**ESPECIFICACIÓN SERVICIO REST**

**PARA LA ENTIDAD PODER**

**Definición:**

Si bien el término **REST** se refería originalmente a un conjunto de principios de arquitectura —descritos más abajo—, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente **HTTP** para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (**XML, JSON**, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo **SOAP**. Es posible diseñar sistemas de servicios web de acuerdo con el estilo arquitectural **REST** de Fielding y también es posible diseñar interfaces **XMLHTTP** de acuerdo con el estilo de llamada a procedimiento remoto (**RPC**), pero sin usar **SOAP**. Estos dos usos diferentes del término **REST** causan cierta confusión en las discusiones técnicas, aunque **RPC** no es un ejemplo de **REST**.

Los sistemas que siguen los principios **REST** se llaman con frecuencia **RESTful**.

**REST** afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

* Un protocolo cliente/servidor sin estado: cada mensaje **HTTP** contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en **HTTP** utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de **URLs**, no son permitidas por **REST**).
* Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: **HTTP** en sí define un conjunto pequeño de operaciones, las más importantes son **POST**, **GET**, **PUT** y **DELETE**. Con frecuencia estas operaciones se equiparan a las operaciones **CRUD** en bases de datos (**CLAB** en castellano: crear,leer,actualizar,borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
* Una sintaxis universal para identificar los recursos. En un sistema **REST,** cada recurso es direccionable únicamente a través de su **URI**.
* El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema **REST** son típicamente **HTML** o **XML**. Como resultado de esto, es posible navegar de un recurso **REST** a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructuraadicional**.**

**Control de versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Revisado por** | **Cambios realizados** |
| 0.1 | Marzo 21/2017 | Andres Sierra |  | Primera versión del documento |
| 0.2 | Marzo 22/2017 | Andres Sierra |  | Especificación de métodos en formato JSON |

**Descripción:**

**Servicio Restful “Poder”**

Se ha creado un servicio **Restful** que permita consumir datos de la tabla “poder”, además de la tabla dependiente “imagen”, ambas pertenecientes a la base de datos **p\_h01**, el servicio entregará un **JSON/XML** (para efectos del proyecto se usará **JSON**)que ejecutará desde la aplicación y permitirá hacer un **GET** de los datos alojados en las tablas mencionadas. Usando el **id\_poder** se puede obtener la imagen correspondiente al poder usado por el personaje.

|  |  |  |
| --- | --- | --- |
| **Parámetro** | **Tipo de Dato** | **Descripción** |
| Id\_poder | Integer | Identificación única que tiene cada uno de los poderes que pueden ser usados por cada personaje. |
| Id\_imagen | Integer | Identificación única que tiene cada imagen de poder, estás están unidas a un poder específico. |

**Métodos:**

**findAllPower()**

Este método devolverá un listado de todos los registros en la tabla de poder:

**Salida:**

* **Evento Satisfactorio:**

**Status:** 200 (OK)

**Response:** [{"descripcionPoder":"Poder efectuado desde el frente","formaPoder":1,"imagen":{"foto":"/poderes/poder1","idImagen":7},"nombrePoder":"golpe frontal","poderPK":{"idImagen":7,"idPoder":1},"potenciaPoder":10,"tipoPoder":1},{"descripcionPoder":"Poder efectuado desde la parte superior","formaPoder":2,"imagen":{"foto":"/poderes/poder2","idImagen":8},"nombrePoder":"golpe superior","poderPK":{"idImagen":8,"idPoder":2},"potenciaPoder":15,"tipoPoder":2}]

* **Evento No satisfactorio:**

Si el servicio tiene algún problema con la solicitud, se responderá con el campo “Response” donde se indicará petición fallida, por ejemplo:

**Status:** 404 (Not Found)

**Response:** GET RequestFailed RequestFailed --> Status: (404)

**findPower()**

Este método devolverá un objeto de tipo poder con los argumentos de un registro identificado por una llave de tipo entero llamada id\_poder.

**Entrada:**

{id\_poder:1}

**Salida:**

* **Evento Satisfactorio:**

**Status:** 200 (OK)

**Response:** {"descripcionPoder":"Poder efectuado desde el frente","formaPoder":1,"imagen":{"foto":"/poderes/poder1","idImagen":7},"nombrePoder":"golpe frontal","poderPK":{"idImagen":7,"idPoder":1},"potenciaPoder":10,"tipoPoder":1}

* **Evento No satisfactorio:**

Si el servicio tiene algún problema con la solicitud, se responderá con el campo “Response” donde se indicará petición fallida, por ejemplo:

**Status:** 404 (Not Found)

**Response:** GET RequestFailed RequestFailed --> Status: (404)

**createPower()**

Este método creara un registro en la tabla poder de la base de datos, con los argumentos necesarios.

**Entrada:**

{id\_poder:5,id\_imagen:5,nombre\_poder:”NombrePoder”,descripcion\_poder:”DescripcionPoder”,tipo\_poder:1,forma\_poder:1,potencia\_poder:1}

**Salida:**

* **Evento Satisfactorio:**

**Status:** 200 (OK)

**Response:** Create Sucessful

* **Evento No satisfactorio:**

Si el servicio tiene algún problema con la solicitud, se responderá con el campo “Response” donde se indicará petición fallida, por ejemplo:

**Status:** 404 (Not Found)

**Response:** GET RequestFailed RequestFailed --> Status: (404)

**modifyPower()**

Este método podrá modificar un registro de la tabla de poder buscando dicho registro con el id\_poder.

**Entrada:**

{id\_poder:1}

**Salida:**

* **Evento Satisfactorio:**

**Status:** 200 (OK)

**Response:** Modify Sucessful

* **Evento No satisfactorio:**

Si el servicio tiene algún problema con la solicitud, se responderá con el campo “Response” donde se indicará petición fallida, por ejemplo:

**Status:** 404 (Not Found)

**Response:** GET RequestFailed RequestFailed --> Status: (404)