**API** **REST**

**Versión 1.0**

## Introducción

Este documento sirve como referencia para el uso y consumo de esta API REST la cual interactúa con personajes de una serie animada. Explica métodos, parámetros esperados y las respuestas que resulten de cada una de las solicitudes.

**Autor**

Pablo Bernal Rosas

**Nota**

Para el consumo de la API no será requerido algún tipo de token.

|  |  |
| --- | --- |
| **Nombre** | **Consulta de personajes** |
| **URL** | http://localhost/api/sps/helloworld/v1/ |
| **Método** | GET |
| **Parámetros URL** | Numérico (opcional) |
| **Parámetros** | Ninguno |
| **Respuesta Exitosa** | **Contenido**:  {  "status": 200,  "msg": "success",  "character": {  "id": "11",  "name": "Albert Einstein",  "status": "Dead",  "species": "Human",  "origin": "Earth (C-137)",  "gender": "Male"  }  } |
| **Respuesta Errónea** | {  "status": 400,  "msg": "Bad request"  }  Si se envió un id y no existe se devuelve lo siguiente:  {  "status": 400,  "msg": "Character not found."  } |
| **Notas** |  |

|  |  |
| --- | --- |
| **Nombre** | **Insertar un personaje** |
| **URL** | http://localhost/api/sps/helloworld/v1/ |
| **Método** | POST |
| **Parámetros URL** | Ninguno |
| **Parámetros Body** | name(String)  species(String)  status(String)  gender(String)  origin(String) |
| **Respuesta Exitosa** | **Contenido**:  {  "status": 200,  "msg": "Character created successfully."  } |
| **Respuesta Errónea** | {  "status": 400,  "msg": "Bad request"  } |
| **Notas** |  |

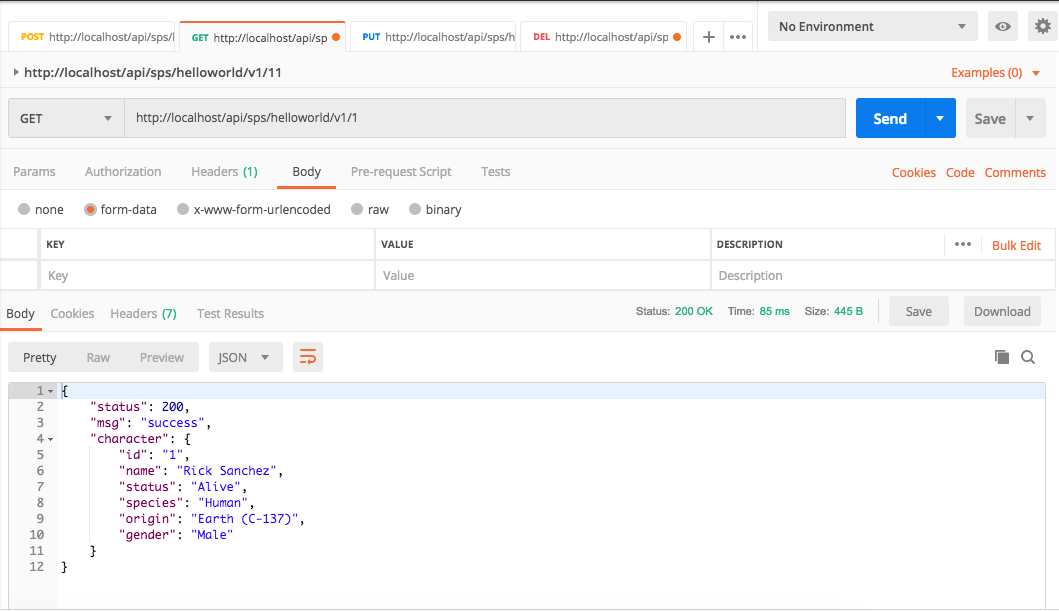
|  |  |
| --- | --- |
| **Nombre** | **Actualizar un personaje** |
| **URL** | http://localhost/api/sps/helloworld/v1/ |
| **Método** | PUT |
| **Parámetros URL** | Numérico (obligatorio) |
| **Parámetros Body** | En formato JSON se enviaran los datos a actualizar, ejemplo:  {  "name": "Summer Smith"  } |
| **Respuesta Exitosa** | **Contenido**:  {  "status": 200,  "msg": "Character updated successfully."  } |
| **Respuesta Errónea** | {  "status": 400,  "msg": "Bad request"  } |
| **Notas** |  |

|  |  |
| --- | --- |
| **Nombre** | **Eliminar un personaje** |
| **URL** | http://localhost/api/sps/helloworld/v1/ |
| **Método** | DELETE |
| **Parámetros URL** | Numérico (obligatorio) |
| **Parámetros Body** | Ninguno |
| **Respuesta Exitosa** | **Contenido**:  {  "status": 200,  "msg": "Character deleted successfully."  } |
| **Respuesta Errónea** | {  "status": 400,  "msg": "Bad request"  }  Si el id, no existe se devuelve lo siguiente:  {  "status": 400,  "msg": "Character not found."  } |
| **Notas** |  |

**Pruebas del API con Postman:**

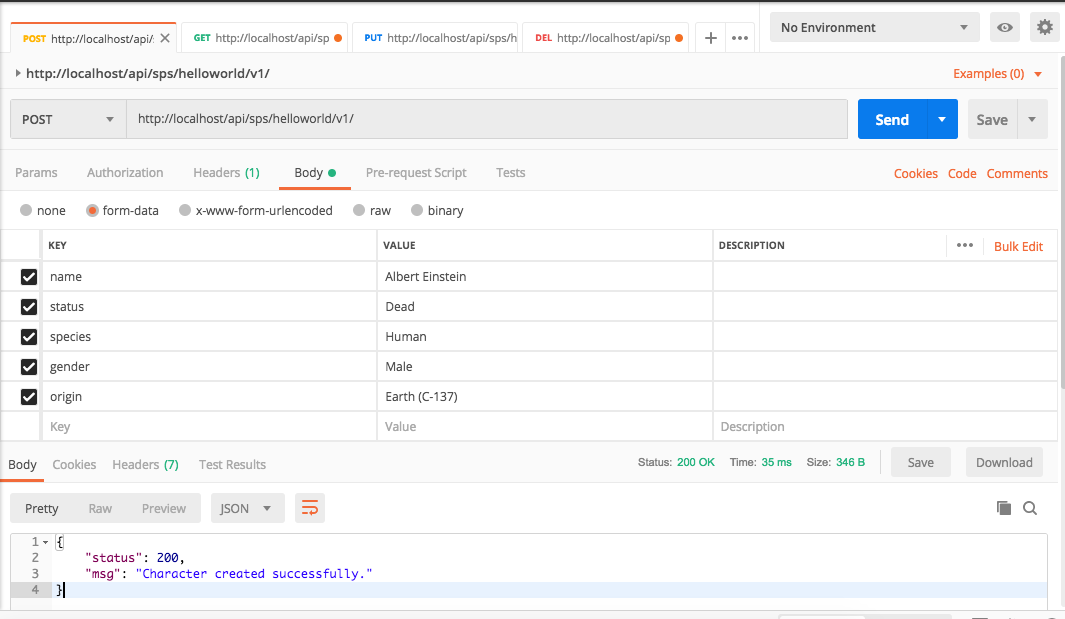
**Metodo GET**

Se envia como parametro en la URL 1, lo cual nos devolvera el personaje con el id 1, si no se envia un id, nos devuelve todos los personajes.

****

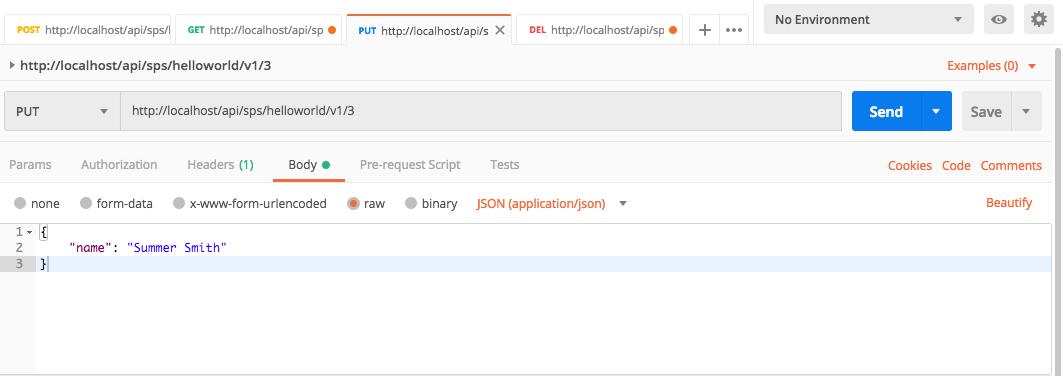
**Metodo POST**

Se envia los parametros a insertar dentro del body

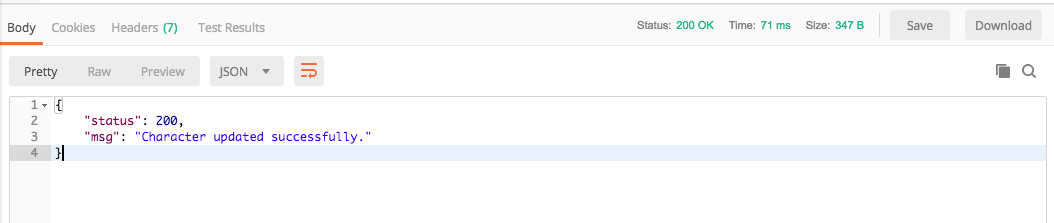
****

**Metodo PUT**

En la URL se envía el id que se quiere actualizar, y en el body, en formato JSON enviaremos los campos a editar

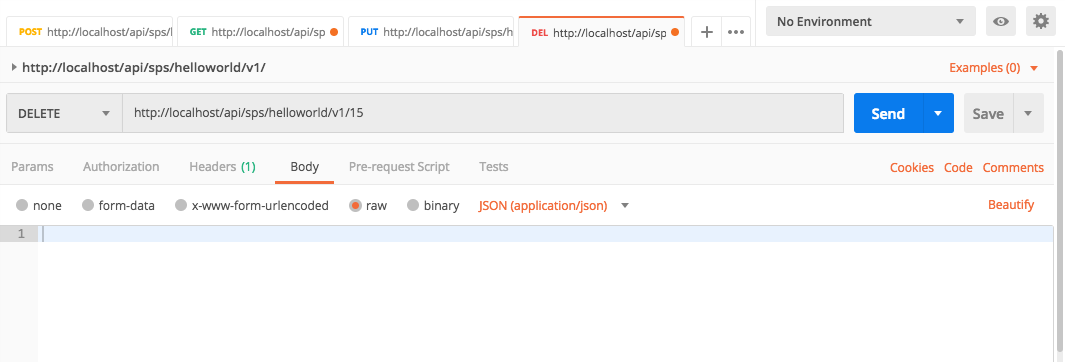


Si la respuesta es exitosa nos regresara un mensaje como el siguiente:

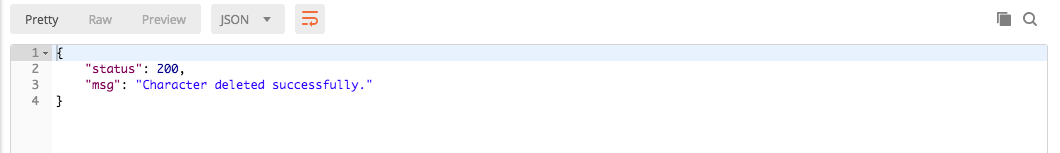


**Metodo DELETE**

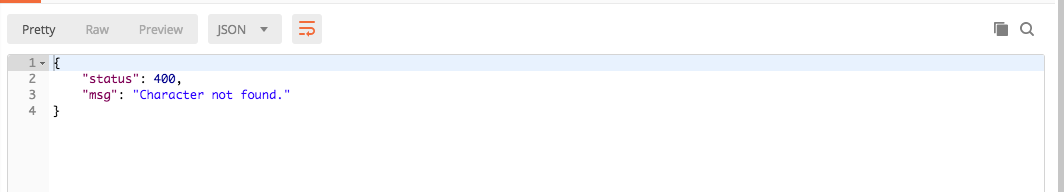
En la URL se manda el id a eliminar, no se envía nada dentro del body:



Si todo ha ido bien nos devolverá un mensaje como el siguiente:



Si enviamos un id de un personaje que no existe veremos el siguiente mensaje:



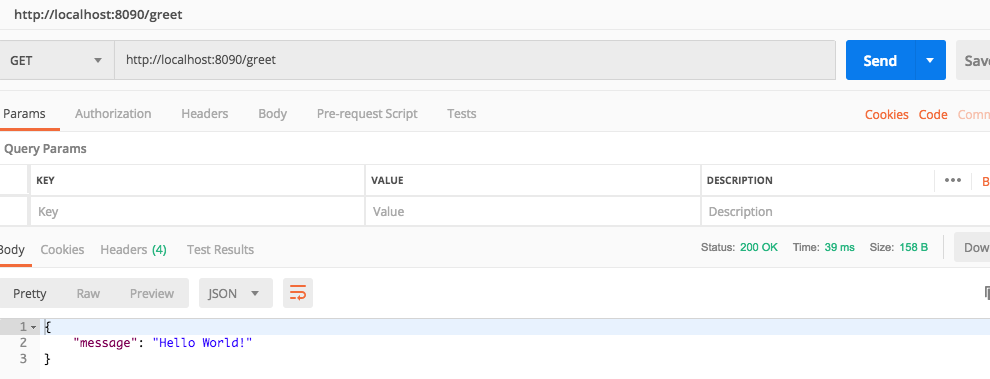
**Aplicación dentro de un contenedor (Docker)**

Pude levantar Docker con Codeigniter pero no en el puerto 8090, tuve unos problemas con el apache y no quería demorarme mas en terminar el examen.

Así que levante un micro servicio con Helidon, con ayuda de su guía rápida.

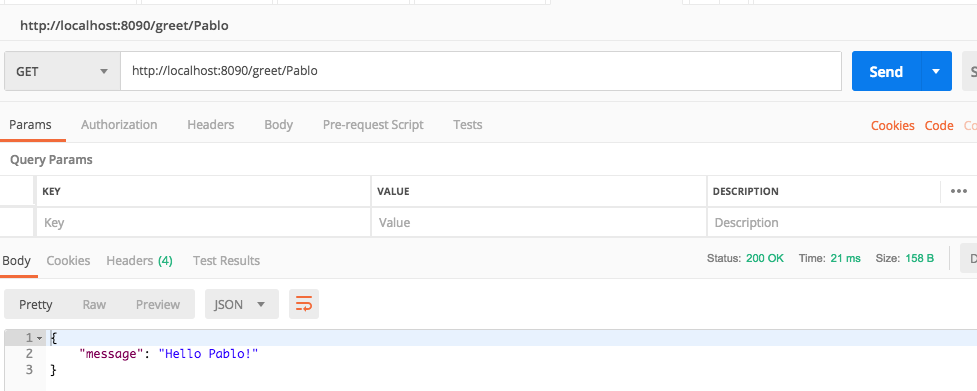
**Metodo GET**

En esta primer pantalla tenemos el método GET, el cual solo nos responde un “”Hello World”



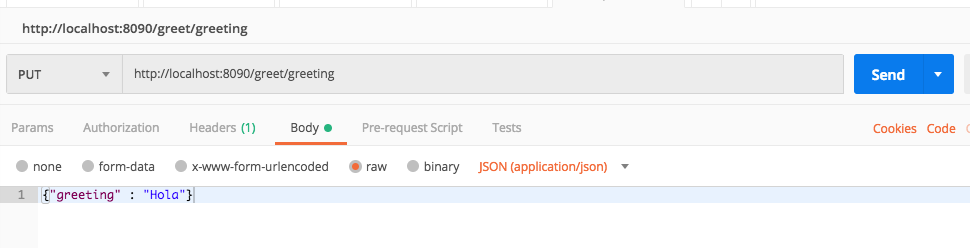
**Metodo GET con parametro**

Este método si lleva parámetro en la URL, y nos responde un saludo, con el nombre que pasamos de parámetro.



**Metodo PUT**

En el body mandaremos un JSON para actualizar el mensaje del saludo, para que de “Hello” pase a “Hola”



**Metodo GET con parametro**

Este método se vuelve a usar para comprobar que el método anterior hizo la actualización correctamente,

