



ESPECIFICACIÓN SERVICIO REST PARA LA ENTIDAD PERSONAJE

Pacto de Honor

Facultad de Ingeniería y Ciencias básicas.
Politécnico Grancolombiano.

Contenido

Control de versiones	3
Objetivo	4
Métodos	4
getID	4
Parámetro	4
Consumo inválido	5
find	6
Parámetros	6
Consumo inválido	6
No satisfactorio	7
create	7
Satisfactorio:	8
No satisfactorio:	9
edit	10
Parámetros	11
Satisfactorio:	11
No satisfactorio:	12
getData	12
Parámetros	12
Satisfactorio	12
No satisfactorio	13
count	13
Parámetros:	13

Control de versiones

Versión	Fecha	Autor	Cambios realizados
0.1	11/04/2017	Nicolás Rubiano	Versión inicial del documento.
0.2	19/04/2017	Nicolás Rubiano	Se agrega el caso no satisfactorio para el método getData
0.3	28/04/2017	Nicolás Rubiano	Todos los servicios reciben como parámetro el token que se otorgó al realizar el login. Se inserta una descripción a cada imagen. Se agrega el método getId.

Objetivo

El objetivo del presente documento, es brindar herramientas de conocimiento los desarrolladores que desean consumir los servicios REST que proporcionará el área de desarrollo de Backend, específicamente sobre la entidad Personaje.

Los siguientes, son las especificaciones que se proporcionarán para tener acceso a todos los datos referentes sobre la entidad:

Métodos

getID

EndPoint: /Polifight/webresources/personaje/getId

Este método retorna una lista con los Id y los nombres de cada uno de los personajes. Se debe consumir por método GET y recibe como parámetro el token que fue enviado al realizar el login.

Parámetro

“token”



Imagen 1: Lista de los Personaje al consumir el método getId, enviando el token

Consumo inválido

En caso de enviar un token inválido, se enviará un JSON informando lo sucedido:

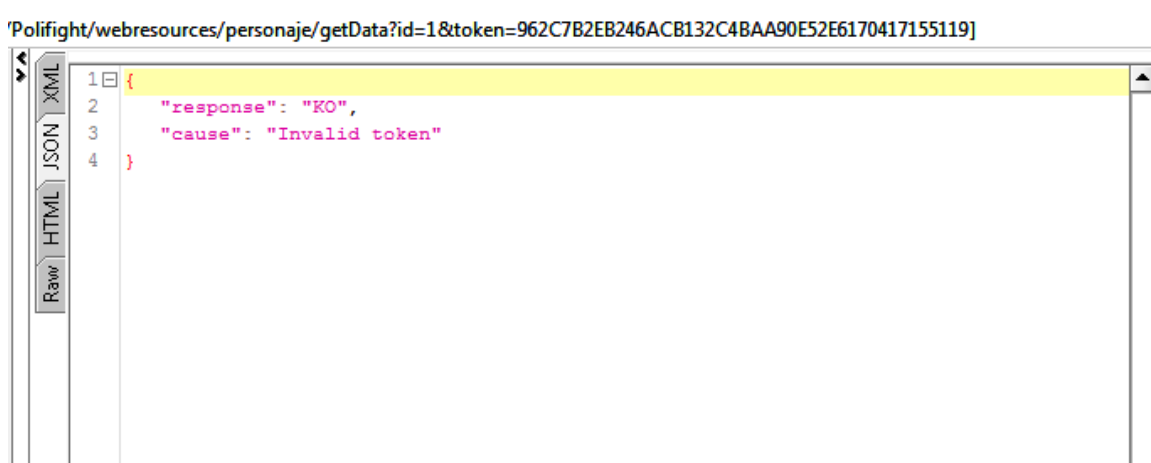


Imagen 2: Consumir método getId con un token inválido.

find

EndPoint: /Polifight/webresources/personaje/find

Este método se deberá consumir por método GET. Retornará los atributos del Personaje de acuerdo al id enviado.

Parámetros

“id”: id del Personaje que se desea consultar.

“token”: String correspondiente al token que se otorgó en el momento de hacer el login.

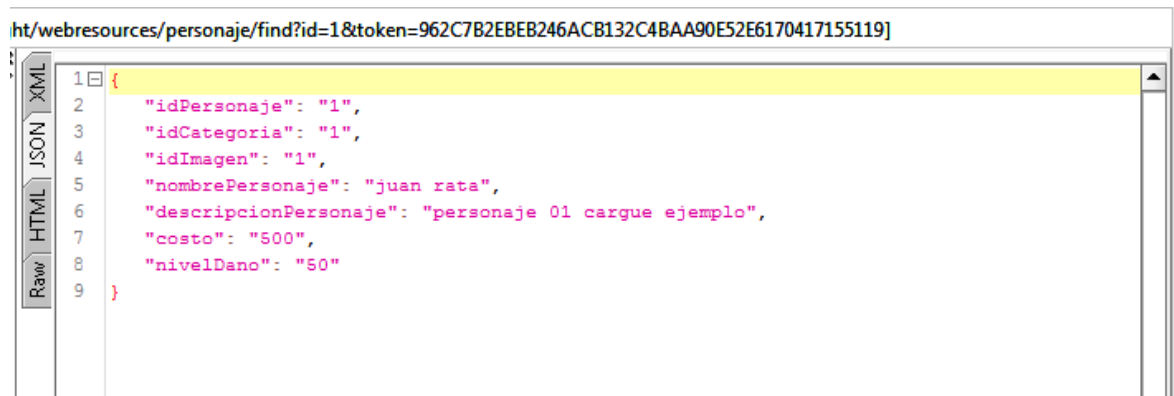


Imagen 3: Consumir método find con un token válido.

Consumo inválido



Imagen 4: Consumir método find con un token inválido.

No satisfactorio

Si el servicio tiene algún problema con la solicitud, se responderán dos campos:

- "response": En caso de algún inconveniente se responderá "KO".
- "cause": Causa del error, para éste método se identifica sólo 1 error, Id del personaje no encontrado en la DB

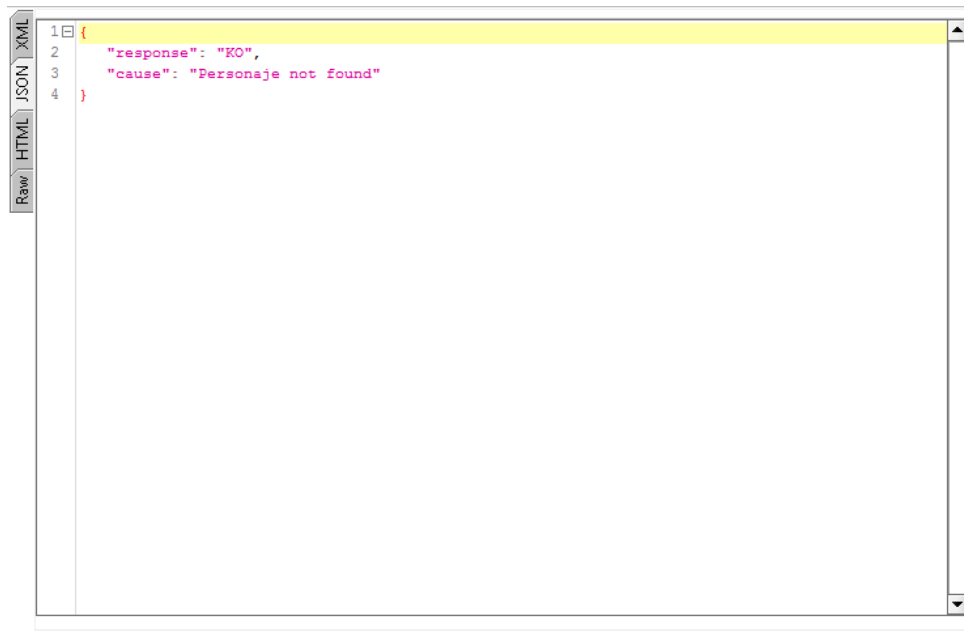


Imagen 5: Envio de un id que no se encuentra en la DB.

create

EndPoint: /Polifight/webresources/personaje/create

Se debe enviar un JSON por el método POST con los atributos de la entidad Personaje y se debe enviar el token en la URL de acuerdo como se muestra en la imagen 6:

ersonaje/create?token=962C7B2EBEB246ACB132C4BAA90E52E6170417155119]

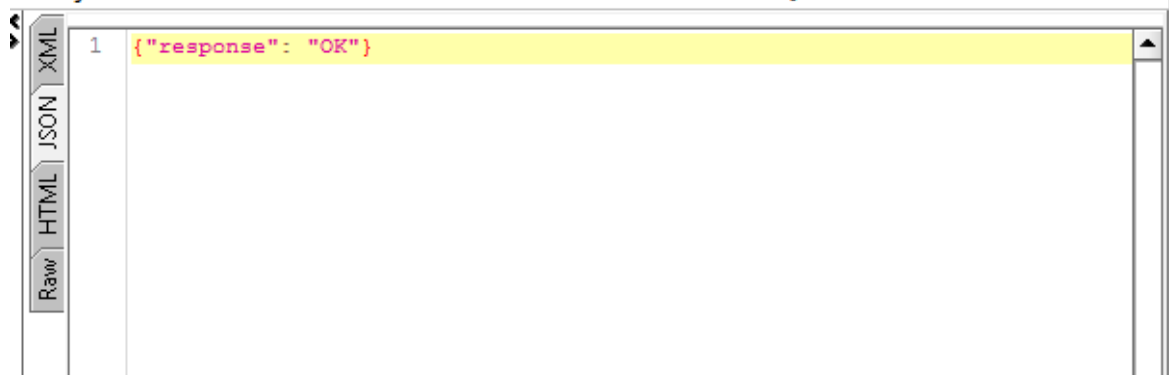
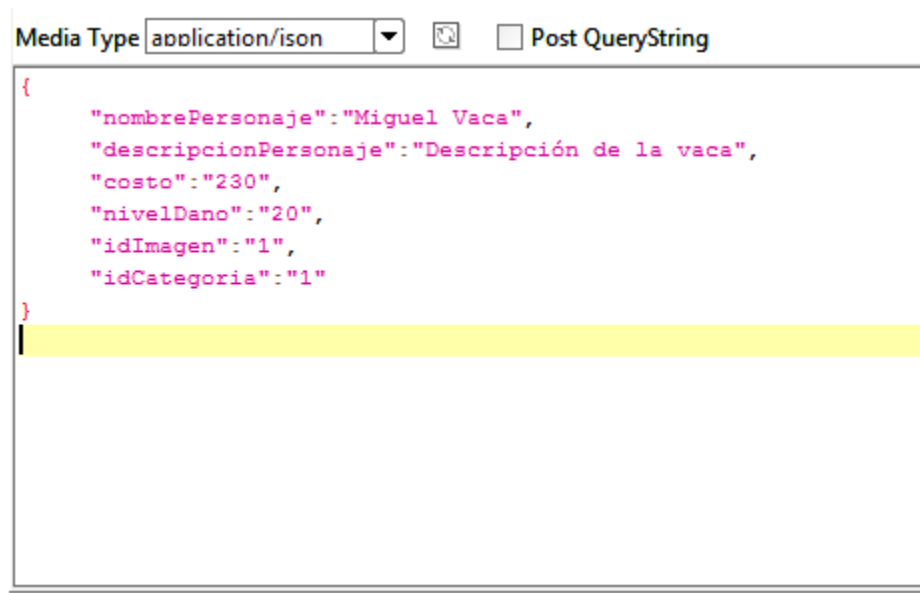


Imagen 6: Envio del token en la URL para crear un Personaje.



The image shows a REST client interface. At the top, there is a 'Media Type' dropdown menu set to 'application/json'. To its right is a checkbox labeled 'Post QueryString' which is unchecked. Below this, a text area contains a JSON object with the following properties: 'nombrePersonaje' (Miguel Vaca), 'descripcionPersonaje' (Descripción de la vaca), 'costo' (230), 'nivelDano' (20), 'idImagen' (1), and 'idCategoria' (1). The JSON is formatted with pink text and is enclosed in curly braces. A yellow highlight bar is visible at the bottom of the text area.

```
{
  "nombrePersonaje": "Miguel Vaca",
  "descripcionPersonaje": "Descripción de la vaca",
  "costo": "230",
  "nivelDano": "20",
  "idImagen": "1",
  "idCategoria": "1"
}
```

Imagen 7: Envio de los atributos en formato JSON para crear un Personaje.

Es importante recalcar que los nombres de los atributos que se envían deben tener los nombres exactos que muestran en la imagen anterior.

Satisfactorio:

El servicio responde OK en el campo “response” en caso de que la creación se haya realizado con éxito:



Imagen 8: Respuesta satisfactoria del servicio crée.

Result Grid							
Filter Rows:							
Edit:							
Export/Import:							
Wrap Cell Content:							
	id_personaje	id_categoria	id_imagen	nombre_personaje	descripcion_personaje	costo	nivel_dano
	7	1	1	Pedro Conejo	Descripción del nuevo personaje	100	3
	8	1	1	Pedro Pez	Descripción del nuevo personaje	100	3
	9	1	1	Miguel Vaca	Descripción de la vaca	230	20
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Imagen 9: Visualización de los datos creados en la DB.

No satisfactorio:

- Si el servicio no puede procesar la solicitud se responderá un KO en el campo “response”. En este caso se incluirá el campo “cause” donde se indicará la causa del error

- ```
{
 response: KO
 cause: Time Out
}
```
- Si se envía un token inválido, se responderá un JSON igual que el mostrado en la imagen 4 que se encuentra en la sección [find](#)

## edit

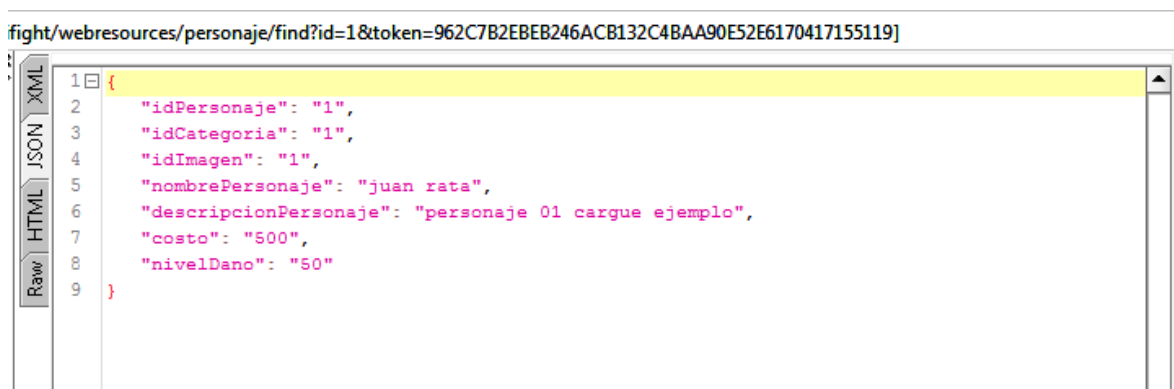
EndPoint: /Polifight/webresources/personaje/edit

Al igual que todos los métodos descritos en el presente documento, se debe enviar el token obtenido al momento de realizar el login. El consumo debe ser por método POST.

Para efectos de ejemplo, modificaremos el nombre del dato insertado anteriormente en el ejemplo del método [create](#), por “Lola Vaca” enviando el siguiente JSON:

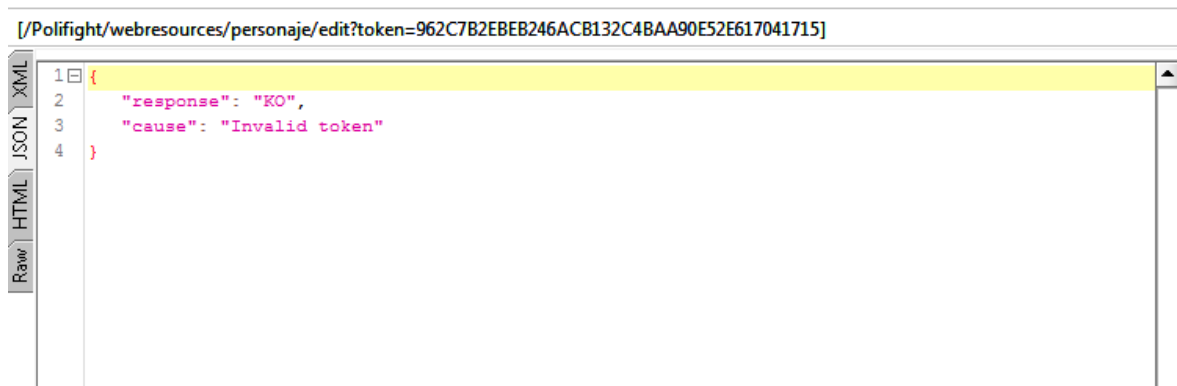


**Imagen 10: Formato de JSON que debe ser enviado para consultar el Personaje.**



**Imagen 11: Envío del token en la URL para el método edit.**

En caso de que el token no sea válido, se enviarán el JSON informando al consumidor:



The screenshot shows a REST client interface with the URL `[/Polifight/webresources/personaje/edit?token=962C7B2EBEB246ACB132C4BAA90E52E617041715]`. The response is displayed in JSON format, showing an error message.

```
1 {
2 "response": "KO",
3 "cause": "Invalid token"
4 }
```

**Imagen 12: Token inválido enviado en el método edit.**

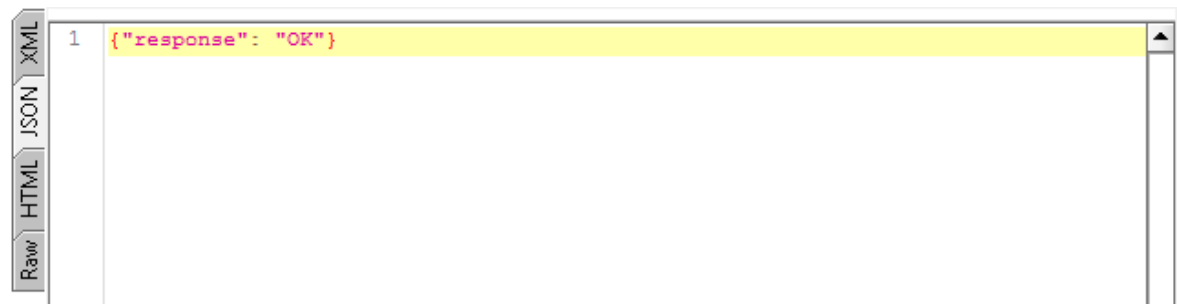
## Parámetros

“id”: Corresponde al id del Personaje que se desea consulta.

“token”: String correspondiente al token que se obtiene al momento de realizar el login.

## Satisfactorio:

Si la modificación al atributo fue satisfactorio, se responderá con el campo “response”:"OK":



The screenshot shows a REST client interface with a successful JSON response.

```
1 { "response": "OK" }
```

**Imagen 13: Mensaje del servicio notificando al consumidor el éxito del cambio**

### No satisfactorio:

Si el servicio no puede procesar la solicitud se responderá un KO en el campo “response”. En este caso se incluirá el campo “cause” donde se indicará la causa del error



*Imagen 14: Mensaje del servicio notificando al consumidor que el id no se encuentra en la DB.*

### getData

URL: /Polifight/webresources/personaje/getData

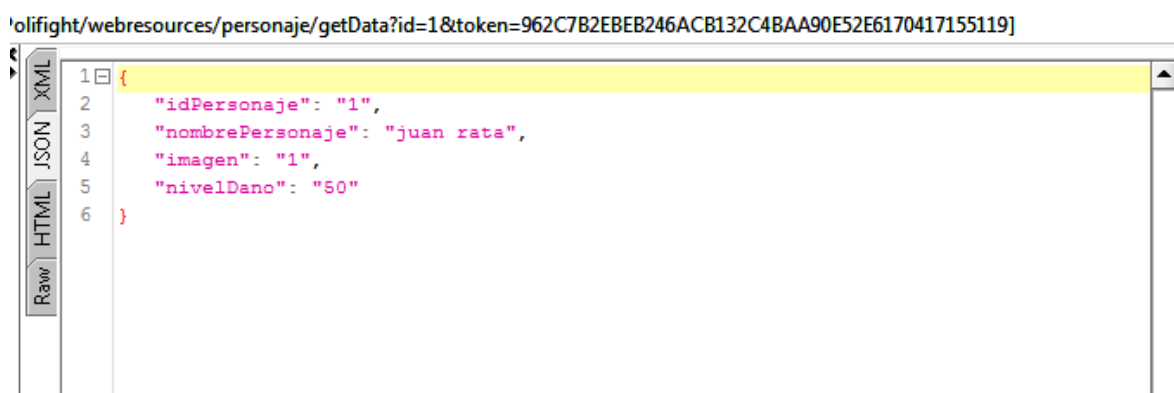
Este método es solicitado con una estructura específica por el equipo de FrontEnd el día Miércoles 5 de Abril. Se consume por método GET.

### Parámetros

“id”: id del Personaje del que se desea obtener los datos.

“token”: String correspondiente al token que se obtuvo al momento de hacer el login.

### Satisfactorio



*Imagen 15: Estructura que retorna el método getData al enviarle el token y el id válido.*

## No satisfactorio

En caso de que no se envíe un id que se encuentre en la base de datos o que se ocasione algún tipo de excepción, el servicio retornará un JSON con dos campos:



*Imagen 16: Estructura que retorna el método `getData` al enviarle el token válido y el id inválido.*

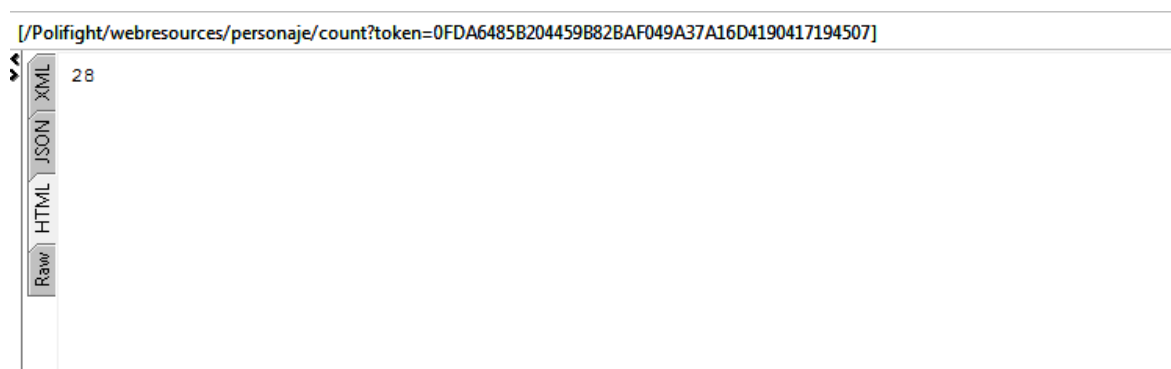
## count

EndPoint: `/Polifight/webresources/personaje/count`

Este método retorna la cantidad de datos que se encuentran en la DB. Se debe consumir por método GET enviándole en Token correspondiente.

### Parámetros:

“token”: Token obtenido al momento de realizar el login.



*Imagen 17: Respuesta del servicio `count` al enviar un token válido.*