

1.- Configurar el **http.conf** de apache y cambiar para la carpeta raíz:

“AllowOverride none” por “AllowOverride all”

```
#<Directory "${SRVROOT}/htdocs">
DocumentRoot "${ALLROOT}/www"

<Directory "${ALLROOT}/www">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride all

#
# Controls who can get stuff from this server.
#
Require all granted
</Directory>
```

3.- Descomentar la siguiente línea

```
#LoadModule rewrite_module modules/mod_rewrite.so
# Así
LoadModule rewrite_module modules/mod_rewrite.so
```

4.- Reiniciar el servicio de apache

5.- Redireccionar todo al index.php, para tener soporte de url amigables:
Crear archivo **.htaccess** en la carpeta donde pondremos el WS (para ambiente de desarrollo Scriptcase lo ponemos en la carpeta de la aplicación, pero scriptcase

elimina el archivo cada vez que se generan fuentes). Incluiremos una función que permita crear el archivo .htaccess en la carpeta de la aplicación cuando ésta es cargada sin incluir ningún parámetro o método.

```
DirectoryIndex index.php
RewriteEngine On

# Si la ruta no es un archivo existente, ni una carpeta
# Reescribir al index
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.+)?/?$ index.php?url=$1 [L,QSA]
```

6.- Utilidades de Rest para reusar desde una librería interna de SC:

Estructura json sugerida para la respuesta del servicio:

```
{
  "status" : "success",
  "data" : null,
  "code" : 200,
  "message" : "datos consultados exitosamente"
}
```

Status: retornar el estado de la petición realizada.

success : cuando se completa la petición y se retorna la respuesta esperada.

warning : puede ser por el uso de un método no permitido

error : Error en base de datos o cualquier exención en el proceso.

Data : Este llevará los datos de respuesta en un array, siempre debe de incluirse en la respuesta, en caso de no retornar valor se le asigna el valor null.

Code: Este representa el código de respuesta del servidor.

Los códigos más frecuentes son:

- 200 OK : Cuando todo sale bien.
- 201 Created (Creado) : Cuando se hace un post y todo sale bien.
- 304 Not Modified (No modificado)
- 400 Bad Request (Error de consulta) : Si la petición no se puede ejecutar por problema en los parámetros recibidos.
- 401 Unauthorized (No autorizado)
- 403 Forbidden (Prohibido)
- 404 Not Found (No encontrado) : Cuando el ID contenido en la consulta (GET, PUT o PATCH) no existe.
- 405 Method not Allowed : Si el verbo en la petición no es permitido en el servicio.

- 422 (Unprocessable Entity (Entidad no procesable))
- 500 Internal Server Error (Error Interno de Servidor)

Informational Status Codes	Client Request Incomplete	Server Errors
<p>100 — Continue [The server is ready to receive the rest of the request.]</p> <p>101 — Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]</p>	<p>400 — Bad Request [The server detected a syntax error in the client's request.]</p> <p>401 — Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]</p> <p>402 — Payment Required [reserved for future.]</p> <p>403 — Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]</p> <p>404 — Not Found [The requested document does not exist on the server.]</p> <p>405 — Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]</p> <p>406 — Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]</p> <p>407 — Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]</p> <p>408 — Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]</p> <p>409 — Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]</p> <p>410 — Gone [The requested resource is permanently gone from the server.]</p> <p>411 — Length Required [The client must supply a Content-Length header in its request.]</p> <p>412 — Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]</p> <p>413 — Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]</p> <p>414 — Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]</p> <p>415 — Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]</p> <p>417 — Expectation Failed [The server failed to meet the requirements of the Expect request-header.]</p>	<p>500 — Internal Server Error [A server configuration setting or an external program has caused an error.]</p> <p>501 — Not Implemented [The server does not support the functionality required to fulfill the request.]</p> <p>502 — Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]</p> <p>503 — Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]</p> <p>504 — Gateway Time-Out [The gateway or proxy has timed out.]</p> <p>505 — HTTP Version Not Supported [The version of HTTP used by the client is not supported.]</p>
Client Request Successful	Request Redirected	Unused status codes
<p>200 — OK [Success! This is what you want.]</p> <p>201 — Created [Successfully created the URI specified by the client.]</p> <p>202 — Accepted [Accepted for processing but the server has not finished processing it.]</p> <p>203 — Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]</p> <p>204 — No Content [Request is complete without any information being sent back in the response.]</p> <p>205 — Reset Content [Client should reset the current document. I.e. A form with existing values.]</p> <p>206 — Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]</p>	<p>300 — Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]</p> <p>301 — Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]</p> <p>302 — Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]</p> <p>303 — See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]</p> <p>304 — Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the specified date, and the client should use a cached copy.]</p> <p>305 — Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]</p> <p>307 — Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]</p>	<p>306- Switch Proxy</p> <p>416- Requested range not satisfiable</p> <p>506- Redirection failed</p>

HTTP protocol version 1.1 Server Response Codes

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>

Chart created September 5, 2000 by Suso Banderas(suso@suso.org). Most of the summary information was gathered from Appendix A of "Apache Server Administrator's Handbook" by Mohammed J. Kabir.

Message: se retorna un string con un mensaje, para informar o aclarar las condiciones del servicio.

Para garantizar consistencia en las respuestas de todas las funciones que retorne la estructura json. Es necesario invocar la función con los 4 parámetros correspondientes a las propiedades que fueron descrita anteriormente. Crearemos una serie de funciones orientada a estandarizar cada una de las funcionalidades que necesitamos

para servir una API Rest, para eso crearemos una librería interna ([ws_metodos.php](#)) de SC con los siguientes segmentos de códigos.

```
function getResponse($status, $data, $code, $message)
{
    if (empty($data) || is_null($data))
        $data1=null;
    else
        $data1 = convertirUTF8($data);

    http_response_code($code);

    return json_encode(array(
        "status" => $status,
        "data" => $data1,
        "code" => $code,
        "message" => utf8_encode( $message)
    ));
}
```

Convertir el formato de codificación de caracteres de matrices (array) a utf-8. Evita que la función json_encode retorne una cadena vacía al encontrarse con caracteres especiales en cualquier elemento de la matriz:

```
function convertirUTF8($array){
    if (!is_null($array) && is_array($array))
        array_walk_recursive($array, function(&$item,$key){
            if(!mb_detect_encoding($item,'utf-8',true)){
                $item = utf8_encode( $item );
            }
        });
    return $array;
}
```

Métodos o Verbos mas usados en los servicios web:

GET : Obtener datos (SELECT).

POST : Para crear o insertar uno o varios registros (INSERT)

PUT : Actualización (update) completo de registro(s).

PATCH : Actualización (update) parcial parcial de registro(s).

DELETE : Eliminar un registros (por lo general es preferible solo marcarlo como borrado o lo que podríamos llamar un borrado lógico, es decir tener un campo bandera para utilizarlo como marca de borrado).

Función para consulta SQL y retorna una respuesta estándar (**GET**):

```

function ws_read($check_sql){
    $status = "success";
    $data = array();
    $code = 200;

    sc_set_fetchmode(0);
    sc_select(ds, $check_sql);

    if (false == {ds}) {
        $status = "error";
        $message={ds_erro};
        $code = 500;
    } elseif ($ds->EOF) {
        $status = "warning";
        $message='Not record Found!';
        $code = 404;
        $ds->Close();
    } else {
        while (!$ds->EOF){
            // Arreglo asociativo de cada una de las filas.
            $data[] = $ds->getRowAssoc(false);

            // Paso a la siguiente fila
            $ds->moveNext();
        }
        $message="record found!";
        $ds->Close();
    }
    return getResponse($status, $data, $code, $message);
}

```

Función para crear o insertar registros y retornar una respuesta estándar (**POST**):

```

function ws_insert($insert_table, $insert_fields, $seRetornald) {
    $id = null;
    $status = "success";
    $data = array();
    $code = 201;
    $message="record inserted!";

    $insert_sql = 'INSERT INTO ' . $insert_table
        . ' (' . implode(', ', array_keys($insert_fields)) . ') '
        . ' VALUES (' . implode(', ', array_values($insert_fields)) . ')';

    if ($seRetornald==true) {
        // Versión MySQL
        // $insert_sql .="; SELECT LAST_INSERT_ID() as lastId;";

        // Versión MS-SQL Server
        $insert_sql .="; SELECT SCOPE_IDENTITY() as lastId;";
    }

    sc_set_fetchmode(0);
    sc_select(ds, $insert_sql);

    if ({ds}===false) {
        $status = "error";
        $message={ds_erro};
        $code = 500;
    } elseif ($seRetornald==true) {
        //$data[] = $ds->getRowAssoc(false);
        $data[] = array('lastId' => (int)$ds->fields[0], );
        $ds->Close();
    }

    return getResponse($status, $data, $code, $message);
}

```

Función para modificar o actualizar registros y retornar una respuesta estándar (**PUT**):

```

function ws_update($update_table, $update_fields, $update_where) {
    $status = "success";
    $data = null;
    $code = 200;
    $message="record updated!";

    $update_fields[] = "modified_at = getdate()";

    $update_sql = 'UPDATE ' . $update_table
        . ' SET ' . implode(' ', $update_fields)
        . ' WHERE ' . $update_where;

    sc_set_fetchmode(0);
    sc_select(ds, $update_sql);

    if ({ds}===false) {
        $status = "error";
        $message={ds_errro};
        $code = 500;
    } else {
        $ds->Close();
    }

    return getResponse($status, $data, $code, $message);
}

```

Función para borrar o eliminar registros y retornar una respuesta estándar (**DELETE**):

```

function ws_delete($delete_table, $delete_where, $soloMarcar ) {
    $status = "success";
    $data = array();
    $code = 200;
    $message= ($soloMarcar==true)?"Record marked to be deleted!":"record deleted!";

    if ($soloMarcar==true) {
        $delete_fields = array("deleted_at = getdate()");
        $delete_where .= " and deleted_at is null";

        $delete_sql = 'UPDATE ' . $delete_table
            . ' SET ' . implode(', ', $delete_fields)
            . ' WHERE ' . $delete_where;
    } else {
        $delete_sql = 'delete ' . $delete_table
            . ' WHERE ' . $delete_where;
    }

    sc_set_fetchmode(0);
    sc_select(ds, $delete_sql);

    if ({ds}===false) {
        $status = "error";
        $message={ds_erro};
        $code = 500;
    } else {
        $ds->Close();
    }

    return getResponse($status, $data, $code, $message);
}

```

Es necesario colocar cada una de estas funciones en una librería interna, para que estos códigos puedan ser reusados por todas las aplicaciones de nuestro proyecto.

Las siguientes dos funciones debemos incluirla también en nuestra librería interna, puesto que la utilizaremos en el script de entrada de nuestro servicio:


```

function ws_getRequest() {
    $oret = new stdClass();
    $oret->verb = $_SERVER['REQUEST_METHOD'];

    return $oret;
}

function ws_htaccess() {
    $file = '.htaccess';

    //Use the function is_file to check if the file already exists or not.
    if(!is_file($file)){
        $contents =
            'DirectoryIndex index.php'. PHP_EOL
            .'RewriteEngine On'. PHP_EOL
            .''. PHP_EOL
            .'# Si la ruta no es un archivo existente, ni una carpeta'. PHP_EOL
            .'# Reescribir al index'. PHP_EOL
            .'RewriteCond %{REQUEST_FILENAME} !-f'. PHP_EOL
            .'RewriteCond %{REQUEST_FILENAME} !-d'. PHP_EOL
            .'RewriteRule ^(.+)?/?$ index.php?url=$1 [L,QSA]';

        file_put_contents($file, $contents);
    }
}

```

Ya tenemos en una librería interna todas las funciones necesarias para que nuestra API Rest pueda funcionar, ahora crearemos una aplicación blank que servirá como punto de entrada para las peticiones

7.- Aplicación blank como punto de entrada de nuestra API:

Creemos una aplicación blank en este caso el nombre sería **ws_cliente**. Debemos vincular la librería interna [ws_metodos.php](#) con la nueva aplicación blank.

Agregamos el siguiente código en el onExecute:

```

// Evento OnExecute

// sitios permitidos para interactuar con el servicio
header('Access-Control-Allow-Origin: *');
// Métodos o acciones permitidas
header("Access-Control-Allow-Methods: PUT, GET, POST, DELETE");
// Tipo de contenido que retornará como respuesta y su código de caracteres
header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

// Generar archivo .htaccess si es necesario.
ws_htaccess();

// Obtener el tipo de petición
$rq = ws_getRequest();

switch ($rq->verb) {

    // GET ----- //
    case 'GET':
        if (isset($_GET['idcliente'])) { // id = Retornar un cliente específico.
            echo readone($_GET['idcliente']);

        } elseif (isset($_GET['all'])) { // all = Retornar todos los clientes
            echo read();

        } else {
            die("API cliente activada.");
        }
        break;

    // POST ----- //
    case 'POST':
        // Obtener todo el body de la petición
        $data = json_decode(file_get_contents("php://input"));

        if ( empty($data->nombre) ) {

```

```

        echo getResponse('warning', null, 400, 'imcomplete post data');

    } else {
        echo create($data);
    }
    break;

    // PUT ----- //
case 'PUT':
    $update_fields = array();
    $update_where = "";

    // Obtener todo el body de la petición
    $data = json_decode(file_get_contents("php://input"));

    foreach($data as $key => $value)
    {
        if ( trim(strtoupper($key)) == "idcliente" && !empty($value) ) {
            // Si es clave primaria armamos el where
            $update_where = "$key = '$value'";
        } else {
            // Agregamos cada campo con su valor en un arreglo.
            $update_fields[] = "$key = '$value'";
        }
    }
    if (empty($update_where)) {
        echo getResponse('warning', null, 400, 'Key field is required for
update');
    } elseif ( empty($update_fields) ){
        echo getResponse('warning', null, 400, 'fields are required for
update');
    } else {
        echo update($update_fields,$update_where);
    }
    break;

    // DELETE ----- //
case 'DELETE':
    // Obtener todo el body de la petición
    $data = json_decode(file_get_contents("php://input"));

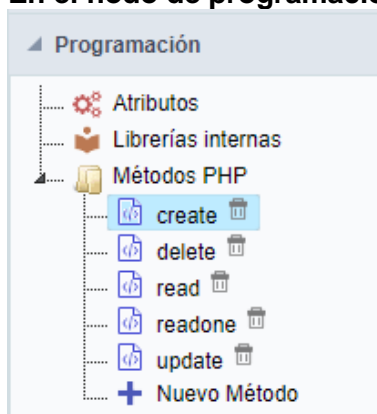
    if ( !isset($data->idcliente) || empty($data->idcliente) ) {
        echo getResponse('warning', null, 400, 'Key field is required for
delete');
    } else {
        echo delete($data->idcliente);
    }

```

```
        break;

    default:
        echo getResponse('warning', null, 405, 'Method not allow');
}
```

En el nodo de programación crearemos los siguientes métodos PHP



create(\$pdata)

```
// SQL statement parameters
$insert_table = "tblcliente";
$insert_fields = array( 'idcliente' => "$pdata->idcliente",
                       'nombre' => "$pdata->nombre",
                       );
return ws_insert($insert_table,$insert_fields,true);
```

delete(\$idcliente)

```
$where_stmt = "idcliente = '$idcliente'";

return ws_delete('tblcliente', $where_stmt,false);
```

read()

```
$sqlselect = "SELECT idcliente
               ,nombre
               FROM tblcliente";
return ws_read($sqlselect);
```

readone(\$idcliente)

```
$sqlselect = "SELECT idcliente
               ,nombre
               FROM tblcliente
               where idcliente= '$idcliente'";

return ws_read($sqlselect);
```

update(\$update_fields, \$update_where)

```
return ws_update('tblcliente', $update_fields, $update_where);
```

Este sería todo el código necesario. Para que se pueda crear el archivo .htaccess es necesario ejecutar la aplicación blank, por lo menos una sola vez desde el navegador sin pasar parámetros, para que se active el servicio REST, luego a partir de ese momento puede ser consumido el servicio REST. Para consumir el servicio REST

desde Scriptcase existen videos muy buenos en youtube, pero este servicio puede ser consumido desde cualquier otro lenguaje de programación o plataforma. La tabla utilizada (tblcliente) es una tabla con una estructura simple de dos campos con el objetivo de hacer mas corto el ejemplo, para el correcto funcionamiento de este ejemplo es necesario crear dicha tabla, revise la función **ws_insert** en la librería interna para que la adecue a MySQL o SQL Server o cualquier otro motor de base de datos.