

[jc-mouse.net](http://jc-mouse.net)

# Crea un servicio web REST con PHP y MYSQL

8-10 minutu

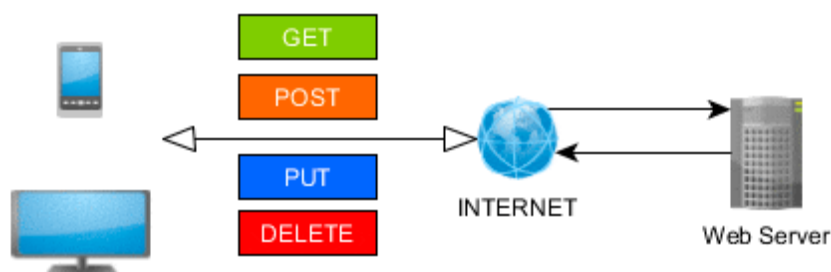
## Servicio Web

Un servicio web (*en inglés, Web Service o Web services*) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Entre los estandares empleados en servicios web tenemos a **REST**

## ¿Que es REST?

*REST* o Transferencia de Estado Representacional (*Representational State Transfer*) es una arquitectura de desarrollo web que se apoya en el estándar HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos. Para el intercambio de información se usa *XML* o *JSON* este ultimo más utilizado actualmente.



En este tutorial construiremos un **REST Api** empleando el lenguaje

**PHP** de lado del servidor, empleando las operaciones *GET*, *POST*, *PUT* y *DELETE* equiparables a las operaciones **CRUD** de base de datos (*Crear, leer, actualizar y borrar*) el intercambio de datos e realizar mediante **JSON** (*JavaScript Object Notation*) la base de datos sera MySQL.

### ¿Que necesitamos?

- IDE Netbeans 8.x
- Servidor web (XAMPP, Appserv, etc) Nosotros usaremos XAMPP
- [Insomnia REST Client](#)
- Navegador Google Chrome

Conocimientos necesarios sobre PHP, SQL y JSON

### Base de Datos

La base de datos que usaremos en este tutorial se llama “**dbTest**” y cuenta solo con una tabla “**people**” y es la siguiente:

```
--  
-- Estructura de tabla para la tabla people  
--
```

```
CREATE TABLE people (  
  id int(10) NOT NULL,  
  name varchar(32) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

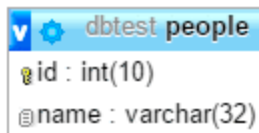
```
INSERT INTO people (id, name) VALUES  
(1, 'Leandro Gado'),  
(2, 'Aitor Tilla');
```

```
--  
-- Indices de la tabla people  
--
```

```
ALTER TABLE people  
  ADD PRIMARY KEY (id);
```

```
--  
-- AUTO_INCREMENT de la tabla people  
--  
ALTER TABLE people  
  MODIFY id int(10) NOT NULL AUTO_INCREMENT,  
  AUTO_INCREMENT=3;
```

Crea una base de datos y restaura el anterior código SQL



## Insomnia REST Client

[Insomnia](#) es una aplicación que se instala como complemento en el navegador **Google Chrome**, *Insomnia* te permite consumir los servicios de un **Api REST** de una manera sencilla y cómoda gracias a su interfaz gráfica.

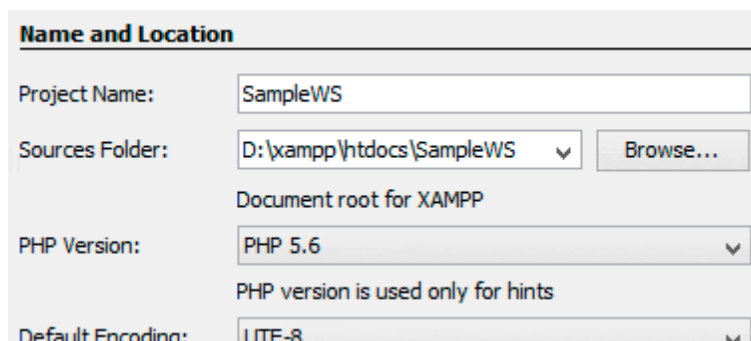
Descarga e instala este complemento en *Google Chrome*. A medida que avancemos con el tutorial, se hará uso de esta aplicación para testear el progreso del REST Api.

## PROYECTO APIREST

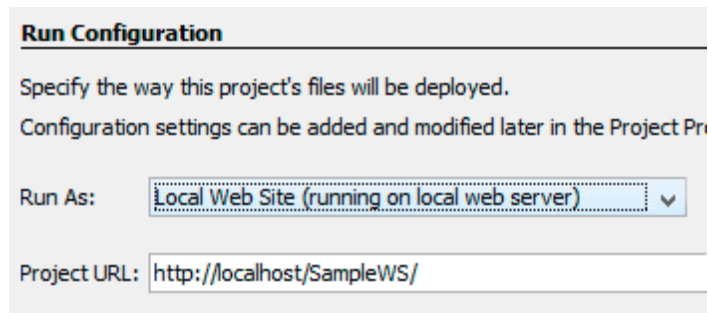
### Paso 1. El proyecto en Netbeans

Clic en menú **Archivo** → **Proyecto nuevo...** en *categorias* seleccionar **PHP** y en *proyecto* **PHP Application**, presiona el botón *Siguiente* para continuar.

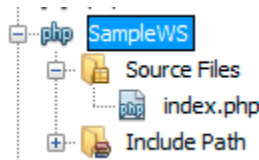
En la siguiente ventana, indicaremos el nombre del proyecto que es “**SampleWS**”, la ruta donde salvar los archivos “**D:\xampp\htdocs\SampleWS**”, la versión de **PHP 5.6** y la codificación de los archivos “**UTF-8**”. presionamos “*siguiente*”.



En la ventana de “**Run Configuración**” dejaremos el valor que nos da por defecto la *Project URL* “**http://localhost/SampleWS/**”, con esta *URL* es que podremos acceder al servicio web.



Las siguientes opciones “**PHP Frameworks**” y “**Composer**” no son necesarias para este proyecto, así que presionamos directamente en “**terminar**” para crear el proyecto.



## Paso 2. Clases

Clic derecho sobre la carpeta “**sources files**” → **Nuevo** → **PHP Class...** en nombre de archivo escribimos “**PeopleAPI**” y presionamos Terminar. Repetimos el paso anterior pero esta vez escribimos “**PeopleDB**”. Estas serán las dos clases que conformaran el Servicio Web.

## Paso 3. .HTACCESS

Un fichero **.htaccess** (*hypertext access*) es un fichero que permite definir diferentes directivas de configuración para cada directorio (*con sus respectivos subdirectorios*) sin necesidad de editar el archivo de configuración principal de Apache. Entre sus muchos usos el que nos interesa es el de la creación de **URL amigables** que nos permita implementar las diferentes acciones **CRUD** (*Create, Read, Update and Delete*) siguiendo las buenas practicas en el desarrollo de **APIREST**.

- Mantener una estructura lógica en las URL
- Usar sustantivos NO verbos

Para este tutorial, las *URL* que emplearemos son los siguientes:

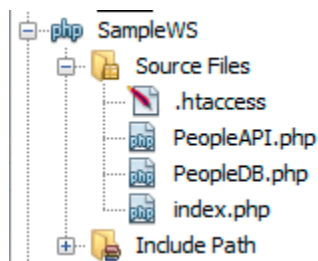
- GET /peoples → recuperara una lista de personas.
- GET /peoples/1 → recupera la información de una persona en específico.
- POST /peoples → Crea una nueva persona
- PUT /peoples/1 → Actualiza el registro con el ID 1
- DELETE /peoples/1 → Elimina el registro con ID 1

Un ejemplo de lo que NO se debe hacer:

- POST /personas/agregar <<NO usar verbos>>
- GET /personas/sillas/crustaceos <<wtf>>

Para crear un archivo *.htaccess* desde netbeans clic derecho en la carpeta raíz del proyecto “**Sources Files**” → **Nuevo** → **Otro...** en la ventana que aparece buscamos la categoría “**otras**” y en “**Tipo de Archivos**” seleccionamos “**Archivo vacío**” y presionamos siguiente. En nombre de archivo escribimos “*.htaccess*” y presionamos terminar para crear el archivo.

Nuestro proyecto hasta este punto debe lucir de la siguiente forma



Pega en el archivo *.htaccess* lo siguiente:

```
RewriteEngine On
RewriteRule ^([a-zA-Z_-]*)$ index.php?action=$1
RewriteRule ^([a-zA-Z_-]*)/([0-9]+)
index.php?action=$1&id=$2 [L,QSA]
```

Con este par de reglas nos aseguramos de que las solicitudes al servidor sean de la forma */accion* y */accion/idnumerico*

#### Paso 4. Código para el APIREST

## 4.1 Comenzando con el código php

Lo primero que haremos sera crear una estructura que nos permitirá según la solicitud del usuario realizar las diferentes acciones soportadas por el *API REST*

Abre el archivo **PeopleAPI** y pega el siguiente código:

```
class PeopleAPI {
    public function API(){
        header('Content-Type: application/JSON');
        $method = $_SERVER['REQUEST_METHOD'];
        switch ($method) {
            case 'GET'://consulta
                echo 'GET';
                break;
            case 'POST'://inserta
                echo 'POST';
                break;
            case 'PUT'://actualiza
                echo 'PUT';
                break;
            case 'DELETE'://elimina
                echo 'DELETE';
                break;
            default://metodo NO soportado
                echo 'METODO NO SOPORTADO';
                break;
        }
    }
}

} //end class
```

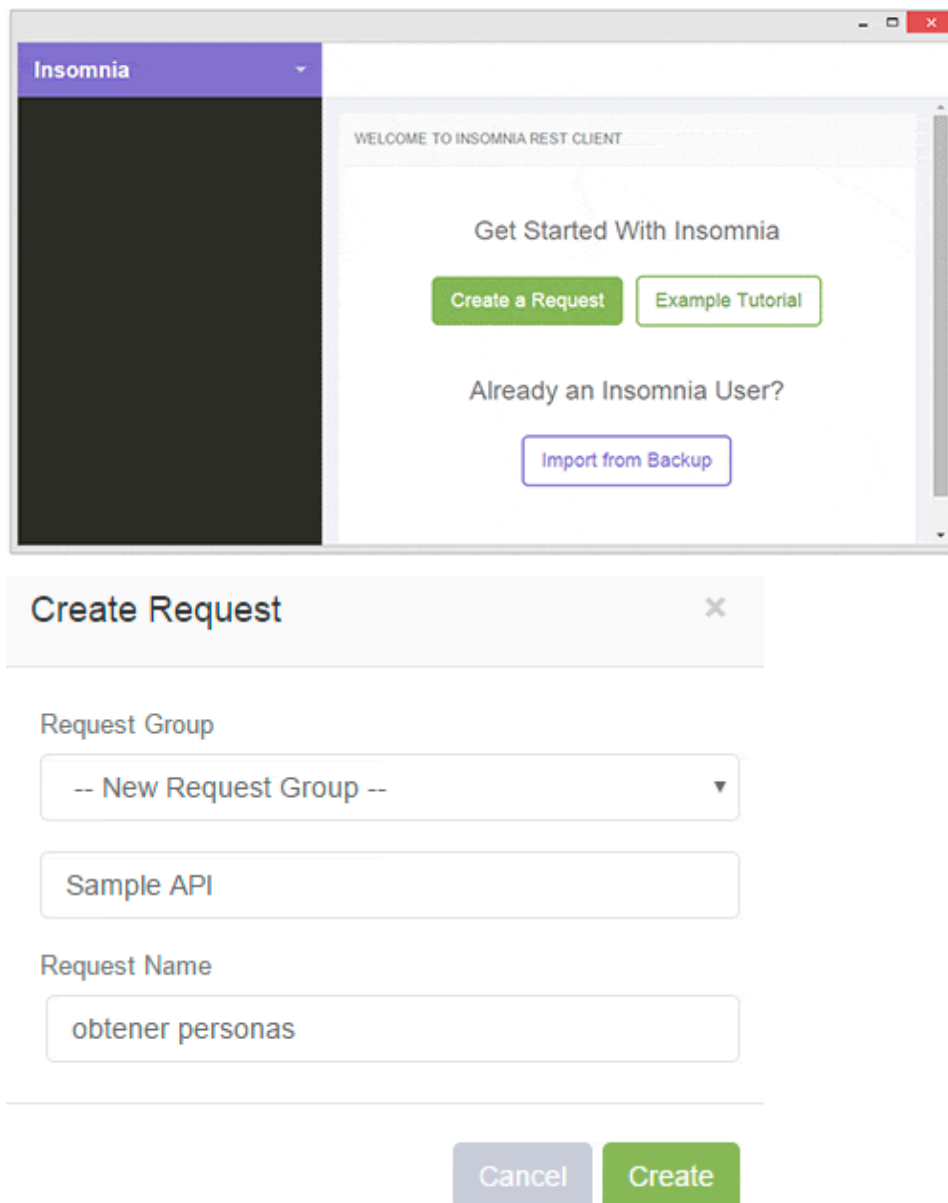
Abre el archivo **index.php** y reemplaza su contenido por:

```
<?php
    require_once "PeopleAPI.php";
    $peopleAPI = new PeopleAPI();
    $peopleAPI->API();
?>
```

## 4.2 Insomnia

Si ejecutamos el proyecto desde un navegador web, solo podremos ver que se imprime un *GET* en pantalla, si quisiéramos llamar a *POST*, *PUT* y *DELETE* tendríamos que escribir código php, pero para eso es que instalamos *Insomnia REST Client*.

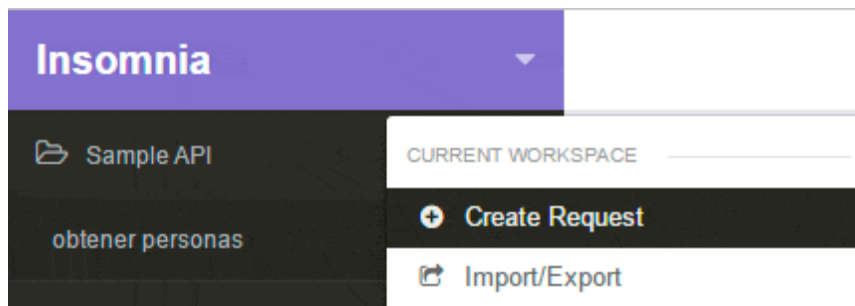
Inicia *Insomnia* desde el menú de aplicaciones de *Google Chrome*, a continuación presiona el boton “**create a request**” y escribe en *Request Group* “**Sample API**” y en *Request Name* “*obtener personas*” y presiona create



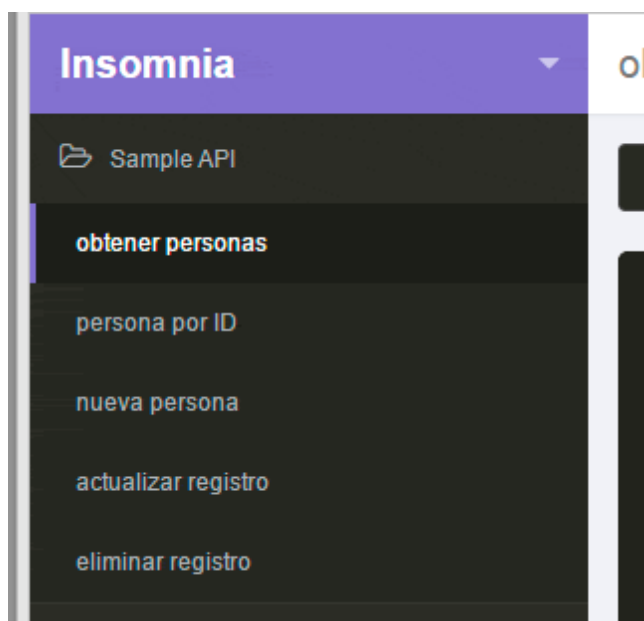
The image shows the Insomnia REST Client application window. The main window has a purple header with the 'Insomnia' logo. The main content area displays a 'WELCOME TO INSOMNIA REST CLIENT' message, followed by 'Get Started With Insomnia' and two buttons: 'Create a Request' (green) and 'Example Tutorial' (green). Below this, it asks 'Already an Insomnia User?' with an 'Import from Backup' button (purple). Below the main window, a 'Create Request' dialog box is open. It has a title bar 'Create Request' with a close button. The dialog contains three input fields: 'Request Group' with a dropdown menu showing '-- New Request Group --', 'Request Name' with the text 'Sample API', and 'Request Name' with the text 'obtener personas'. At the bottom of the dialog are two buttons: 'Cancel' (grey) and 'Create' (green).

Tenemos que escribir 4 solicitudes más (*request*) , clic en el triángulo a lado del nombre de insomnia → Create Request,

aparece el mismo recuadro anterior, y en esta ocasión escribe **“persona por ID”**.



Repite el paso anterior para crear nuevas solicitudes y escribe en *Request Name* **“nueva persona”**, **“actualizar registro”** y **“eliminar registro”**, te debe quedar algo así:



#### 4.3 Primer test API

Se puede mandar una solicitud con *Insomnia* en unos cuantos pasos:

- 1) Selecciona la solicitud
- 2) Seleccionar el verbo
- 3) Escribir la URL del servicio web
- 4) Si la solicitud requiere de información JSON, parámetros, autenticación o encabezados van en este espacio (*lo vemos con más detalle mas adelante*)
- 5) Para enviar la solicitud se presiona **SEND**



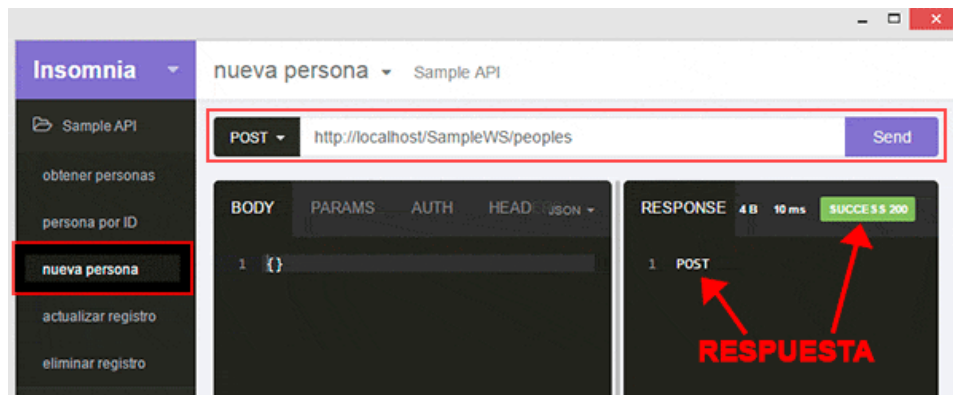
6) La respuesta del servidor aparecerá en este sector

[tutorial\_insomnia.gif]

Para cada una de las solicitudes que creamos, la configuración es la siguiente:

- obtener personas : GET : `http://localhost/SampleWS/peoples`
- persona por ID : GET : `http://localhost/SampleWS/peoples/1`
- nueva persona : POST: `http://localhost/SampleWS/peoples`
- actualizar registro : PUT : `http://localhost/SampleWS/peoples/1`
- eliminar registro : DELETE : `http://localhost/SampleWS/peoples/1`

Ve probando cada una de ellas, como en estos momentos no tenemos nada en nuestro api pues solo obtendremos como respuesta una cadena de texto que definimos en el código más arriba, por ejemplo para “**nueva persona**” tendremos:



Dejaremos las cosas hasta aquí para no hacer muy largo el post, en la segunda parte terminaremos de construir el REST Api.

[Descargar archivos](#)

Segunda Parte: [Crea un servicio web REST con PHP y MYSQL](#)

enjoy!!!