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	Revisado por	Cambios realizados
0.1	22/03/2017	Nicolás Rubiano		Versión inicial del documento.

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:

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 médoto 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
}
```