
# ESPECIFICACIÓN SERVICIO REST PARA LA ENTIDAD JUGADOR

Pacto de Honor

Facultad de Ingeniería y Ciencias básicas.
Politécnico Grancolombiano.

# Contenido

## Control de versiones

| Versión | Fecha | Autor | Cambios realizados |
|---|---|---|---|
| **0.1** | 22/03/2017 | Nicolás Rubiano | Versión inicial del documento. |
| **0.2** | 03/05/2017 | Nicolás Rubiano | Se agrega el método buscar de acuerdo a la estructura solicitada en el issue |
| **0.3** | 16/05/2017 | Nicolás Rubiano | Se agrega el método comprar de acuerdo a la solicitud del Excel acordado. Solicitud #1 issue #479 |

# 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 Jugador.

Todos los servicios deberán ser consumidos por método POST

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

## find

Endpoint: http://IP:Puerto/Polifight/webresources/jugador/find

Método que retorna la estructura del Jugador de acuerdo a lo solicitado en el issue # . Éste servicio debe ser consumido por método GET.

### Parámetros

“idJugador”: id perteneciente al jugador al que se le van a listar los atributos

“idPersonaje”: id del Personaje asociado al jugador del cual se obtendrá el avatar.

“token”:  String que corresponde al token otorgado al momento de realizar el login.

### Satisfactorio

En caso de que los datos sean válidos, se retornará la siguiente estructura:


***Imagen 1: Estructura de retorno en caso de que los datos enviados sean inválidos.***

Es necesario tener en cuenta que el avatar corresponde al primer jugador seleccionado por el usuario.

### No satisfactorio

En caso de que el token enviado no corresponda al otorgado en el momento de hacer el login, se enviará la siguiente estuctura:

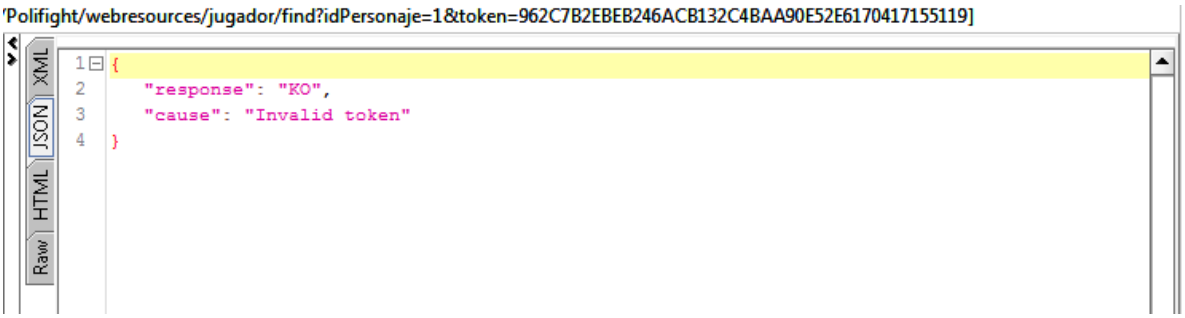


***Imagen 2: Estructura que retorna en caso de que el token no sea válido.***

## compra

Endpoint: [http://IP:Puerto/Polifight/webresources/jugador/compra](http://IP:Puerto/Polifight/webresources/jugador/compra)

Este servicio será consumido por método GET y retornará los personajes buenos que ha desbloqueado el jugador, para ello se debe tener en cuenta los siguientes parámetros:

### Parámetros

“idJugador”: Corresponde al id del jugador que se desea consultar.

“token”: Corresponde al token que se otorga al momento de realizar el login en el aplicativo.

### Caso satisfactorio

Si el jugador tiene personajes asignados y si dichos personajes son “buenos” se enviará una lista con el id y el nombre de cada uno.

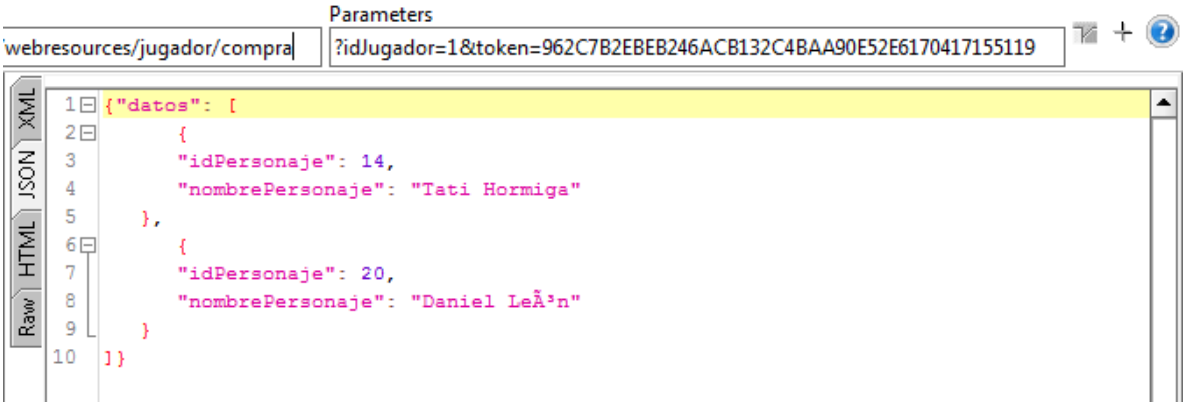


***Imagen 3: Estructura de respuesta para el método de “compra”***

## No satisfactorio

En caso de que el jugador no tenga personajes asignados, se notificará con la siguiente estructura.

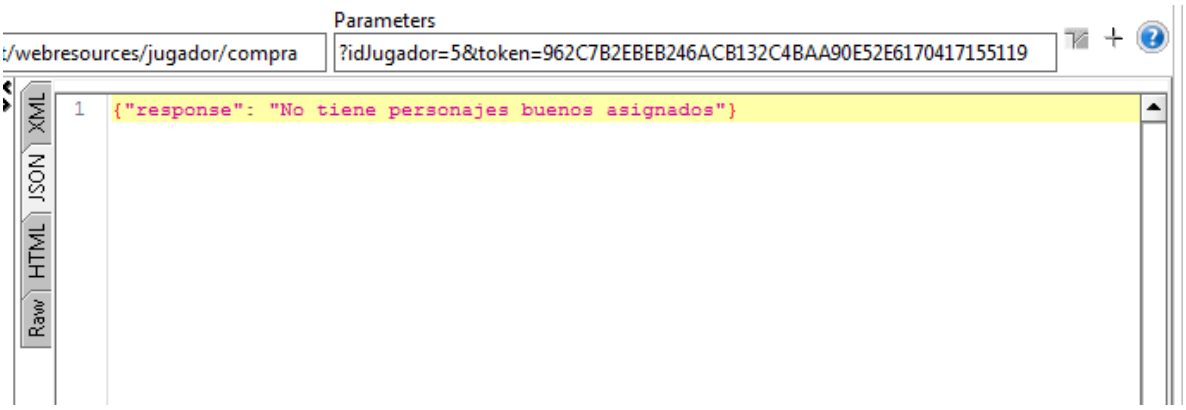


***Imagen 4: Estructura de respuesta en caso de que el jugador no tenga asignado personajes.***

Si se envía un token inválido, el servicio responderá con la siguiente estructura:

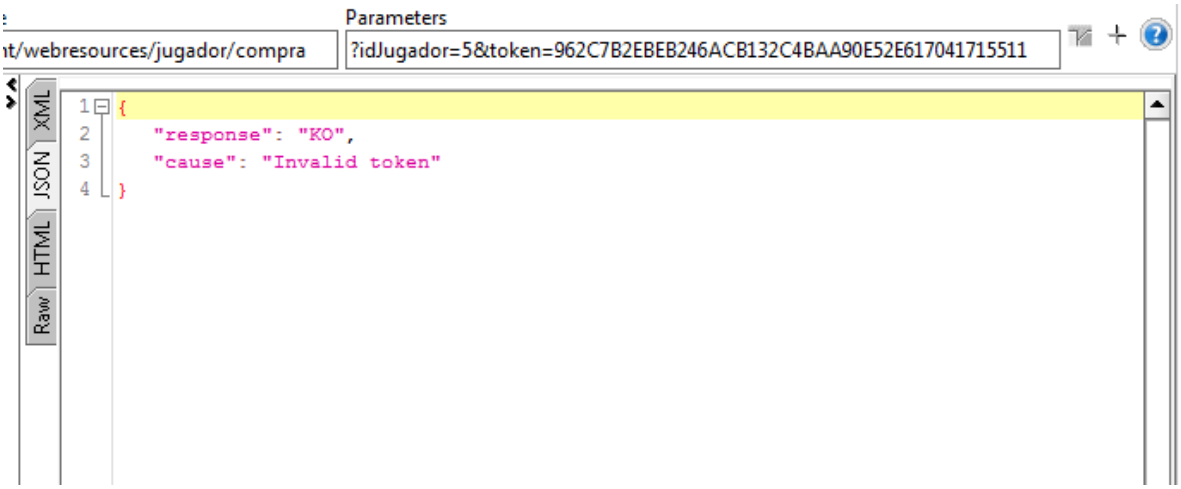


***Imagen 5: Estructura de respuesta en caso de que el token sea inválido***

## findId

Este método recibirá el nickname del jugador y retornará el Id del mismo para ser usado con los demás métodos, por ejemplo:

```
{
  {
    nickname:  “NicknameDelJugador”,
  }
}
```

### Satisfactorio:

```
{
 id: 15
}
```

### No satisfactorio:

Si el servicio tiene algún problema con la solicitud, se responderá con el campo “cause” donde se indicará la causa del error, por ejemplo:

```
{
  cause: Data not found
}
```

## create

Método que permite la creación de un jugador en la base de datos. Éste método recibe todos los atributos del jugador en formato JSON, por ejemplo:

```
{
  {
    nickname:  “NicknameDelJugador”,
    moneda: “NúmeroMonedasJugador”,
    experiencia: “ExperienciaDelJugador”,
    nivel: “NivelDelJugador”,
    numeroNivel: “NúmeroDelNivelDondeEstáElJugador”
  }
}
```

### Satisfactorio:

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

```
{
  response: OK
}
```

### 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
}
```

## getJugador

El método recibe el id del Jugador y retorna los atributos asociados al mismo.

```
{
  {
    id:  “IdDelJugador”,
  }
}
```

La respuesta del servicio es en formato JSON con la siguiente estructura de ejemplo:

```
{
  {
    nickname:  “NicknameDelJugador”,
    moneda: “NúmeroMonedasJugador”,
    experiencia: “ExperienciaDelJugador”,
    nivel: “NivelDelJugador”,
    numeroNivel: “NúmeroDelNivelDondeEstáElJugador”
  }
}
```

## setJugador

Método que servirá para modificar uno o más Jugadores. Éste método recibirá el id del Jugador y la información a actualizar antecedido por la palabra “Jugador”+incremental de acuerdo al número de datos a actualizar, por ejemplo:

```
{
  Jugador1:
  {
    id:  “IdDelJugador”,
    nivel: “NivelDelJugador1”,
    numeroNivel: “NúmeroDelNivelDondeEstáElJugador1”
  }
  Jugador2:
  {
    id:  “IdDelJugador2”,
    moneda: “NúmeroMonedasJugador2”,
    experiencia: “ExperienciaDelJugador2”,
    nivel: “NivelDelJugador2”,
    numeroNivel: “NúmeroDelNivelDondeEstáElJugador2”
  }
  .
  .
  .
  Jugador(n):
  {
    id:  “IdDelJugador(n)”,
    moneda: “NúmeroMonedasJugador(n)”,
  }
}
```

El servicio responde OK en el campo “response” en caso de que la creación se haya realizado con éxito y KO en caso contrario; si “response” es igual a KO, se incluirá el campo “cause” donde se indicará la causa del error, por ejemplo:

### Satisfactorio:

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

```
{
  response: OK
}
```

### 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: Unexpected value in field moneda
}
```

## listPersonajes

(Se necesita el método para buscar personaje por id para poder implementarlo)

Éste método, recibirá el id del Jugador y retornará los personajes que ha desbloqueado el mismo:

```
{
  {
    id:  “IdDelJugador”,
  }
}
```

### Satisfactorio:

```
{

  {

    categoria: “CategoriaDelPersonaje”,

    imagen: “ImagenDelPersonaje”,
```

nombre: “NombreDelPersonaje”,

decripcion: “DescripcionDelPersonaje”,

costo: “CostoDelPersonaje”,

dano: “DañoQueCausaElPersonaje

}

}

### No satisfactorio:

En caso de que el método no pueda retornar la lista de personajes, se enviará el campo “cause” donde se indicará la causa del error:

{
cause: Data not found
}