

especificación servicio rest para la entidad Personaje

Pacto de Honor

Facultad de Ingeniería y Ciencias básicas.

Politécnico Grancolombiano.

Contenido

[Control de versiones 3](#_Toc482919502)

[Objetivo 4](#_Toc482919503)

[Métodos 4](#_Toc482919504)

[getID 4](#_Toc482919505)

[Parámetro 4](#_Toc482919506)

[Consumo inválido 5](#_Toc482919507)

[find 5](#_Toc482919508)

[Parámetros 5](#_Toc482919509)

[Consumo inválido 6](#_Toc482919510)

[No satisfactorio 6](#_Toc482919511)

[create 7](#_Toc482919512)

[Satisfactorio: 8](#_Toc482919513)

[No satisfactorio: 9](#_Toc482919514)

[edit 9](#_Toc482919515)

[Parámetros 10](#_Toc482919516)

[Satisfactorio 11](#_Toc482919517)

[No satisfactorio 11](#_Toc482919518)

[getData 11](#_Toc482919519)

[Parámetros 11](#_Toc482919520)

[Satisfactorio 12](#_Toc482919521)

[No satisfactorio 12](#_Toc482919522)

[count 12](#_Toc482919523)

[Parámetros: 12](#_Toc482919524)

[badCharacter 13](#_Toc482919525)

[Parámetros 13](#_Toc482919526)

[Satisfactorio 13](#_Toc482919527)

[No satisfactorio 14](#_Toc482919528)

[preBattle 15](#_Toc482919529)

[Parámetros 15](#_Toc482919530)

[Satisfactorio 15](#_Toc482919531)

[No satisfactorio 15](#_Toc482919532)

# 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. |
| 0.4 | 02/05/2017 | Nicolás Rubiano | Se agrega el método badCharacter |
| 0.5 | 19/05/2017 | Nicolás Rubiano | Se agrega el método preBattle |

# 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.

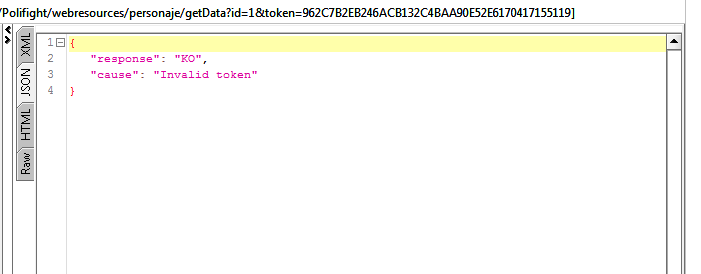
### Parámetro

“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

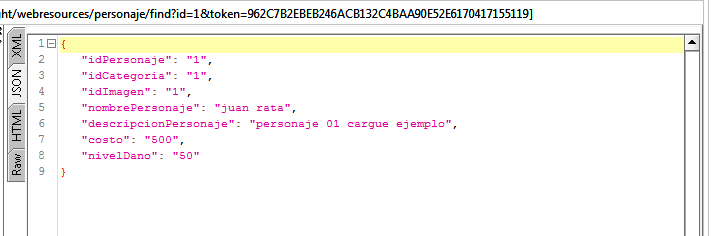
EndPoint: /Polifight/webresources/personaje/find

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

### Parámetros

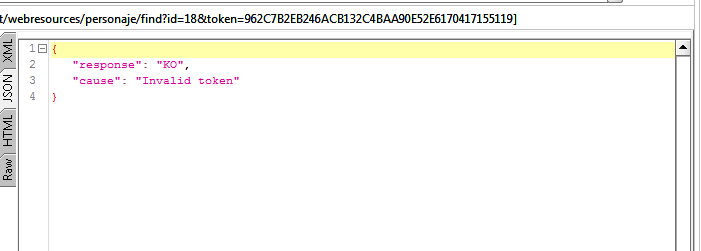
“id”: id del Personaje que se desea consultar.

“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

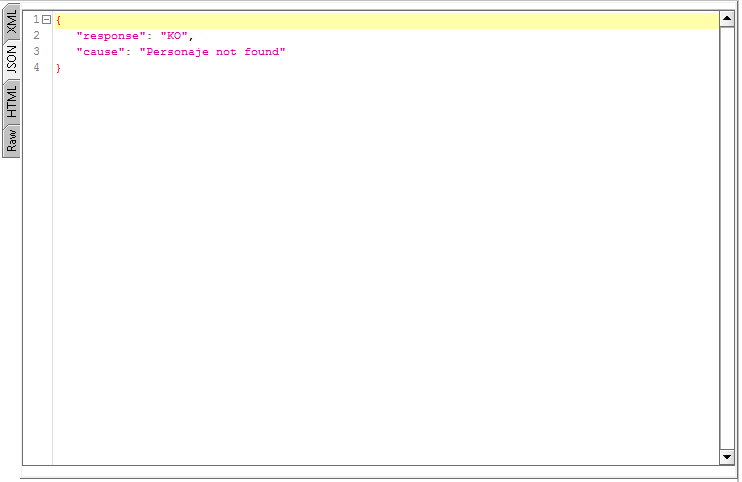


***Imagen 4: Consumir método find con un token inválido.***

### 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

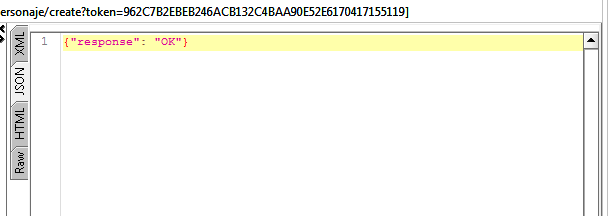


***Imagen 5: Envio de un id que no se encuentra en la DB.***

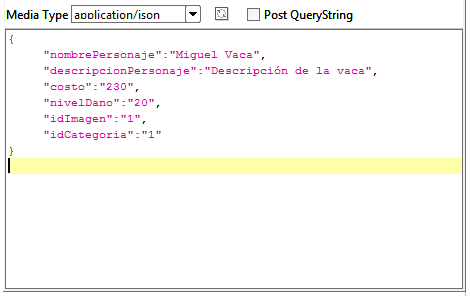
## create

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:



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

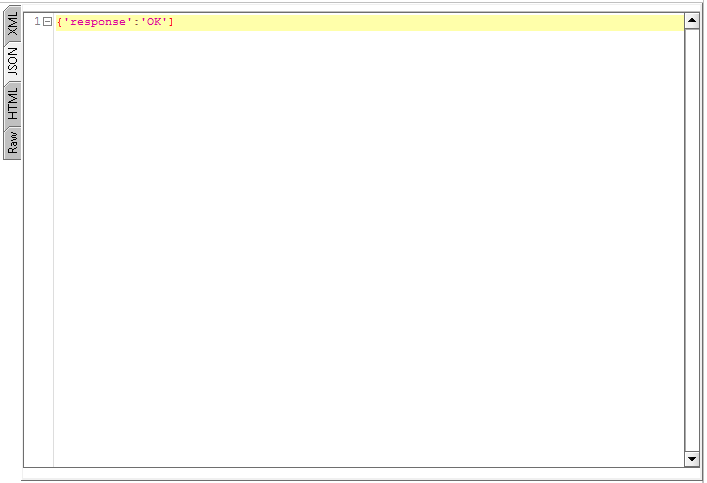


***Imagen 7: Envio de los atributos en formato JSON para crear un Personaje.***

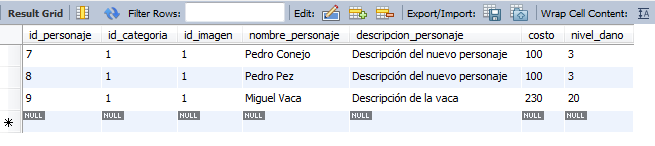
Es importante recalcar que los nombres de los atributos que se envían deben tener los nombres exactos que muestran en la imagen anterior.

### 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 créate.***



***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 envia un token inválido, se responderá un JSON igual que el mostrado en la imagen 4 que se encuentra en la sección [find](#_Consumo_inválido)

## edit

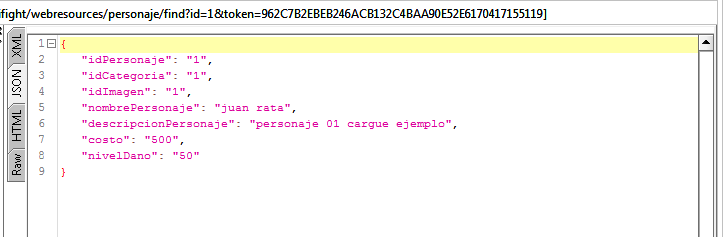
EndPoint: /Polifight/webresources/personaje/edit

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

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

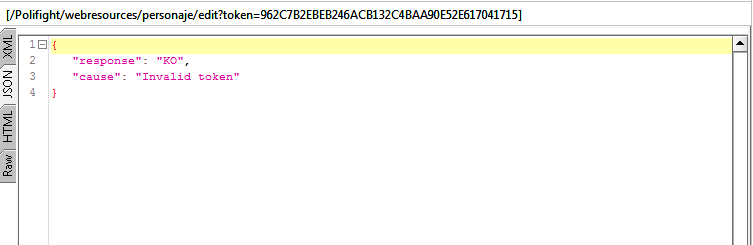


***Imagen 10: Formato de JSON que debe ser enviado para consultar el Personaje.***



***Imagen 11: Envio 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:



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

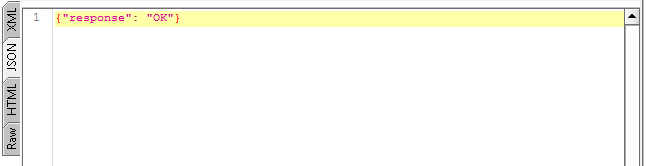
### Parámetros

“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

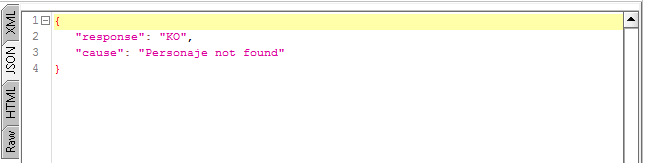
Si la modificación al atributo fue satisfactorio, se responderá con el campo “response”:”OK”:



***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

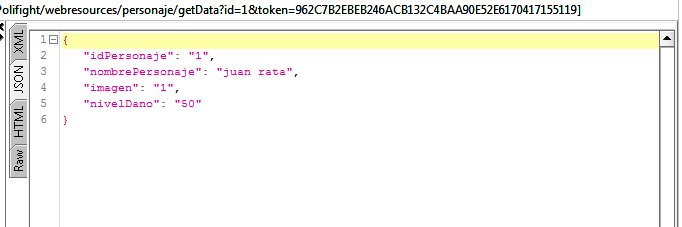
Este método es solicitado con una estructura específica por el equipo de FrontEnd el día Miércoles 5 de Abril. Se consume por método GET.

### 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.

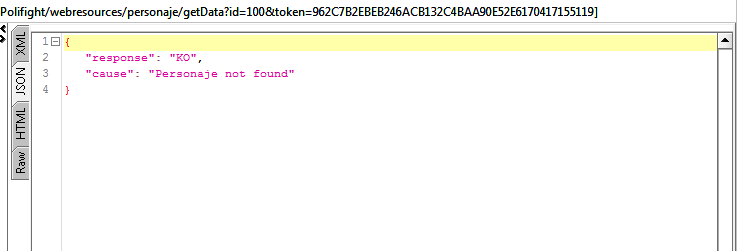
### Satisfactorio



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

### No satisfactorio

En caso de que no se envie 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.***

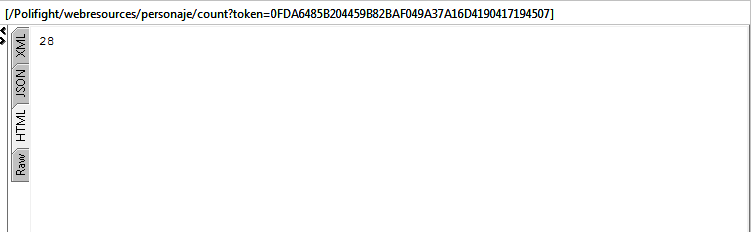
## count

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.***

## badCharacter

EndPoint: /Polifight/webresources/personaje/badCharacter

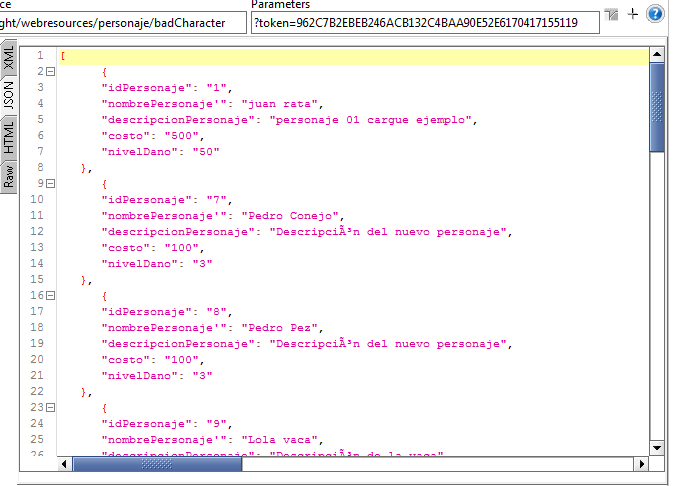
Este método retorna una listar con los Personajes malos que se encuentran registrados en la DB y debe ser consumido por método GET.

### Parámetros

“token”: String correspondiente al token otorgado al momento de realizar el login

### Satisfactorio

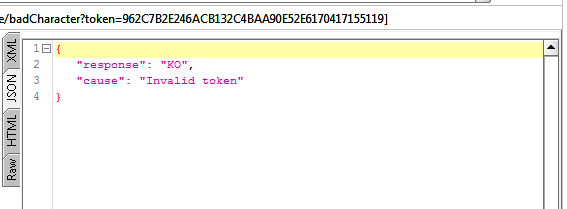
En caso de que el token sea válido, se enviará una lista de todos los personajes que fueron categorizados como malos.



***Imagen 18: Lista que retorna el método badCharacter***

### No satisfactorio

En caso de que el servicio reciba un token que no es válido, se retornará la siguiente estructura:



***Imagen 19: Respuesta en caso de que el token sea inválido***

## preBattle

URL: hhtp// IP:Puerto/Polifight/webresources/personaje/preBattle

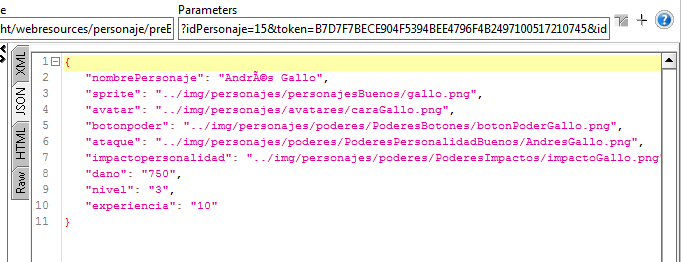
Este método retorna los datos necesarios para cargar una batalla, debe ser consumido a través del método POST .

### Parámetros

* “idPersonaje”: Entero correspondiente al id del personaje que el jugador ha seleccionado.
* “idJugador”: Entero correspondiente al id del jugador que se encuentra en el juego.
* “token”: String con el token que se otorgó al momento de hacer el login.

### Satisfactorio

Si los datos enviados son correctos, el servicio responderá con la siguiente estructura:



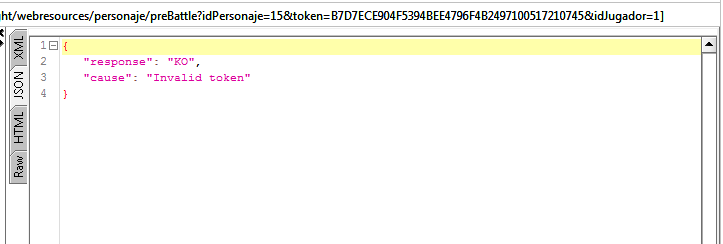
***Imagen 20: Estructura de respuesta al momento de consumir el método preBattle***

### No satisfactorio

Se identifican dos casos en el que el servicio puede enviar un mensaje de error

#### Token inválido

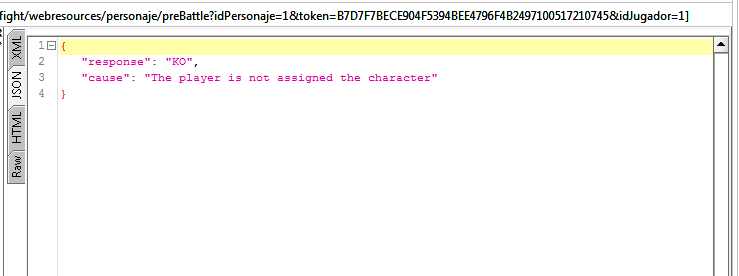
En caso de que el token se envie de forma errónea o no corresponda a uno que se haya otorgado al momento de dar un login, se mostrará la siguiente estructura



***Imagen 21: Estructura de respuesta cuando el token es inválido***

#### Personaje no asignado

Al momento de enviar el id del jugador y el id del personaje, se verifica que el jugador tenga asociado dicho personaje, en caso de que no se encuentren asociados el servicio enviará un mensaje de error.



***Imagen 22: Estructura de respuesta cuando el personaje no se encuentra asociado al jugador.***