UNIVERSIDAD NACIONAL
DEL COMAHUE
FACULTAD DE INFORMÁTICA

# Trabajo Práctico N°1

Framework e Interoperabilidad


Integrantes

Marcela Fabiana Muller – Leg: FAI-304

Julio Samir Chaar – Leg: FAI-

Alam Boris Ravbar – Leg: FAI-103

# Introducción:

## ¿Qué es Rest?

Según Wikipedia Rest es:

"**La Transferencia de Estado Representacional** (en inglés Representational State Transfer) o *REST es un estilo de arquitectura* software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo."

**REST** es una forma simple de organizar interacciones entre sistemas independientes. REST te permite trabajar de forma sencilla con clientes con diferentes sistemas operativos y plataformas como smartphones. En principio no está atado a la web, pero casi siempre se implementa en ella, ya que se fundamenta en HTTP.

## ¿Que és Hipermedia?

Según Wikipedia Hipermedia es:

"**Hipermedia** es el término con el que se designa al conjunto de métodos o procedimientos para escribir, diseñar o componer contenidos que integren soportes tales como: *texto, imagen, video, audio, mapas y otros soportes de información emergentes*, de tal modo que el resultado obtenido, además, tenga la posibilidad de interactuar con los usuarios".

## Y ¿Qué es una arquitectura?

Según Wikipedia una Arquitectura es:

"En el libro "An introduction to Software Architecture", David Garlan y Mary Shaw definen que la **Arquitectura** es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el *diseño y especificación de la estructura global del sistema es un nuevo tipo de problema*".

## ¿Pero para qué me sirve REST?

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, 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.*

### ¿Qué es HTTP?

HTTP es un protocolo que permite el envío y recibo de documentos a través de la web. Un protocolo es un conjunto de normas que determina qué mensajes se pueden intercambiar y cómo deben ser.

### ¿Cómo está compuesto HTTP?

Los mensajes HTTP están compuestos de headers y de un body. El body puede ir vacío, contiene datos que se pueden transmitir por la red en función de las instrucciones de los headers. Los headers contienen metadatos, como información sobre la codificación de los mensajes. En el caso de un request, también contiene métododos HTTP.  En REST, los headers son más importantes que el propio body.

### ¿Cuáles son los métodos más usados en HTTP?

Los métodos mas usados por HTTP y que nos van a servir son:

- **GET** es el método más simple de HTTP. Es el que utilizan los navegadores cada vez que se hace click en un enlace o se escribe una URL en la barra de navegación. Le dice al servidor que transmita los datos identificados por la URL al cliente. Los datos nunca deberían ser modificados en el lado del servidor con un GET request. Un GET request es sólo de lectura, pero una vez que el cliente recibe los datos, puede hacer cualquier operación, como darle formato para mostrarlo.
- Un **PUT** request se utiliza cuando se quiere crear o editar el resource identificado por la URL. No hay nada en el request que informe al servidor como se deben crear los datos, sólo que debe crearlos. Esto permite poder cambiar la tecnología del backend si en algún momento es necesario. Los PUT requests contienen los datos a usar a la hora de actualizar o crear un resource en el body. En cURL se pueden añadir datos al request con -d.
- El método **DELETE** hace lo contrario que PUT, se usa cuando se quiere eliminar un *resource* identificado por la **URL del request**. Esto eliminará todos los datos asociados con el resource.
- **POST** se emplea cuando el proceso que se quiere que ocurra pueda ser repetido si el POST se repite (esto es, que no es idempotente). Además los POST requests deben procesar el request body como subordinado de la URL a la que se está enviando.

### Clasificación de métodos HTTP

- **Métodos seguros e inseguros**: los métodos seguros son los que nunca modifican *resources*. El único seguro listado en este artículo es GET.
- **Métodos idempotentes**: obtienen el mismo resultado independientemente del número de veces que se repita el request: GET, PUT y DELETE. El único método no idempotente es POST. PUT y DELETE parece raro que lo sean pero aquí está la explicación: repetir un método PUT con exactamente el mismo body modifica un *resource* de forma que permanece idéntico al descrito en el PUT request anterior, por lo que nada cambia. De la misma forma, no tiene sentido con DELETE eliminar un *resource* dos veces. Dicho esto, no importa cuantas veces se utilice PUT y DELETE, el resultado será el mismo como si sólo se hubiera hecho una vez.

Es importante utilizar el método correcto en la situación adecuada, lo que depende totalmente del desarrollador.

### ¿Cuales son las partes de Rest?

Rest utiliza una serie de diseños claves que ya se encuentran en la web, estos son:

- *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
- 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 infraestructura adicional*.


El cliente HTTP y el servidor HTTP intercambian información sobre los resources identificados por las URLs. El request y el response contienen una representación del resource. Por representación se entiende información en un formato específico, sobre el estado del resource o sobre cómo estará el estado en el futuro. Tanto los headers como el body son parte de esa representación.

Los headers HTTP, que contienen metadatos. Sólo pueden contener texto y están formateados de una manera específica.

El body puede contener datos en cualquier formato, y aquí es donde se puede ver el poder de HTTP. Se pueden enviar textos, imágenes, HTML y XML en cualquier idioma. A través de los metadatos del request de diferentes URLs, se puede elegir entre diferentes representaciones del mismo resource. Por ejemplo se puede enviar una página web a los navegadores y un JSON a las aplicaciones.

La respuesta HTTP debería especificar el content type del body. Esto se realiza en el header, en el campo Content-Type. Por ejemplo: "Content/Type: application/json"

## ¿Qué es URI?

Un **identificador de recursos uniforme** o URI —del inglés uniform resource identifier— es una cadena de caracteres que **identifica los recursos de una red de forma unívoca**.La diferencia respecto a un localizador de recursos uniforme (URL) es que los URI pueden ser localizador de recursos uniforme (URL), uniform resource name (URN), o ambos.

Un URI consta de las siguientes partes:

- Esquema: nombre que se refiere a una especificación para asignar los identificadores, e.g. urn:, tag:, cid:. En algunos casos también identifica el protocolo de acceso al recurso, por ejemplo http:, mailto:, ftp:, etc.
- Autoridad: elemento jerárquico que identifica la autoridad de nombres (por ejemplo //www.example.com).
- Ruta: Información usualmente organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres (e.g. /domains/example).
- Consulta: Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?'.
- Fragmento: Permite identificar una parte del recurso principal, o vista de una representación del mismo. El comienzo de este componente se indica mediante el carácter '#'.

Aunque se acostumbra llamar URL a todas las direcciones web, URI es un identificador más completo y por eso es recomendado su uso en lugar de la expresión URL.

*Un URI se diferencia de un URL en que permite incluir en la dirección una subdirección, determinada por el "fragmento"*.

Para poder usar URI en php usamos una librería que se llama: curl.

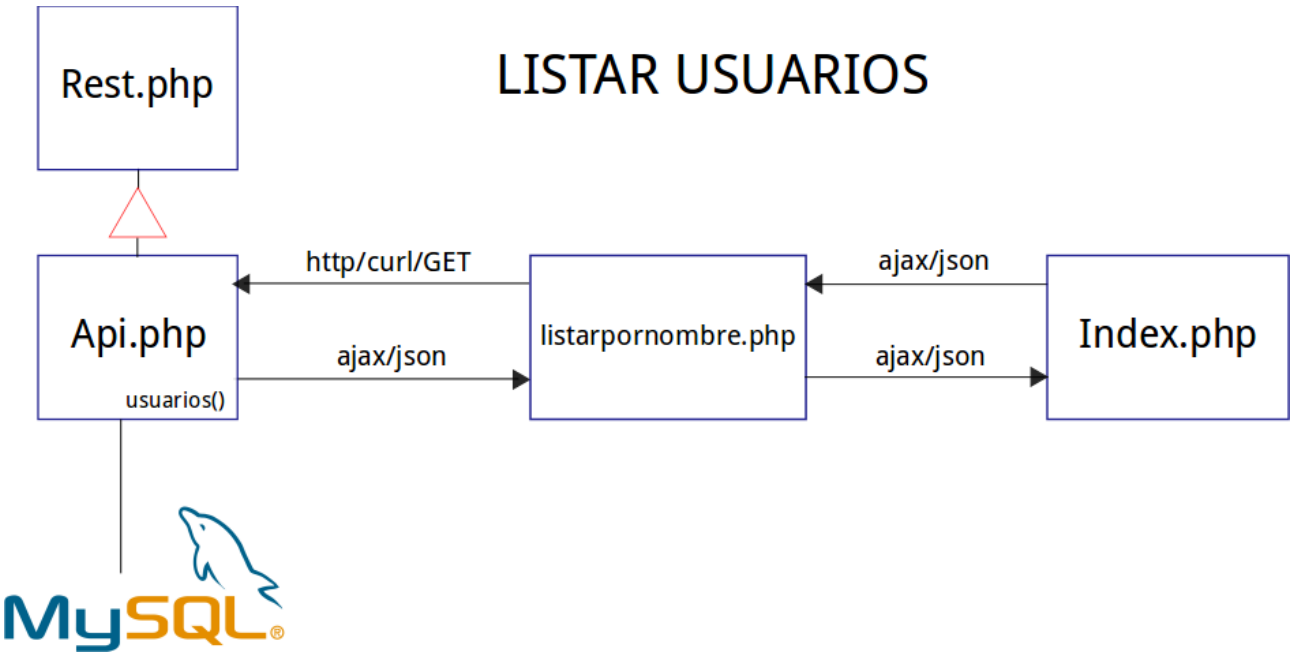
# Vayamos a nuestra aplicación

Realizamos una aplicación para hacer consultas a una base de datos que tenia usuarios. Las consultas que realizamos fueron:

- Listar Usuarios
- Crear Usuarios
- Comprobar existencia de usuario
- Actualizar nombre de usuario
- Borrar usuario

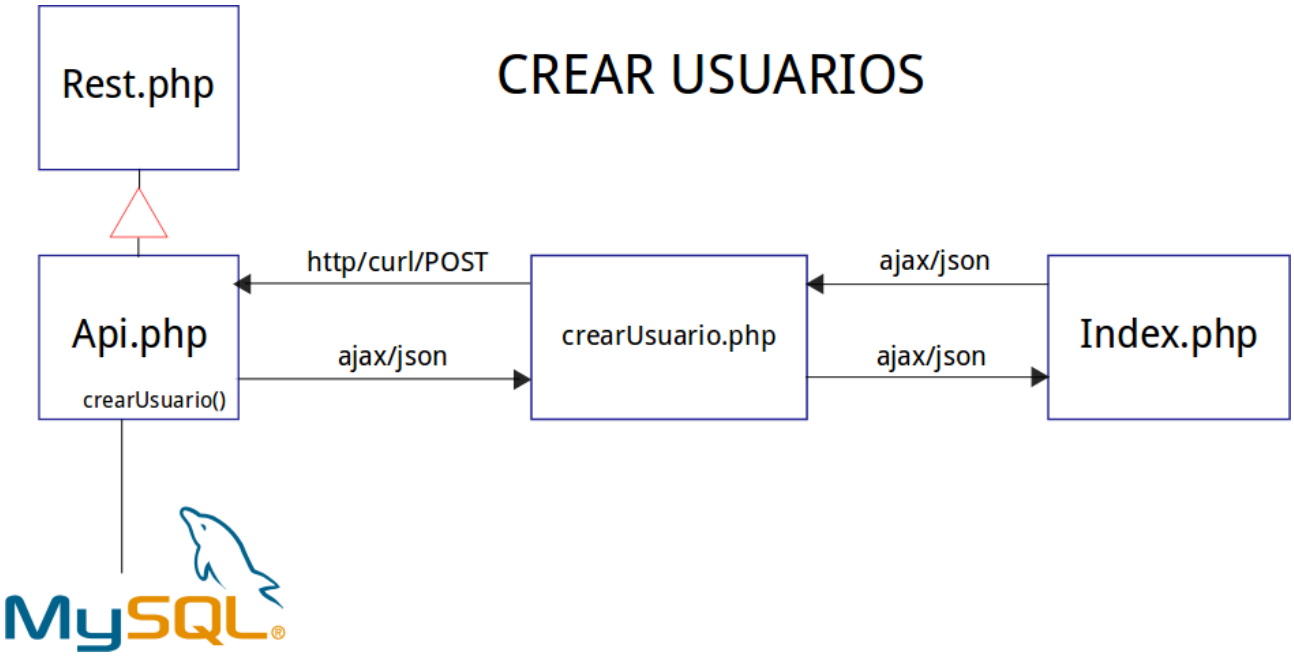
# Listar Usuarios:

Listar usuarios, es una función que a través de un boton en index.php nos envía por medio de ajax a listarpornombre.php quien se comunica con Api.php (a través de curl/HTTP) donde se obtiene el listado de todos los usuarios que se encuentra en la base datos, que por ajax vuelve a Index.php.



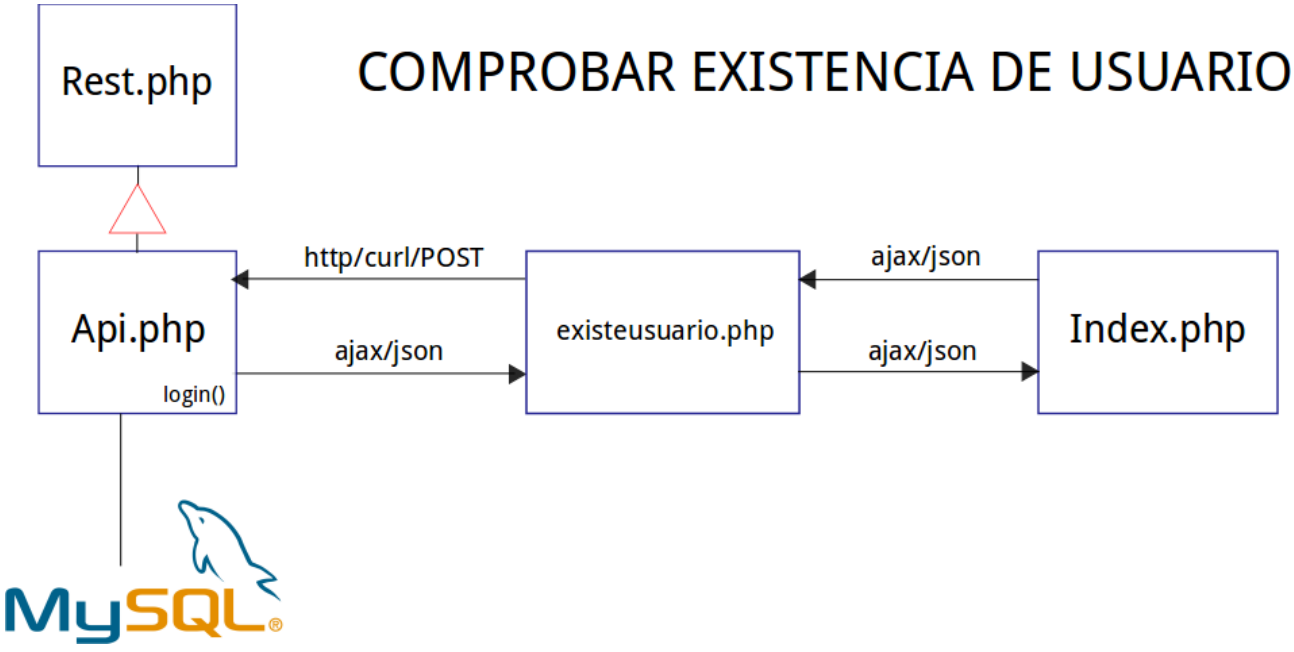
# Crear Usuario

Crear usuario, es una función que a través de un formulario en index.php nos envía por medio de ajax a crearUsuario.php quien se comunica con Api.php (a través de curl/HTTP) donde genera un usuario nuevo en caso de no existir (por medio de la comprobación si el mail ya existe en la base de datos) y lo ingresa en la base de datos. Devuelve a index.php un mensaje de cargado.



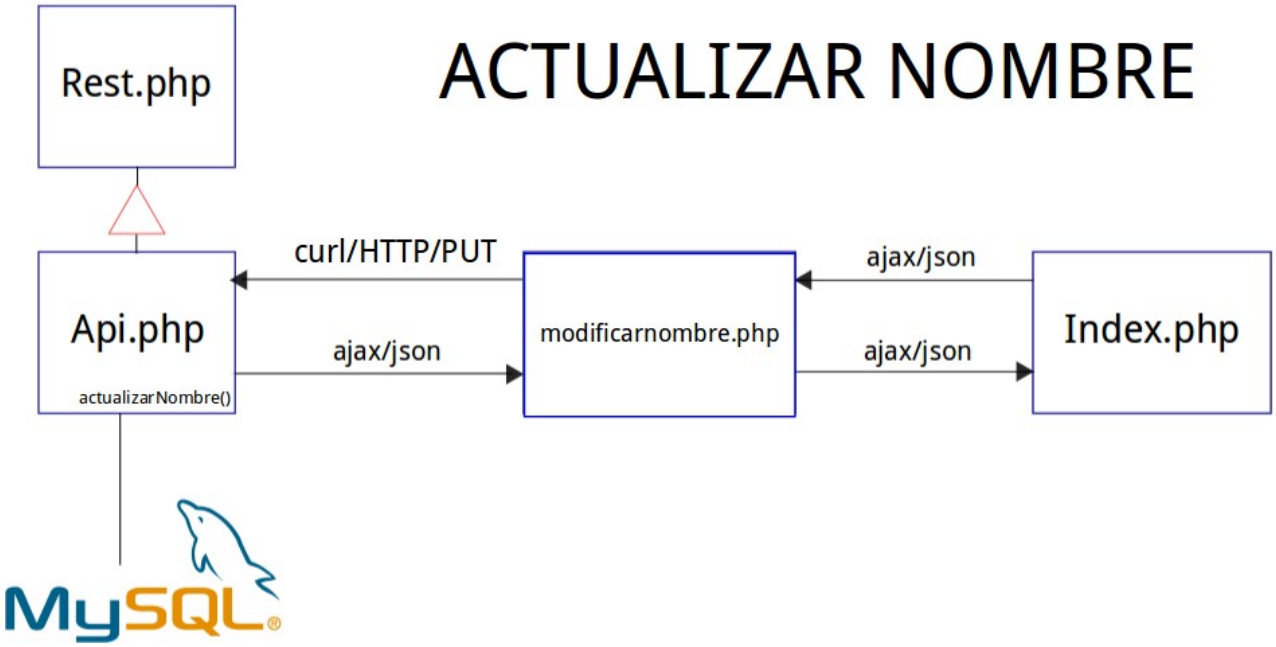

# Comprobar existencia de usuario

Comprobar existencia de usuario, es una función que a través de un formulario en index.php nos envía por medio de ajax a existeusuario.php quien se comunica con Api.php (a través de curl/HTTP) comprueba si existe el mail y la clave en la base de datos luego devuelve por medio de ajax un mensaje a el index.php.



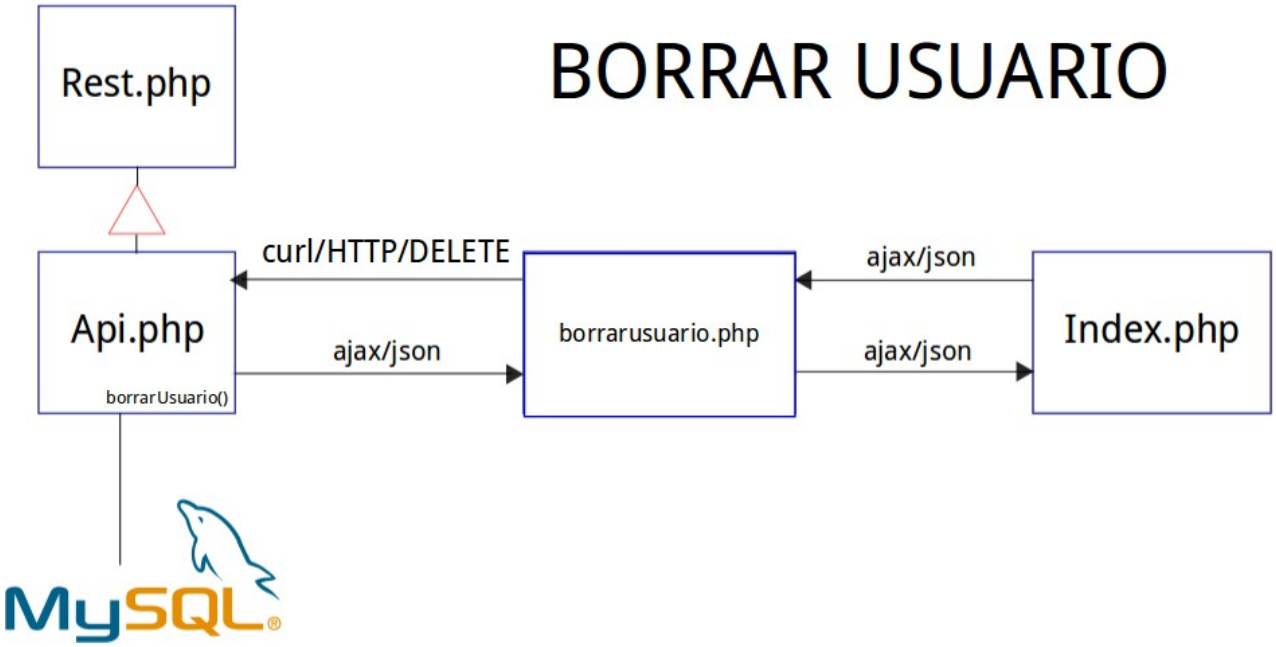
# Actualizar nombre de usuario

Actualizar nombre de Usuario, es una función que a través de un formulario en index.php nos envía por medio de ajax a modificarnombre.php quien se comunica con Api.php (a través de curl/HTTP) y cambia el nombre en un registro de la base de datos. Luego devuelve en Json un mensaje corroborando.




# Borrar usuario

Borrar Usuario, es una función que a través de un formulario en index.php con un combo, nos envía por medio de ajax a borrarusuario.php quien se comunica con Api.php (a través de curl/HTTP) y elimina el registro de la base de datos, enviando al finalizar un mensaje de borrado index.php.



# Conclusión

En conclusión REST junto con Ajax/json nos permiten hacer aplicaciones robustas y no tan complicadas, mejorando la comunicación Cliente-Servidor usando de interfaz Rest (con todo lo que conlleva – HTTP- GET, PUT, etc) ordenando en URI para el acceso a las distintas funciones.

# INDICE

# Bibliografía:

REST :https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

HIPERMEDIA: https://es.wikipedia.org/wiki/Hipermedia

ARQUITECTURA DE SOFTWARE: https://es.wikipedia.org/wiki/Arquitectura_de_software

Tutorial 1 REST + JSON + PHP:
http://programandolo.blogspot.com.ar/2013/08/ejemplo-php-de-servicio-restful-parte-1.html

Tutorial 2 REST + JSON + PHP:
http://programandolo.blogspot.com.ar/2013/08/ejemplo-php-de-servicio-restful-parte-2.html

Introducción a REST: https://diego.com.es/introduccion-a-rest-en-php