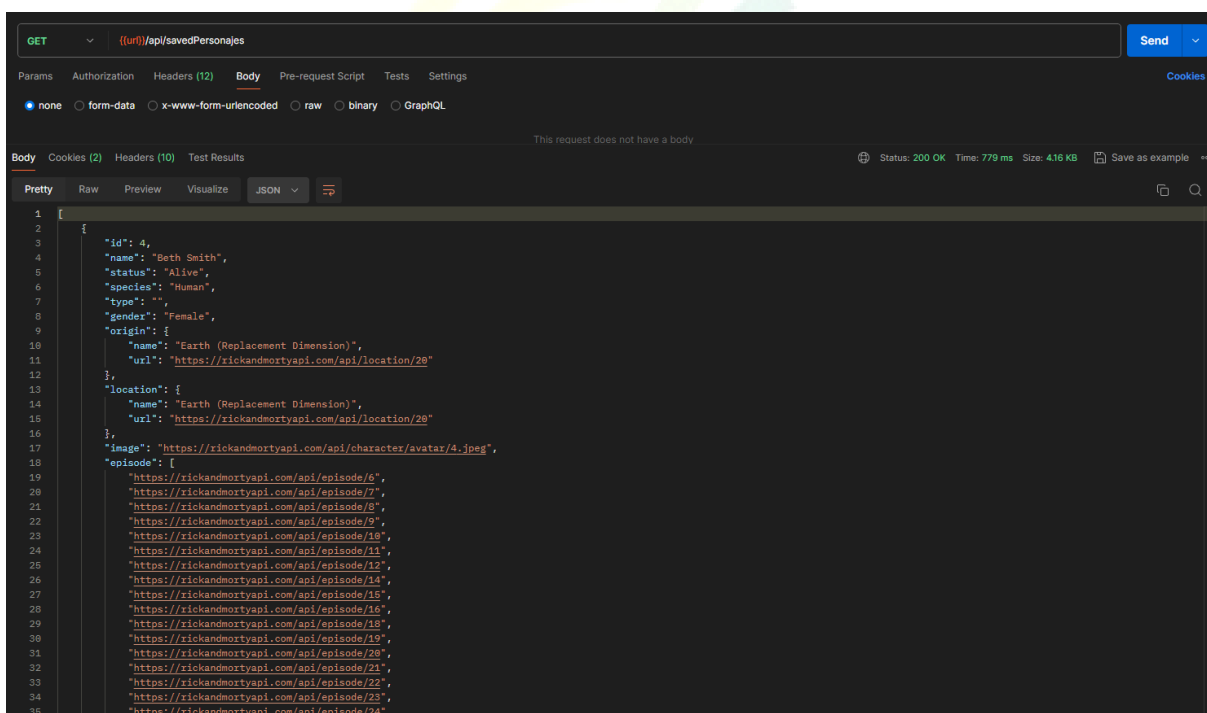


A la petición le pasaremos como parámetro el ID del personaje que queremos guardar, nos devolverá que el personaje se ha guardado correctamente.

## - Obtener los personajes guardados:

Podremos obtener los detalles de todos los personajes que tenemos guardados.

Url para obtener la lista de personajes guardados: **GET** {{url}}/api/savedPersonajes

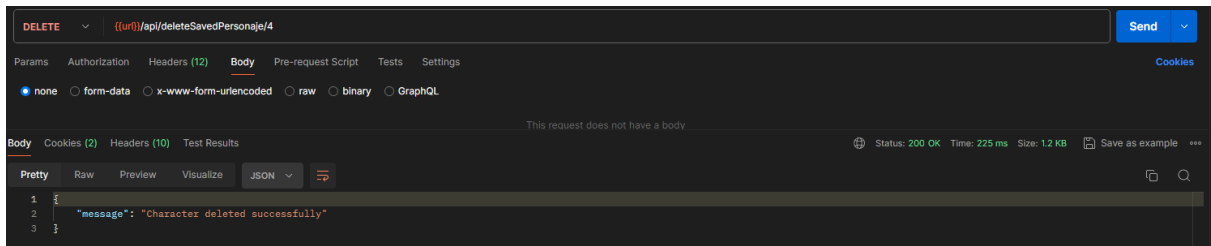


Obtendremos como respuesta la lista con los detalles de todos los personajes guardados.

## - Eliminar un personaje guardado:

También podremos eliminar personajes de nuestra lista de favoritos.

Url para eliminar un personaje de la lista de guardados: **DELETE** {{url}}/api/deleteSavedPersonaje/{id}



Obtendremos como respuesta que el personaje se ha eliminado correctamente.

