Índice

Sumario

RESTful Web Service en PHP (API Rest)

Antes de nada, vamos a ver una serie de conceptos para tener claro que es RESTful.

¿Qué es un web service?

Un web service es un programa, diseñado para el intercambio de información máquina a máquina, sobre una red. Se trata de una interfaz mediante la que dos máquinas (o aplicaciones) se comunican entre sí. Esto hace que un ordenador o programa pueda solicitar y recibir información de otro ordenador o programa. A quien solicita la información se le llama cliente y a quien envía la información se le llama servidor.
Esta tecnología se caracteriza por estos dos rasgos:

- Multiplataforma: cliente y servidor no tienen por qué contar con la misma configuración para comunicarse. El servicio web se encarga de hacerlo posible.
- Distribuida: por lo general, un servicio web no está disponible para un único cliente, sino que son diferentes los que acceden a él a través de Internet.

¿Qué es API?

La palabra es el acrónimo de **A**pplication **P**rogramming **I**nterface (Interfaz de Programación de Aplicaciones), y se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas. A diferencia de los web services, las API no necesariamente deben comunicarse entre una red, pueden usarse entre dos aplicaciones en un mismo ordenador.
Es decir, la API presenta funciones y reglas que permiten la interacción y comunicación entre diferentes aplicaciones.

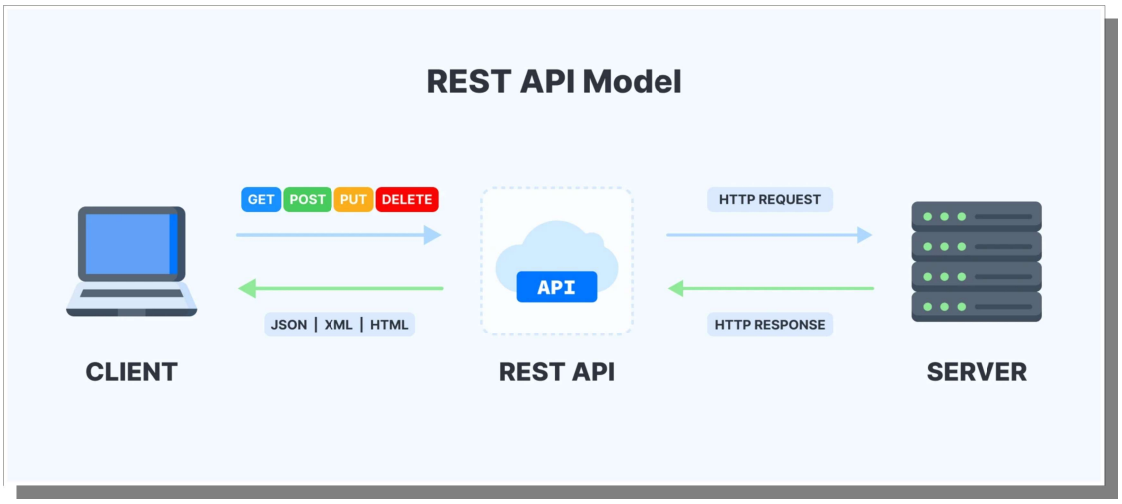
¿REST vs RESTful?

¿Qué es REST?
Es una arquitectura para diseñar API a través del protocolo HTTP. Sus siglas significan **RE**presentational **S**tate **T**ransfer. Su beneficio clave es su gran flexibilidad.

¿Qué es RESTful? RESTful web service o RESTful api, son programas basados en REST.

Pero muchas veces se usan como sinónimos (REST y RESTful).

Los desarrolladores utilizan la API REST siempre que sea necesario proporcionar datos al usuario de una aplicación o sitio web directamente desde el servidor.



Los componentes principales de la API REST:

- Cliente: Un cliente o programa lanzado en el lado del usuario (en su dispositivo) iniciando la comunicación.
- Servidor: Un servidor que utiliza API como acceso a sus funciones y datos.
- Recurso: Cualquier contenido (video, texto, imagen) que el servidor transmite al cliente.

¿Cómo funcionan las APIs REST?

La API REST se comunica a través de solicitudes HTTP y completa las siguientes funciones: crear, leer, actualizar y eliminar datos. También se conocen como operaciones CRUD (**C**reate – **R**ead – **U**pdate – **D**elete).
REST proporciona la información sobre los recursos solicitados y utiliza cuatro métodos para describir qué hacer con un recurso:

POST: creación de un recurso;
GET: obtener un recurso;
PUT: actualizar un recurso;
DELETE: eliminación de un recurso.

Un usuario de la API se denomina cliente. El cliente envía consultas a las APIs web para obtener datos o aplicar modificaciones al programa.

Un recurso puede ser cualquier información: documento, imagen, servicio temporal.
La información puede ser entregada al cliente en varios formatos: JSON, HTML, XLT, Python o texto plano. El más popular y usado es JSON porque es legible por humanos y máquinas, y es independiente del lenguaje.

Para acceder a un recurso, un cliente debe realizar una solicitud. Después de recibirlo, el servidor generará una respuesta con datos codificados sobre un recurso.

La estructura de la solicitud incluye cuatro componentes principales: el método HTTP (CRUD que mencionamos anteriormente), puntos finales, encabezados y cuerpo.

El **método HTTP** describe lo que se debe hacer con el recurso: POST, GET, PUT, DELETE.

El **punto final** contiene un URI (**U**niform **R**esource **I**dentifier o identificador uniforme de recursos), que indica cómo y dónde se puede encontrar el recurso. Una URL o ubicación uniforme de recursos es el tipo de URI más común y representa una dirección web completa.

Los **encabezados** contienen los datos relacionados con el cliente y el servidor. Los encabezados incluyen datos de autenticación: clave API, nombre, dirección IP que pertenece a la computadora en la que está instalado el servidor y también la información sobre el formato de respuesta.

El **cuerpo** se usa para enviar información adicional al servidor, como los datos que desea agregar. Igualmente, el cuerpo de la respuesta puede incluir datos relativos a los puertos de salida y a la lógica de salida.

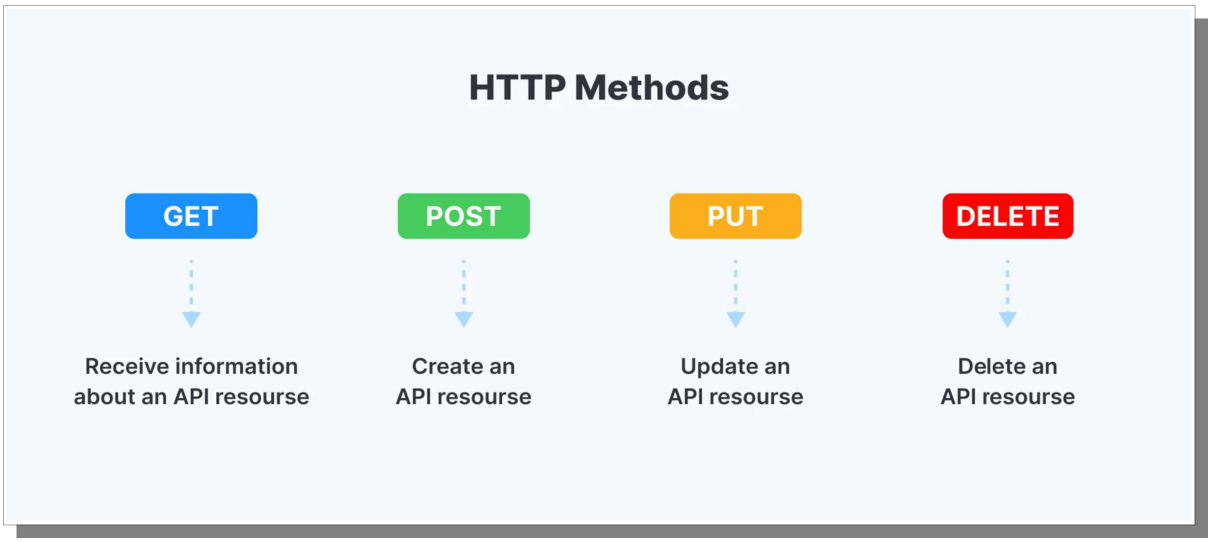
El programa que acepta el cuerpo de la solicitud del cliente utiliza un **servidor web**, que alberga los recursos que el consumidor solicita. El servidor puede comunicarse con los clientes a través de una API sin ofrecer a los clientes acceso inmediato a la información guardada en sus bases de datos. Cuando un usuario accede a los servicios web RESTful para enviar un cuerpo de solicitud, el servidor envía una representación normalizada del estado del recurso al navegador.

Esto significa que, de alguna manera, el servidor no envía el recurso exacto al cliente, sino que refleja su estado, es decir, la situación del recurso en un momento determinado. En otras palabras, las respuestas se devuelven en un formato ligero.

Métodos HTTP que usan

Cuando solicitamos una página web, podemos hacerlo por diferentes métodos. El más común es el GET, que es el que usamos cuando introducimos una dirección en nuestro navegador.
En otras ocasiones utilizamos POST, cuando enviamos un formulario con datos.
En cualquier caso, las aplicaciones pueden usar otros métodos. Resumiendo:

- Utilizar la petición GET para devolver los datos solicitados o los recursos solicitados.
- Utilizar la petición POST para hacer un nuevo recurso
- Utilice la instrucción PUT para realizar cambios o seguir actualizando un recurso existente
- Utilizar la instrucción DELETE para deshacerse de un recurso.



Algunos códigos de estado HTML

Cuando se recibe una página HTML, también se recibe un código de estado HTTP (solo uno) en los web service RESTful. Estos se usan para saber el estado de la ejecución del servicio.

Los códigos de estado HTTP se dividen en 5 «tipos». Se trata de agrupaciones de respuestas que tienen significados similares o relacionados. Las cinco clases incluyen:

- 100s: Códigos informativos que indican que la solicitud iniciada por el navegador continúa.
- 200s: Los códigos con éxito regresaron cuando la solicitud del navegador fue recibida, entendida y procesada por el servidor.
- 300s: Códigos de redireccionamiento devueltos cuando un nuevo recurso ha sido sustituido por el recurso solicitado.
- 400s: Códigos de error del cliente que indican que hubo un problema con la solicitud.
- 500s: Códigos de error del servidor que indican que la solicitud fue aceptada, pero que un error en el servidor impidió que se cumpliera.

Los códigos de estado HTTP más usados en los web service RESTful son:

- 200 (Ok), cuando la operación fue exitosa.
- 201 (Created), cuando se creó un registro (¿recuerdan? Con el método POST)
- 403 (Forbidden), cuando intentamos leer un registro para el que no tenemos acceso.
- 404 (Not found), cuando intentamos leer un registro que no existe.
- 500 (internal Server Error), cuando se produce un error al intentar acceder al servidor.

¿Para qué se utilizan las APIs REST?

Uno de los principales beneficios de los servicios web RESTful es que ofrece una gran flexibilidad, lo que le permite utilizar esta API RESTful de manera más eficaz. A continuación se muestran algunos ejemplos de para qué se utilizan las APIs REST.

Aplicaciones basadas en la nube

Una aplicación en la nube es un programa de software que combina componentes locales y basados en Internet. Un ejemplo de aplicación en la nube es Google Docs. Todo lo que se requiere para Google Docs es un ordenador que pueda ejecutar motores de búsqueda y servicios web de Internet. Las redes informáticas proporcionan la interfaz de usuario y las operaciones, incluido el almacenamiento de información.

Servicios en la nube

Los programadores utilizan los servicios web RESTful para conectarse a los servicios en la nube utilizando Internet. Un desarrollador crea un código que utiliza la API del proveedor de Internet, da las entradas y variables necesarias y luego comprueba la respuesta para asegurarse de que la acción funciona como se espera.

Los usuarios finales pueden acceder a las aplicaciones cliente o a los servicios web ofrecidos por los servicios web en la nube, como la infraestructura informática, los sistemas de almacenamiento o los sistemas de seguimiento, utilizando un servicio en la nube como los servicios web RESTful.

Estos servicios web se crean para ofrecer un acceso rápido y económico a herramientas y aplicaciones sin necesidad de hardware o infraestructura.

Código de PHP para el web service RESTful

Para este ejemplo vamos a asumir que tenemos creada la base de datos vista en documentos anteriores, **dwes_tarde_pruebas**, y necesitamos un servicio RESTful para acceder a los mensajes y editarlos, entonces nuestro servicio va a realizar estas operaciones:

- Listar todos los mensajes.
- Mostrar un mensaje.
- Agregar un mensaje.
- Eliminar un mensaje.
- Actualizar uno o más campos de un mensaje.

Ahora vamos a crear 3 archivos de PHP

- **config.php**: En este archivo se encuentra la configuración para la base de datos.
- **utils.php**: Funciones que hacen más fácil interactuar con la base de datos.
- **mensaje.php**: El código propio del web service RESTful.

Archivo config.php

Este es el archivo de configuración, debes asegurarte de que tiene todos los parámetros correctos para que pueda conectarse a tu base de datos

```php
<?php
$db = [
'host' => 'localhost',
'username' => 'dwes_tarde',
'password' => 'dwes_2223',
'db' => 'dwes_tarde_pruebas'
];
?>
```

Archivo utils.php

En este archivo de utilidades se encuentra las funciones para:

- Conectar a la base de datos.
- Obtener los parámetros.
- Añadir los parámetros a las consultas.
- Mostrar la salida con los datos recibidos

```php
<?php
//Abrir conexión a la base de datos
function conectar($db)
    {
        try
    {
        $mysql="mysql:host={$db['host']};dbname={$db['db'];charset=utf8";
        $conexion = new PDO($mysql, $db['username'],
        $db['password']);

        // set the PDO error mode to exception
        $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    }
        catch (PDOException $exception)
        {
            exit($exception->getMessage());
        }
        return $conexion;
    }
//Obtener parámetros
function getParams($params)
    {
        $parametros = [];
        foreach($params as $param => $valor)
        {
            $parametros[] = "$param=:$param";
        }
        return implode(", ", $parametros);
    }
    //Asociar todos los parámetros a un sql
function bindAllParams($consulta, $params)
    {
        foreach($params as $param => $value)
        {
            $consulta->bindValue(':'.$param, $value);
        }
        return $consulta;
    }
    //Mostrar los datos de salida con las cabeceras recibidas
function salidaDatos($data, $httpHeaders=array())
    {
        if (is_array($httpHeaders) && count($httpHeaders))
        {
        foreach ($httpHeaders as $httpHeader)
        {
        header($httpHeader);
        }
        }

        print $data;
        exit();
    }
?>
```

Archivo mensaje.php

Este es el archivo que implementa el servicio RESTful

```
<?php
include "config.php";
include "utils.php";
$dbConexion = conectar($db);
    // listar todos los mensajes o solo uno
if ($_SERVER['REQUEST_METHOD'] == 'GET')
    {
        if (isset($_GET['id'])){
        //Mostrar un mensaje
            $select = "SELECT * FROM mensajes where id=:id";
            $consulta = $dbConexion->prepare($select);
            $consulta->bindParam(':id', $_GET['id']);
            $consulta->execute();
            if ($consulta->rowCount() > 0) {
                salidaDatos (json_encode($consulta-
>fetch(PDO::FETCH_ASSOC)),
                array('Content-Type: application/json', 'HTTP/1.1 200
OK'));
            }   else{
                    salidaDatos('', array('HTTP/1.1 404 Not Found'));
                }
        } else {
            //Mostrar lista de mensajes
            $select = "SELECT * FROM mensajes";
            $consulta = $dbConexion->prepare($select);
            $consulta->execute();
            registros = [];
            while ($registro = $consulta->fetch(PDO::FETCH_ASSOC)) {
            $registros[] = $registro;
            }
            salidaDatos(json_encode($registros),
            array('Content-Type: application/json', 'HTTP/1.1 200 OK'));
            }
    }
    //crear un nuevo mensaje
if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {
        // Transformamos el JSON de entrada de datos a un array asociativo
        $datos = json_decode(file_get_contents('php://input'), true);
        $insert = "INSERT INTO mensajes(nombre, email, mensaje) VALUES
(:nombre, :email, :mensaje)";
        $consulta = $dbConexion->prepare($insert);
        bindAllParams($consulta, $datos);
        $consulta->execute();
        $mensajeId = $dbConexion→lastInsertId();
        if($mensajeId) {
            $datos['id'] = $mensajeId;
            salidaDatos(json_encode($datos),
            array('Content-Type: application/json', 'HTTP/1.1 200 OK'));
        }
    }
```

```php
    //borrar un mensaje
if ($_SERVER['REQUEST_METHOD'] == 'DELETE')
    {
        // Transformamos el JSON de entrada de datos a un array asociativo
        $datos = json_decode(file_get_contents('php://input'), true);
        $id = $datos['id'];
        $delete = "DELETE FROM mensajes where id=:id";
        $consulta = $dbConexion->prepare($delete);
        $consulta->bindParam(':id', $id);
        $consulta->execute();
        if ($consulta->rowCount() > 0) {
        salidaDatos('', array( 'HTTP/1.1 200 OK'));
        }   else {
                salidaDatos('', array('HTTP/1.1 404 Not Found'));
            }
        exit();
    }
        //actualizar un mensaje
if ($_SERVER['REQUEST_METHOD'] == 'PUT')
    {
        // Transformamos el JSON de entrada de datos a un array asociativo
        $datos = json_decode(file_get_contents('php://input'), true);
        $mensajeId = $datos['id'];
        $campos = getParams($datos);
        $update = "UPDATE mensajes SET $campos WHERE id='$mensajeId'";
        $consulta = $dbConexion->prepare($update);
        bindAllParams($consulta, $datos);
        $consulta->execute();
        salidaDatos('', array( 'HTTP/1.1 200 OK'));
    }
//En caso de que ninguna de las opciones anteriores se haya ejecutado
        salidaDatos('', array('Content-Type: application/json', 'HTTP/1.1
400 Bad Request'));
?>
```

Respecto a este último archivo (mensaje.php): En RESTful la acción a ejecutar depende mucho del método HTTP que utilizas, entonces me baso en la variable **$_SERVER['REQUEST_METHOD']**, para determinar si se está haciendo una petición GET, POST, DELETE o PUT.

Con la función **salidaDatos**, puedo enviar información y códigos de respuesta, para informar si el servicio se ejecutó con éxito o con algún error.

Con la sentencia **json_encode** de php, se transforma la salida a un formato JSON, lo cual es necesario en los servicios RESTful.

Con la sentencia **json_decode** de php, recojo los datos de entrada en formato JSON,

Cómo probar el servicio RESTful

Los servicios RESTful no se pueden probar en un navegador, porque no hay forma de enviar peticiones PUT o DELETE, entonces podemos usar un programa especial como postman.
Podéis ver el tutorial siguiente:
https://desarrolloweb.com/articulos/como-usar-postman-probar-api