



## TALLER No 6 DESARROLLO API CON PHP:

### OBJETIVOS DEL TALLER.

1. Comprender qué son las API, REST para usar en PHP
2. Desarrollar una API en PHP aplicando los estándares establecidos
3. Construir FRONTEND para consumir los servicios de la API

### EVIDENCIA(S) A ENTREGAR:

EV1 Desarrollar el reto propuesto en el taller

### CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	JOSE FERNANDO GALINDO SUAREZ	INSTRUCTOR	CGMLTI	8/05/2021

### CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					

## INTRODUCCIÓN.

Las API o interfaz de programación de aplicaciones, son un conjunto de funciones que se ofrecen mediante bibliotecas de programación para su uso.

Para entender el concepto de API debemos leer los siguientes términos:

- **Interfaz:** capa de extracción para que las aplicaciones se comuniquen.
- **API:** del inglés Application Programming Interface, que permite compartir datos y procesos entre ellos.
- **Arquitectura de software:** Es la forma como está diseñado un sistema, como se comunican al interior, como están organizados sus componentes.
- **Servicio WEB:** Permite una comunicación entre aplicaciones de una red usando el protocolo HTTP.
- **REST:** Es una arquitectura, del inglés Representational State Transfer o Representación de transferencia de estado, permitiendo que datos puede acceder, revisar o manipular otra aplicación
- **XML:** Formato tradicional para enviar datos, del inglés Extensible Markup Language.
- **JSON:** Formato actual para transferir datos, del inglés JavaScript Object Notation
- **TOKEN:** las API pueden ser públicas o privadas, esta última requiere de una autenticación que al hacerlo te devuelve un token que es un objeto que contiene datos de la autenticación y su formato es JWT.

## Tipos de API

- **Locales:** Se ejecutan dentro del mismo entorno. Por ejemplo cuando se desarrolla en Android y se quiere que el celular vibre, entonces debo utilizar la API del smartphone.
- **Remotas:** Cuando se consumen datos que están en otro lugar
  - **Servicios WEB**
    - **SOAP** (Simple Object Access Protocol)
    - **REST** (Representational state transfer) y al utilizarlo lo llamamos RESTFUL



Cada recurso que se consuma tiene un identificador único (URI); cuando se consulta un recurso el servidor puede contestar con los siguientes códigos:

- **2XX:** Todo fue exitoso
- **3XX:** Significan redirecciones
- **4XX:** Solicitudes invalidas
- **5XX:** Errores directamente en el servidor

[Fuente](#)

## MÉTODOS HTTP

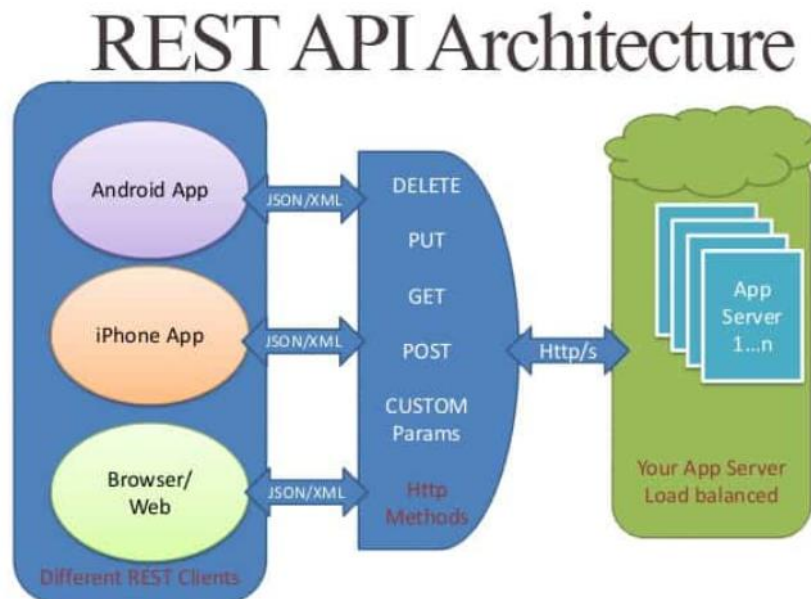


- **GET:** Este método se emplea para leer una representación de un resource. En caso de respuesta positiva (200 OK), devuelve la representación en un formato concreto: HTML, XML, JSON o imágenes, JavaScript, CSS, etc. [Leer mas](#)
- **POST:** Se utiliza POST por las limitaciones de GET. En caso de respuesta positiva devuelve 201 (*created*). Los POST requests se envían normalmente con formularios [Leer mas..](#)

- **PUT:** Utilizado normalmente para actualizar contenidos, pero también pueden crearlos. Tampoco muestra ninguna información en la URL. En caso de éxito devuelve 201 (*created*, en caso de que la acción haya creado un elemento) o 204 (*no response*, si el servidor no devuelve ningún contenido). A diferencia de POST es idempotente, si se crea o edita un resource con PUT y se hace el mismo request otra vez, el resource todavía está ahí y mantiene el mismo estado que en la primera llamada. Si con una llamada PUT se cambia aunque sea sólo un contador en el resource, la llamada ya no es idempotente, ya que se cambian contenidos. [Leer mas...](#)
- **DELETE:** Simplemente elimina un resource identificado en la URI. Si se elimina correctamente devuelve 200 junto con un body response, o 204

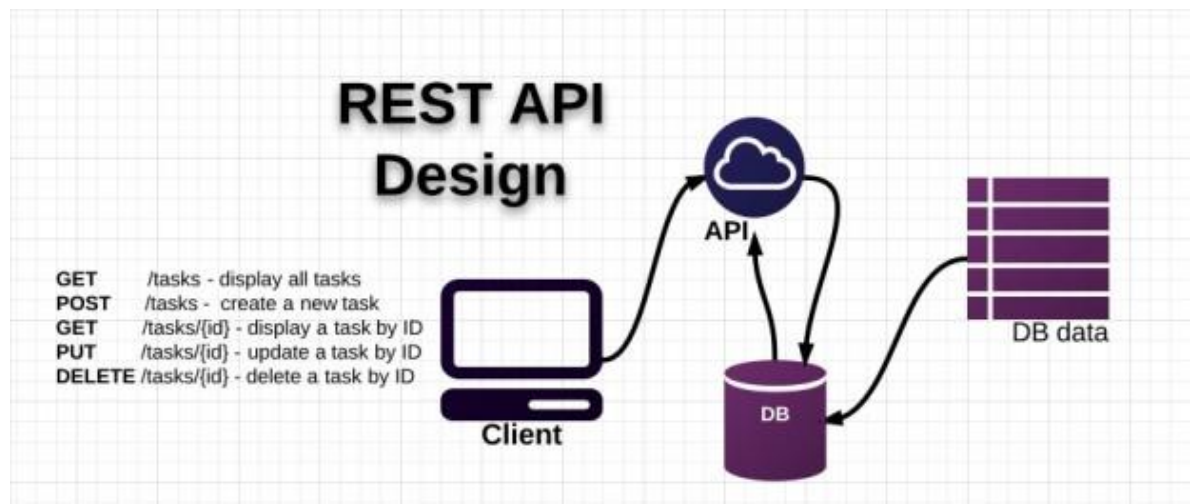
sin body. DELETE, al igual que PUT y GET, también es idempotente, [Leer más...](#)

## 1.API REST EN EL BACKEND.



Fuente: <https://shareurcodes.com/blog/creating%20a%20simple%20rest%20api%20in%20php>

REST API		
Crear usuario	POST	/usuarios
Obtener un usuario	GET	/usuarios/{id}
Obtener usuarios	GET	/usuarios
Actualizar usuario	PUT	/usuarios/{id}
Eliminar usuario	DELETE	/usuarios/{id}



Descargar los archivos desde [GITHUB](https://github.com/fegasu/api) para la práctica o utilizando el comando:  
“`git clone https://github.com/fegasu/api`”, desde git bash o descargue [GITHUB-DESKTOP](#).

Crear un archivo llamado “usuario.php”.

```
1  <?php
2  //echo $_SERVER["REQUEST_METHOD"];
3  switch($_SERVER["REQUEST_METHOD"]){
4      case 'GET':
5          echo "Obtener un usuario o usuarios";
6          break;
7      case 'POST':
8          echo "Crear un usuario";
9          break;
10     case 'PUT':
11         echo "Actualizar un usuario";
12         break;
13     case 'DELETE':
14         echo "Borrar un usuario";
15         break;
16     }
17
```



ARC

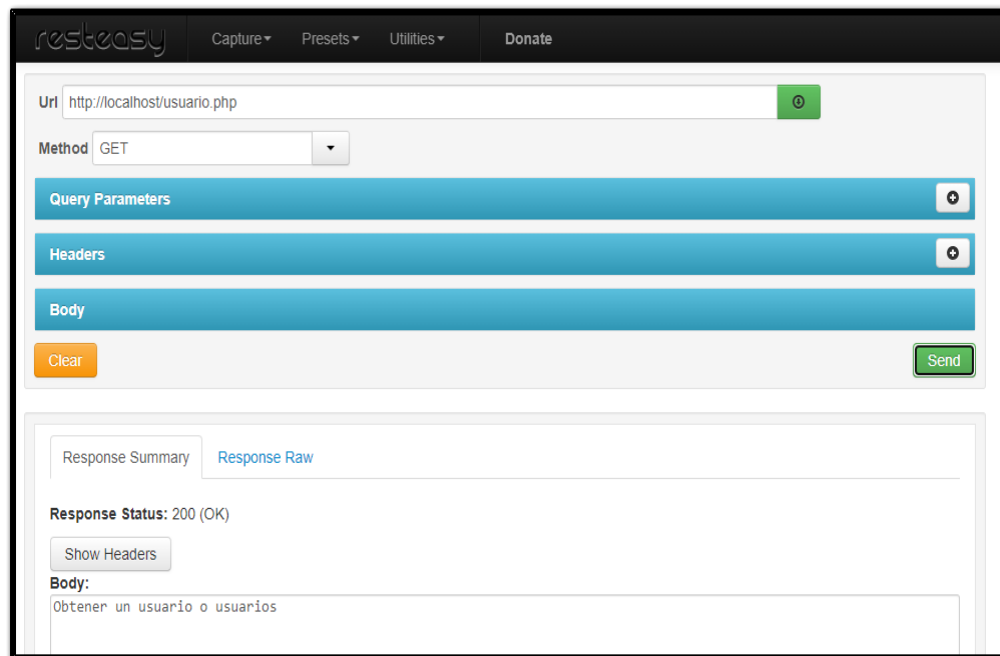


Postman



RestEasy

Utilizar las diferentes herramientas disponibles para probar las API-REST, en el desarrollo de este taller, se recomienda descargar la aplicación [RESTEASY](#) y ejecutarla así:





Servicio Nacional de Aprendizaje  
Formato Taller  
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

resteasy Capture Presets Utilities Donate

Url

Method

Query Parameters

Headers

Body

Response Summary [Response Raw](#)

Response Status: 200 (OK)

Body:  
Crear un usuario

resteasy Capture Presets Utilities Donate

Url

Method

Query Parameters

Headers

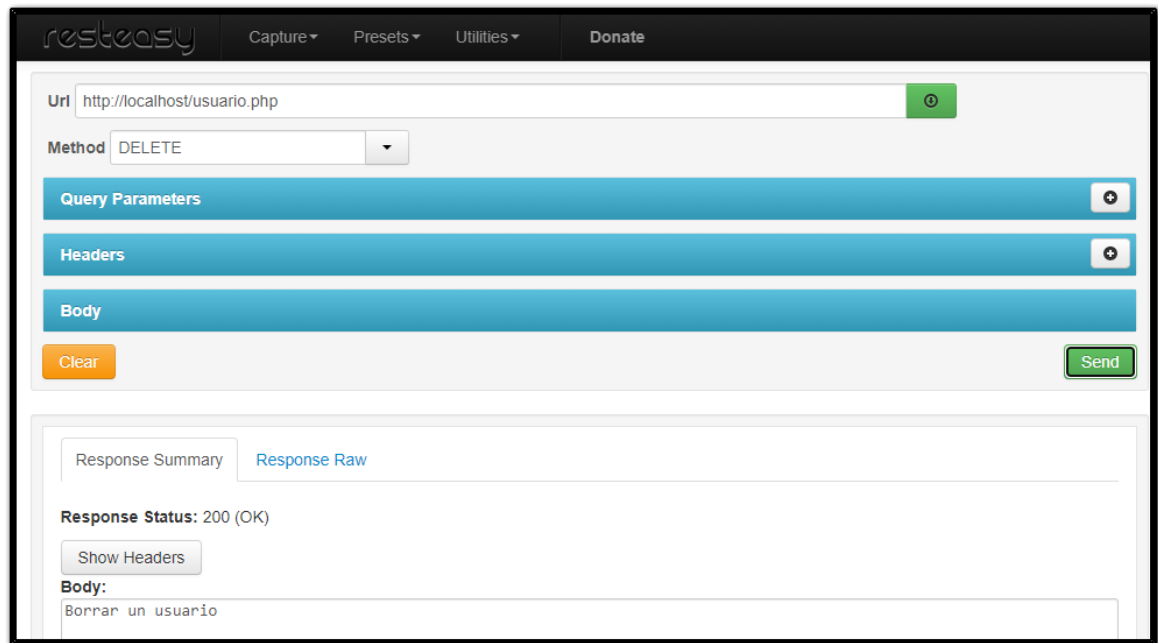
Body

Response Summary [Response Raw](#)

Response Status: 200 (OK)

Body:  
Actualizar un usuario





```
<?php
class SqlitePDO{
var $Cnx;
var $Rs;
var $u;
function __construct($Bd="SALUD.db"){
try {
$this->Cnx= new PDO('sqlite:'.$Bd);
} catch (Exception $e) {die ($e);}
}

function Ejecutar($Sentencia){
try{
$this->Rs = $this->Cnx->prepare($Sentencia.';')
or die(SQLITE_ERROR.' '.$Sentencia);
$this->Rs->execute();
}catch (Exception $e){ die($e);}
}
```

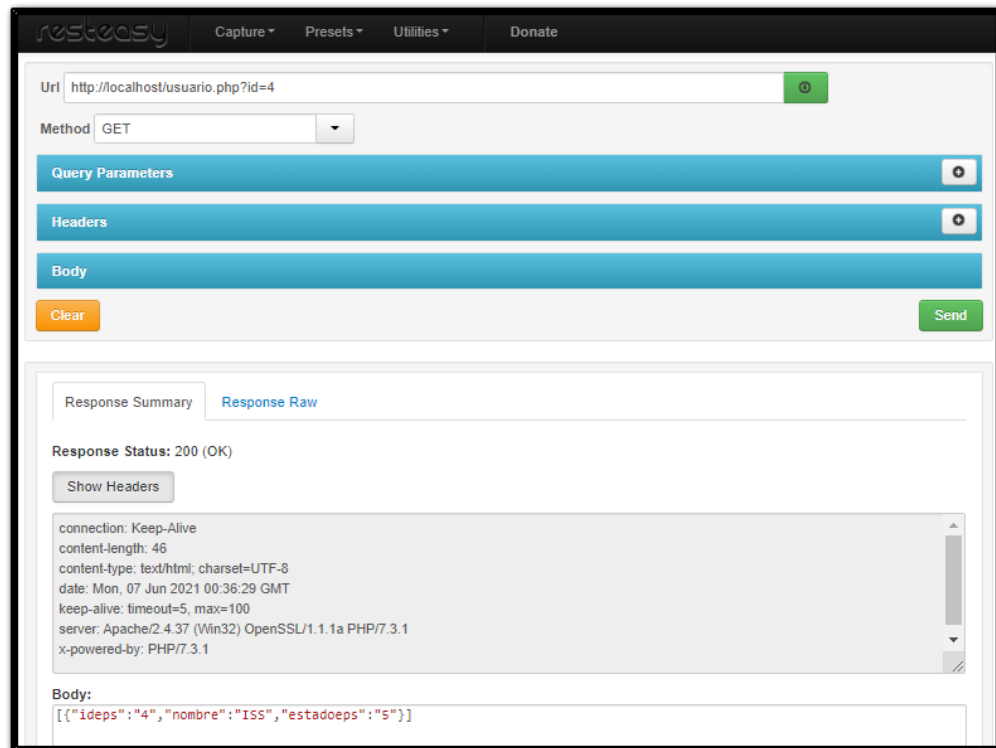
```
function EjecutarJSON($Sentencia){  
    $this->Rs = $this->Cnx->query($Sentencia,PDO::FETCH_OBJ);  
    return json_encode($this->Rs);  
}  
  
function CargarJSON(){  
    $i=0;  
    $rawdata = array();  
    foreach ($this->Rs as $row) {  
        $rawdata[] = $row;  
    }  
    return json_encode($rawdata);  
}  
  
function CargarArray(){  
    $i=0;  
    $rawdata = array();  
    foreach ($this->Rs as $row) {  
        $rawdata[] = $row;  
    }  
    return $rawdata;  
}  
  
function Cargar(){  
    $this->u = $this->Rs->fetch();  
    return ($this->u );  
}  
  
function getdato($col){  
    //$a=$this->Rs->fetchColumn($col);  
    $a=$this->u[$col];  
    return $a;  
}  
}
```

Modificamos el archivo “usuario.php” que será la API que en el transcurso del taller se construye y que después se renombra como “MIAPI.php”.



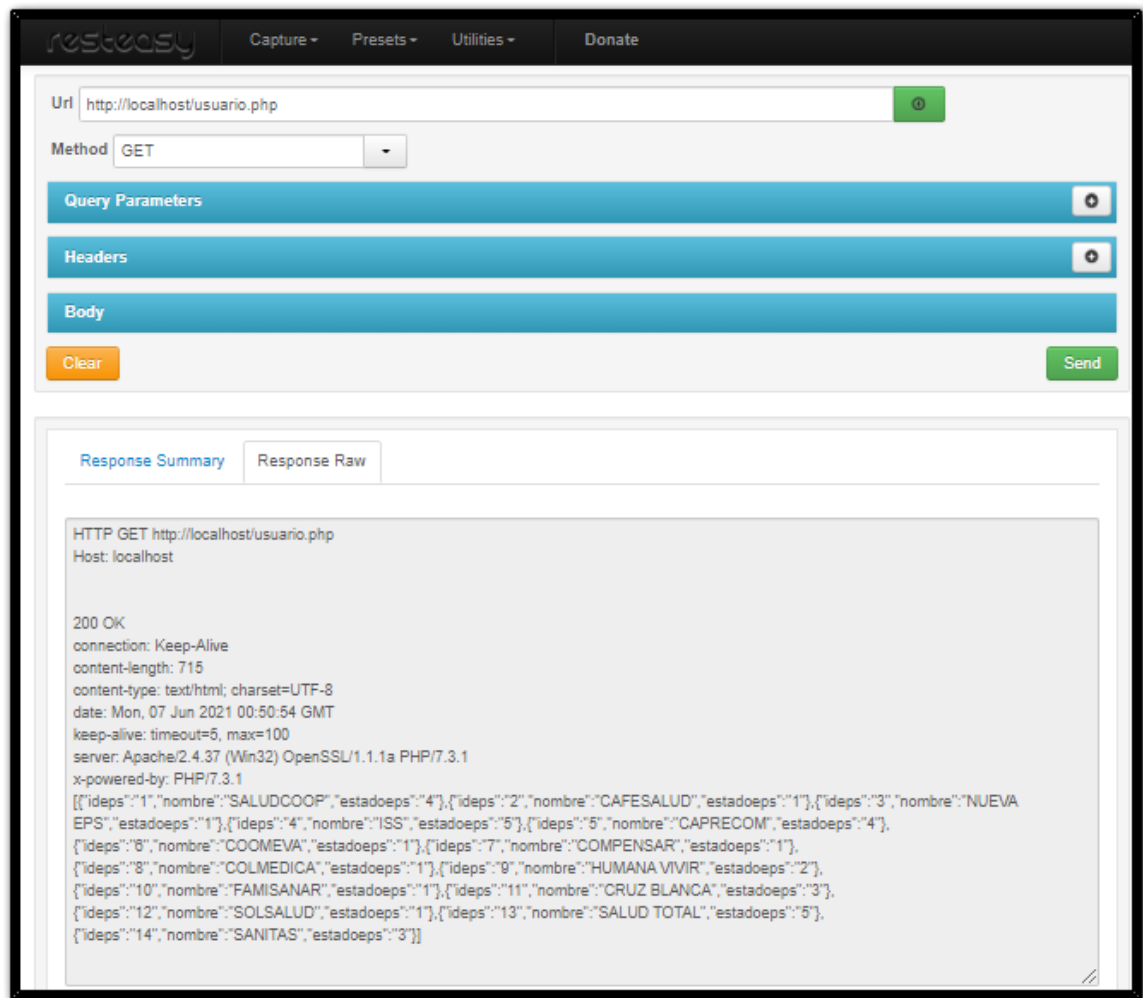
```
<?php
include "cnx.php";
$cn=new SQLitePDO();
switch($_SERVER["REQUEST_METHOD"]){

    case 'GET':
        if(isset($_GET["id"]))
            $sql="SELECT * FROM EPS WHERE IDEPS=".$_GET["id"];
        else
            $sql="SELECT * FROM EPS";
        $cn->EjecutarJSON($sql);
        echo $cn->CargarJSON();
        break;
```





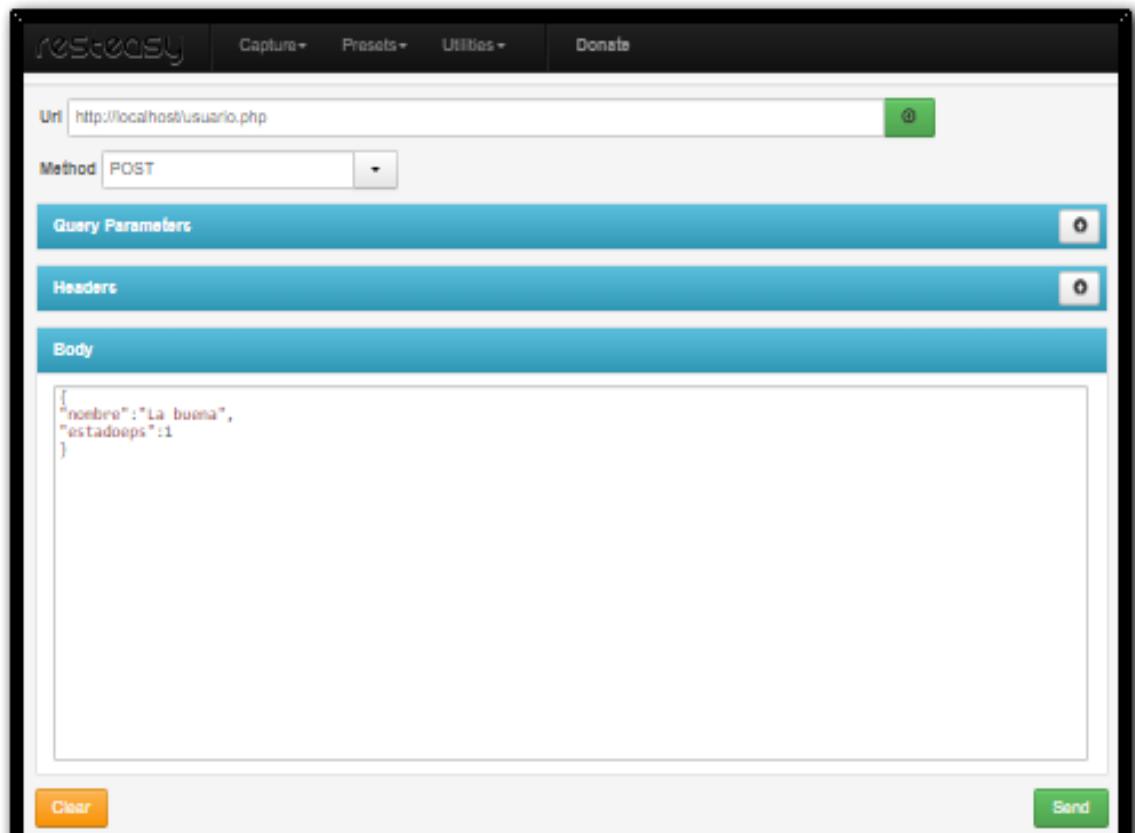
Servicio Nacional de Aprendizaje  
Formato Taller  
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.



Cambiar el archivo “usuario.php” para crear una nueva EPS



```
case 'POST':  
    $_POST=json_decode(file_get_contents("php://input"),true);  
    $sql="INSERT INTO EPS(NOMBRE,ESTADOEPS) VALUES('"  
        .$_POST["nombre"]."',".$_POST["estadoeps"].")";  
    $cn->Ejecutar($sql);  
    break;
```





DB Browser for SQLite - C:\xampp\htdocs\SALUD\SALUD.db

Archivo Editar Ver Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshabilitar

Estructura de la Base de datos Navegar Datos Editar Pragmas Ejecutar SQL

Tabla: EPS

	ideps	nombre	estadoeps
	Filtro	Filtro	Filtro
1	1	SALUDCOOP	4
2	2	CAFESALUD	1
3	3	NUEVA EPS	1
4	4	ISS	5
5	5	CAPRECOM	4
6	6	COOMEVA	1
7	7	COMPENSAR	1
8	8	COLMEDICA	1
9	9	HUMANA VIVIR	2
10	10	FAMISANAR	1
11	11	CRUZ BLANCA	3
12	12	SOLSALUD	1
13	13	SALUD TOTAL	5
14	14	SANITAS	3
15	15	La buena	1



```
case 'PUT':  
    $_POST=json_decode(file_get_contents("php://input"),true);  
    $sql="UPDATE EPS SET NOMBRE='".$$_POST["nombre"].  
    "','ESTADOEPS='".$$_POST["estadoeps"]." WHERE IDEPS='".$$_POST["id"]";  
    $cn->Ejecutar($sql);  
    break;
```

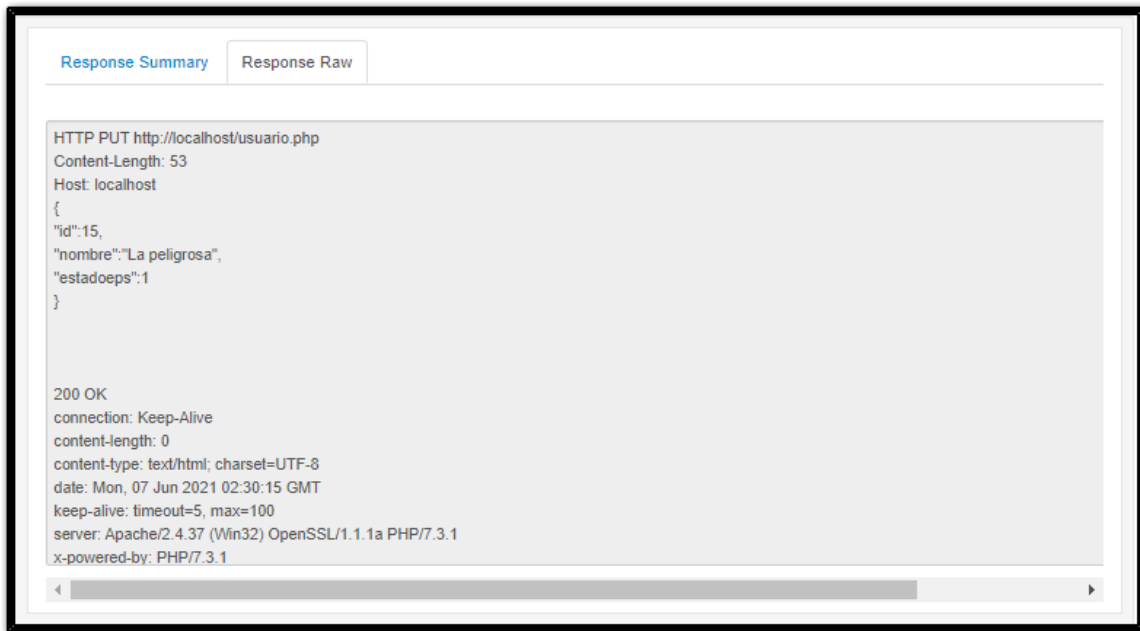


The screenshot shows the RESTEasy web application interface. At the top, there is a navigation bar with the 'resteasy' logo and links for 'Capture', 'Presets', 'Utilities', and 'Donate'. Below this, the 'Url' field is set to 'http://localhost/usuario.php'. The 'Method' dropdown is set to 'PUT'. There are three expandable sections: 'Query Parameters', 'Headers', and 'Body'. The 'Body' section is expanded, showing a JSON payload: 

```
{
  "id": 15,
  "nombre": "La peligrosa",
  "estadoeps": 1
}
```

. At the bottom left is a 'Clear' button, and at the bottom right is a 'Send' button.







DB Browser for SQLite - C:\xampp\htdocs\SALUD\SALUD.db

Archivo Editar Ver Ayuda

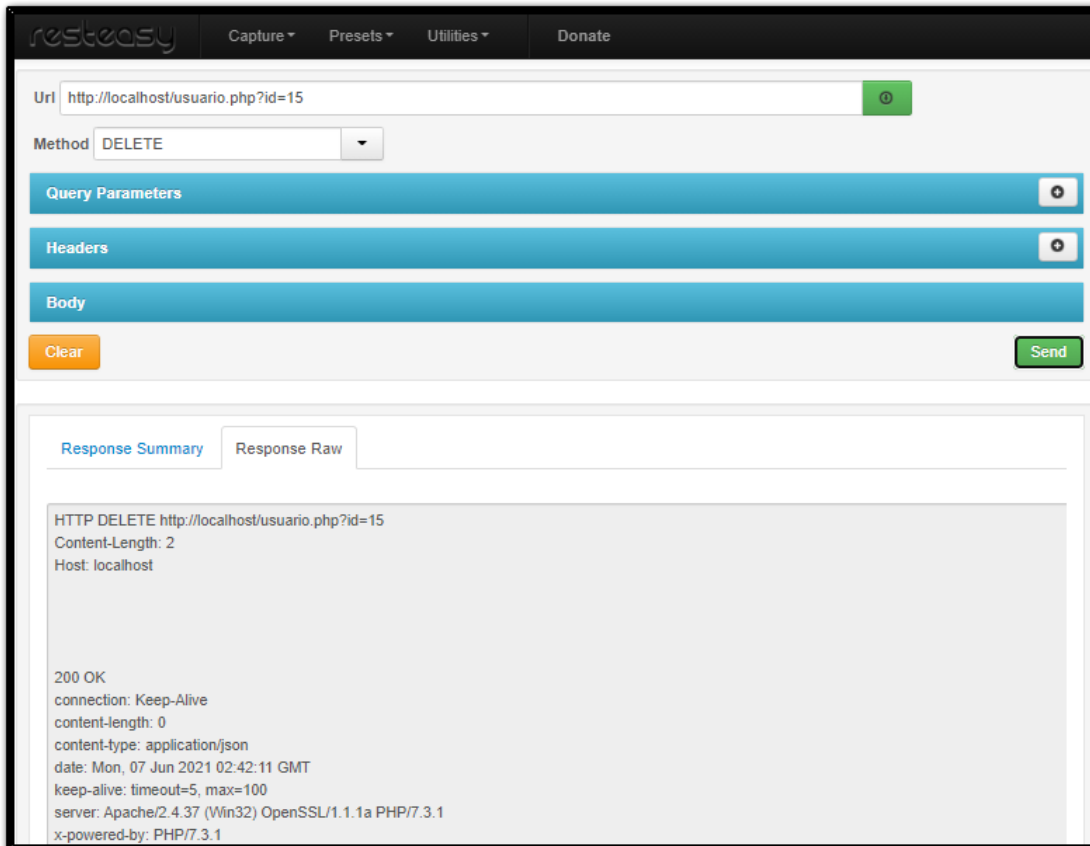
Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios

Estructura de la Base de datos Navegar Datos Editar Pragmas Ejecutar SQL

Tabla: EPS

	ideps	nombre	estadoeps
	Filtro	Filtro	Filtro
1	1	SALUDCOOP	4
2	2	CAFESALUD	1
3	3	NUEVA EPS	1
4	4	ISS	5
5	5	CAPRECOM	4
6	6	COOMEVA	1
7	7	COMPENSAR	1
8	8	COLMEDICA	1
9	9	HUMANA VIVIR	2
10	10	FAMISANAR	1
11	11	CRUZ BLANCA	3
12	12	SOLSALUD	1
13	13	SALUD TOTAL	5
14	14	SANITAS	3
15	15	La peligrosa	1

```
case 'DELETE':  
    $_P=json_decode(file_get_contents("php://input"),true);  
    $sql="DELETE FROM EPS WHERE IDEPS=".$_P["id"];  
    $cn->Ejecutar($sql);  
    break;
```



DB Browser for SQLite - C:\xampp\htdocs\SALUD\SALUD.db

Archivo Editar Ver Ayuda

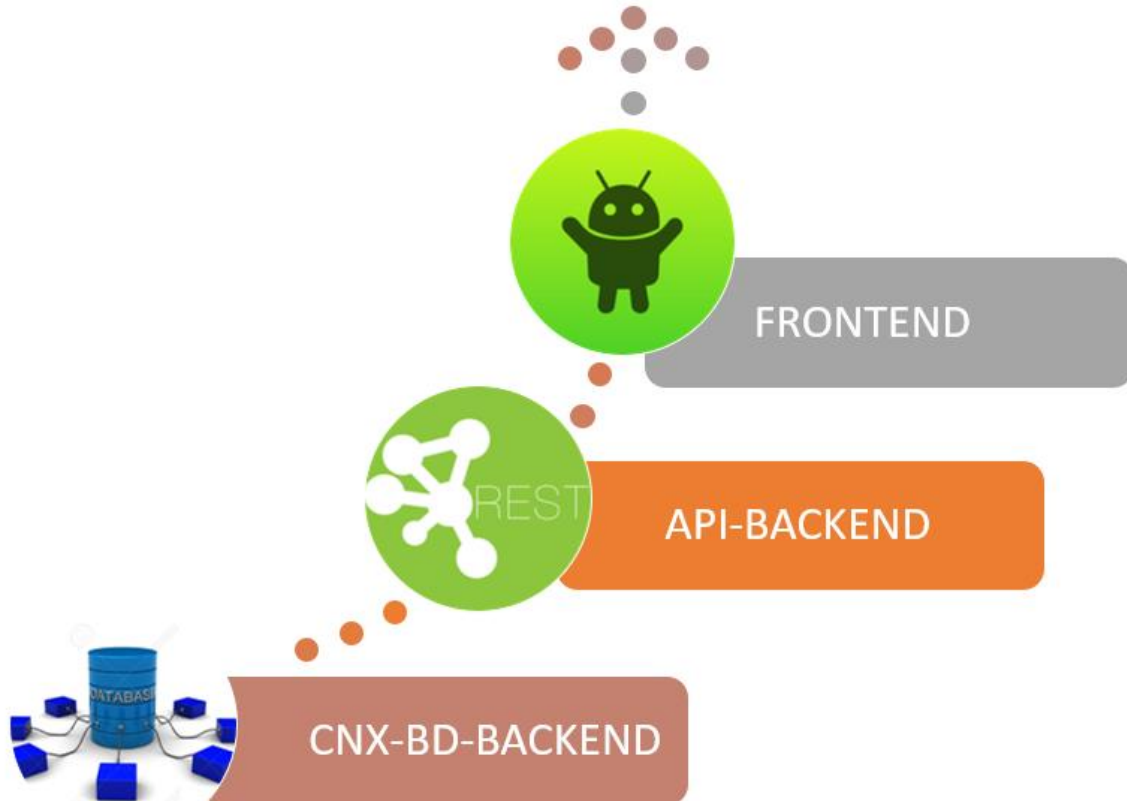
Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios

Estructura de la Base de datos Navegar Datos Editar Pragmas Ejecutar SQL

Tabla: EPS

	ideps	nombre	estadoeps
	Filtro	Filtro	Filtro
1	1	SALUDCOOP	4
2	2	CAFESALUD	1
3	3	NUEVA EPS	1
4	4	ISS	5
5	5	CAPRECOM	4
6	6	COOMEVA	1
7	7	COMPENSAR	1
8	8	COLMEDICA	1
9	9	HUMANA VIVIR	2
10	10	FAMISANAR	1
11	11	CRUZ BLANCA	3
12	12	SOLSALUD	1
13	13	SALUD TOTAL	5
14	14	SANTAS	3

## 2. UTILIZANDO LA API REST DESDE EL FRONTEND PHP.



PHP soporta [libcurl](#), una biblioteca creada por Daniel Stenberg que permite conectarse y comunicarse con diferentes tipos de servidores y diferentes tipos de protocolos. Actualmente, libcurl admite los protocolos http, https, ftp, gopher, telnet, dict, file y ldap. libcurl también admite certificados HTTPS, HTTP, POST, HTTP PUT, subidas mediante FTP (también se puede hacer con la extensión FTP de PHP), subidas basadas en formularios HTTP, proxies, cookies, y autenticación usuario+contraseña. [Leer más....](#)

Se Desarrollan los niveles del archivo “MIAPI.php”, que será la API-REST:

Se declara mediante el header que se trabaja en la API bajo el formato JSON, se incluye la clase de conectividad a la base de datos y se instancia el objeto \$Cnx con la clase SQLitePDO y se inicializa la variable \$sql que se usa para escribir la sentencia sql.

Página **21** de **61**

```
<?php
header("Content-Type: application/json");
include "cnx.php";
$Cnx=new SqlitePDO();
    $sql="";
```

El metodo HTTP que se usa con la API, se puede identificar con el comando `$_SERVER["REQUEST_METHOD"]` que puede ser: GET, POST, PUT o DELETE. Esta será evaluada mediante un comando switch.

Para el caso GET, si se envia el {id} mediante los parametros de la URL, validado mediante el comando “isset”, que llega mediante el protocolo GET asi: `$_GET["id"]`, se dispone a realizar la consulta del registro que se identifica mediante el {id} recibido. Se ejecuta (`$Cnx->Ejecutar`), se carga el Json (`$Cnx->CargarJSON`) y se muestra mediante un echo.

```
switch($_SERVER["REQUEST_METHOD"]){
    case 'GET':
        if(isset($_GET["id"]))
            $sql="SELECT * FROM EPS WHERE IDEPS=".$_GET["id"];
        else
            $sql="SELECT * FROM EPS";
        $Cnx->EjecutarJSON($sql);
        echo $Cnx->CargarJSON();
        break;
```

Para POST, donde tiene como objetivo insertar un nuevo registro y tal como se dijo en la instrucción a las API, esta conformado por un head y un body, para acceder a lo enviado en el body se utiliza “file\_get\_contents”, para esto se debe colocar en su parametro “php://input” que indica que son los datos enviados desde un programa PHP, despues de obtenerlo se convierte en formato array mediante la instrucción “json\_decode” y se almacena en el arreglo asociativo “`$_POST`”, se construye la sentencia sql-DML tipo insert con los datos de este arreglo asociativo. Se ejecuta la sentencia sql.

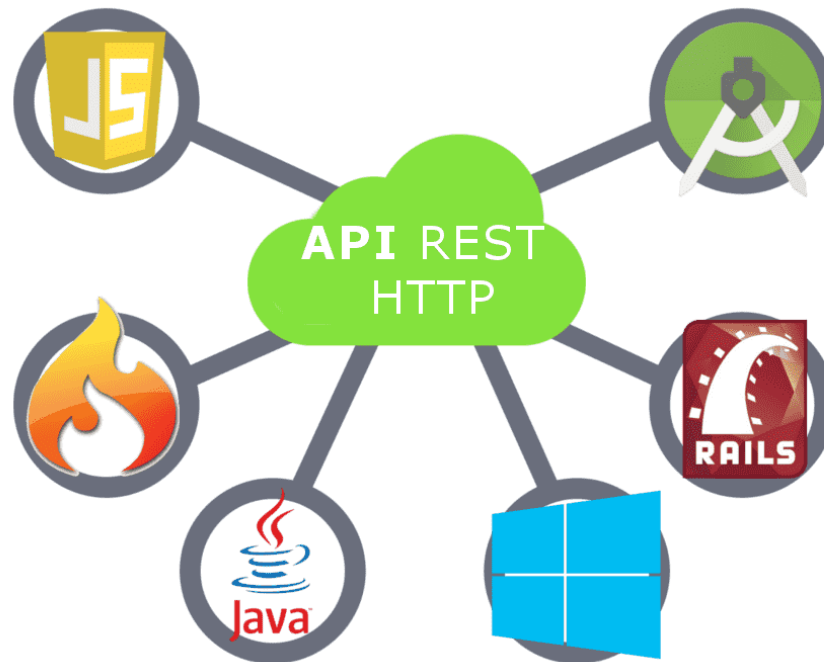
```
case 'POST':  
    $_POST=json_decode( file_get_contents("php://input"),true);  
    $sql="insert into eps(nombre,estadoeps)  
    values('".$_POST["nombre"]."','".$_POST["estadoeps"]."");  
    $Cnx->Ejecutar($sql);  
    break;
```

Para PUT, permite actualizar un registro, se utiliza “file\_get\_contents”, para esto debemos colocar en su parametro “php://input” que indica que son los datos enviados desde un programa PHP, despues de obtenerlo se convierte en formato array mediante la instrucción “json\_decode” y se almacena en el arreglo asociativo “\$\_POST”, se construye una sentencia sql-DML tipo update con los datos de este arreglo asociativo. Se ejecuta la sentencia sql.

```
case 'PUT':  
    $_POST=json_decode( file_get_contents("php://input"),true);  
    $sql="update eps set nombre='".$_POST["nombre"]."',  
    estadoeps='".$_POST["estadoeps"]." where ideps='".$_POST["id"]  
    $Cnx->Ejecutar($sql);  
    break;
```

Para DELETE, permite borrar un registro, se construye una sentencia sql-DML tipo delete con el {id} pasado por el protocolo GET en \$\_GET["id"]. Se ejecuta la sentencia sql.

```
case 'DELETE':  
    $sql="DELETE FROM EPS WHERE IDEPS='".$_GET["id"]";  
    $Cnx->Ejecutar($sql);  
    break;  
}
```



Creamos un archivo llamado “index.php” que será el FRONTEND que se conectará a la API.

Construimos una función que se encarga de comunicarse con la API REST:



La función “CnxCurl” es la encargada de conectarse con la API y utiliza tres parámetros donde el primer parámetro se envía el arreglo con los datos para que la API la procese, la URL donde se encuentra la API y el tercer y último parámetro es el protocolo HTTPS a utilizar (GET, POST, PUT y DELETE).

Su funcionamiento consiste:

Se convierte el arreglo pasado en el primer parámetro a formato json, se instancia la librería curl\_init() mediante la variable \$ch, la instrucción curl\_opt configura la dirección donde se encuentra la API, el



header de comunicación con la API, el método utilizado, el json enviado desde el body hacia la API y si se va transferir datos. Se ejecuta el curl y se cierra éste.

```
function CnxCurl(array $data,$url,$metodo){  
    $data_json = json_encode($data);  
    $ch = curl_init();  
  
    curl_setopt($ch, CURLOPT_URL, $url);  
  
    curl_setopt($ch, CURLOPT_HTTPHEADER,  
    array('Content-Type: application/json','Content-Length: ' . strlen($data_json)));  
  
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $metodo);  
  
    curl_setopt($ch, CURLOPT_POSTFIELDS,$data_json);  
  
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);  
  
    $response = curl_exec($ch);  
  
    curl_close($ch);  
  
    print_r ($response);  
}
```

Construimos los niveles del programa FrontEnd “index.php”:  
Se identifica el valor de la variable de tipo Nivel

```
<?php  
if(!isset($_GET["N"]))  
    $N=0;  
else  
    $N=$_GET["N"];  
?>
```

Se construye en JavaScript dos funciones que permite: la primera redirecciona a un nivel específico y la segunda función permite ir a un nivel específico con un valor; finalmente se crea un botón “Nuevo” que al dar clic en él, se ira al nivel 1 donde presenta un formulario vacío para el ingreso del nuevo registro.

```
<script type="text/javascript">
function Ir(donde){
    location.href="<?php echo $_SERVER['PHP_SELF']?>?N="+donde;
}
function Ir1(donde,que){
    location.href="<?php echo $_SERVER['PHP_SELF']?>?N="+donde+"&donde="+que;
}
</script>
<button onclick="Ir(1)">Nuevo</button>
```

En el nivel 0, se utiliza “file\_get\_contents”, que devuelve valores json desde la API y con la instrucción “json\_decode”, lo convierte en un arreglo asociativo; se construye una tabla HTML y con foreach se carga las filas de la tabla.

```
$url="http://localhost/API2242758/MIAPI.php";

if($N==0){
    $J=file_get_contents($url) ;
    $eps=json_decode($J,true);
    //print_r($eps);
    echo "<table border=1><th>Nombre</th><th>Estado</th><th colspan=2>Funciones</th><tr>";
    foreach($eps as $key =>$dato){
        echo "<tr><td>".$dato["nombre"]."</td><td>";
        . $dato["estadoeps"]."</td><td onclick='Ir1(3,\".$dato[\"ideps\"]";
        .")'>E</td><td onclick='Ir1(5,\".$dato[\"ideps\"]\".'>X</td></tr>";
    }
}
```

Todo lo hecho en el nivel cero tendrá como salida:

Nuevo	
Nombre	Estado
SALUDCOOP	4
CAFESALUD	1
NUEVA EPS	1
ISS	5
CAPRECOM	4
COOMEVA	1
COMPENSAR	1
COLMEDICA	1
HUMANA VIVIR	2
FAMISANAR	1
CRUZ BLANCA	3
SOLSALUD	1
SALUD TOTAL	5
SANITAS	3
XXXXXXXXXXXX	1

En el nivel 1 se construye el formulario para los datos del nuevo registro y al dar submit se dirige al nivel 2.

```
if($N==1){  
    echo "Nivel 1<br>";  
    echo "<form>  
    Nombre:<input type=text name=nombre><br>  
    Estado:<input type=number name=estadoeps><br>  
    <input type=hidden name=N value=2>  
    <input type=submit></form>";  
}
```

En el nivel 2 se construye el arreglo a enviar a la API, se determina la URL donde está la API y se envía estos datos con el método POST mediante la función "CnxCurl" y se devuelve hacia el Nivel 0.

```
if($N==2){  
    $dato=(  
        array("nombre"=>$_GET["nombre"],  
            "estadoeps"=>$_GET["estadoeps"]  
    ));  
    CnxCurl($dato, $url, 'POST') ;  
    echo "<script>Ir(0)</script>";  
}
```

El nivel 3 está encargado de crear un formulario que se llena con los datos del {id} pasado por la URL en \$\_GET["donde"], se utiliza "file\_get\_contents", que devuelve valores json desde la API y con la instrucción "json\_decode", lo convierte en un arreglo asociativo; este arreglo contiene elementos "stdClass", vistos en el taller de POO y que se deben cargar en cada elemento del formulario y que al dar submit nos llevará al nivel 4.

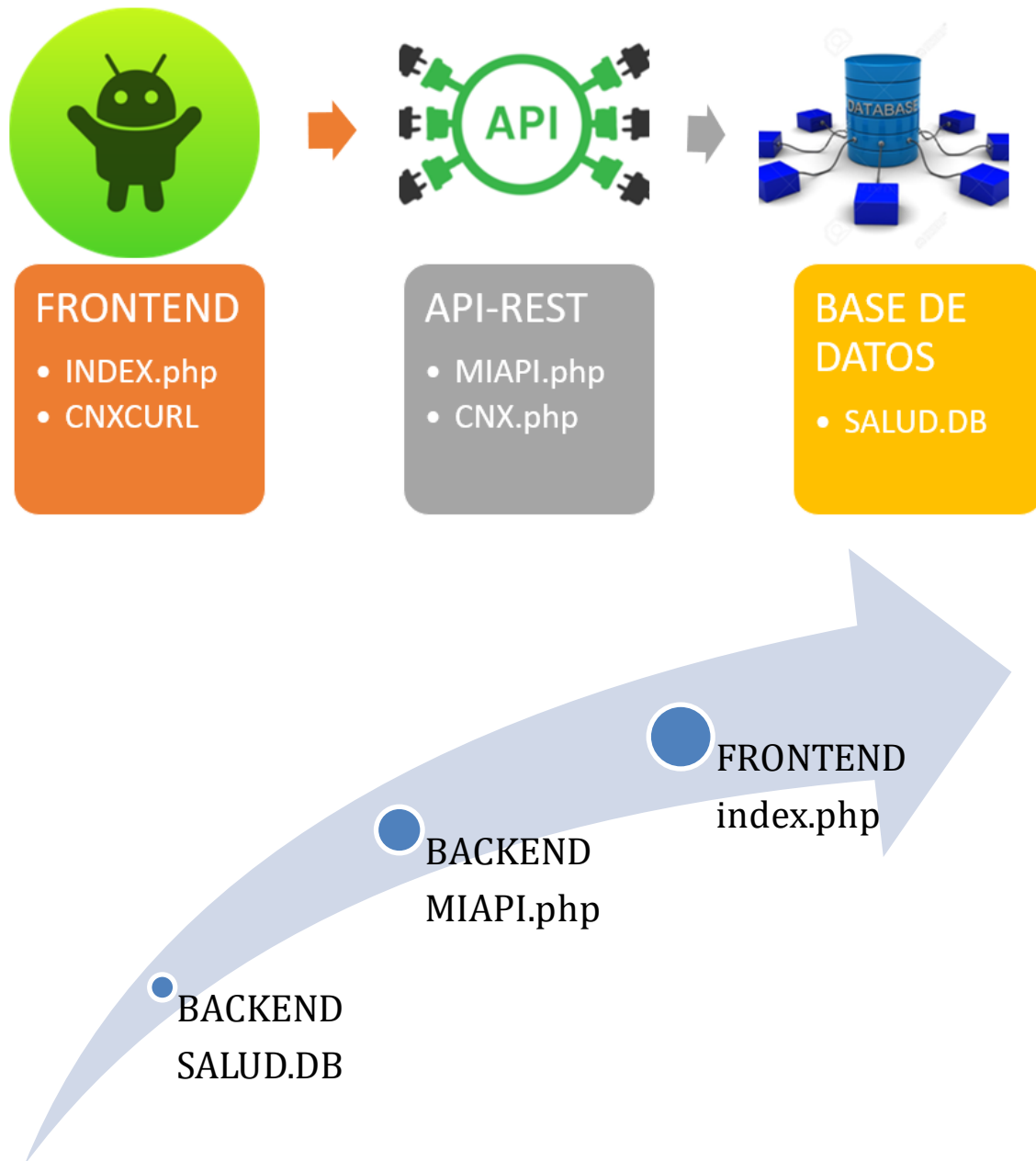
```
if($N==3){
    $J=file_get_contents($url."?id=".$GET[id]);
    $eps=json_decode($J,true);
    //print_r($eps);
    echo "Nivel 1<br>";
    echo "<form>
Nombre:<input type=text name=nombre value='".$eps[0][\"nombre\"]."'><br>
Estado:<input type=number name=estadoeps value='".$eps[0][\"estadoeps\"]."'><br>
<input type=hidden name=N value=4>
<input type=hidden name=id value='".$GET[id]."'>
<input type=submit></form>";
}
```

En el nivel 4 se construye el arreglo a enviar a la API, se determina la URL donde está la API, se envía estos datos con el método PUT mediante la función “CnxCurl” y se devuelve hacia el Nivel 0.

```
if($N==4){
    $dato=(
        array("id"=>$GET[id],
            "nombre"=>$GET[nombre],
            "estadoeps"=>$GET[estadoeps]
        ));
    CnxCurl($dato, $url, 'PUT') ;
    echo "<script>Ir(0)</script>";
}
```

En el nivel 5 se construye el arreglo a enviar a la API, se determina la URL donde está la API con el {id} enviado en la URL que se obtiene con \$GET[“donde”], y se envía estos datos con el método DELETE mediante la función “CnxCurl” y se devuelve hacia el Nivel 0.

```
if($N==5){
    $dato=(array("id"=>$GET[id]));
    CnxCurl($dato, $url, 'DELETE') ;
    echo "<script>Ir(0)</script>";
}
```



Nuevo

1

Nombre	Estado	Funcion	
SALUDCOOP	4	E	X
CAFESALUD	1	E	X
NUEVA EPS	1	E	X
ISS1	5	E	X
CAPRECOM	4	E	X
COOMEVA	1	E	X
COMPENSAR	1	E	X
COLMEDICA	1	E	X
HUMANA VIVIR	2	E	X
FAMISANAR	1	E	X
CRUZ BLANCA	3	E	X
SOLSALUD	1	E	X
SALUD TOTAL	5	E	X
SANITAS	3	E	X
SANTANDER	1	E	X
VALLE DEL CAUCAS	1	E	X
xxxxxxx	1	E	X



Nivel 1  
Nombre EPS:



Nuevo

Nombre	Estado	Funcion	
SALUDCOOP	4	E	X
CAFESALUD	1	E	X
NUEVA EPS	1	E	X
ISS1	5	E	X
CAPRECOM	4	E	X
COOMEVA	1	E	X
COMPENSAR	1	E	X
COLMEDICA	1	E	X
HUMANA VIVIR	2	E	X
FAMISANAR	1	E	X
CRUZ BLANCA	3	E	X
SOLSALUD	1	E	X
SALUD TOTAL	5	E	X
SANITAS	3	E	X
SANTANDER	1	E	X
VALLE DEL CAUCAS	1	E	X
xxxxxx	1	E	X
LA PELIGROSA	1	E	X





Nuevo Nivel 3

Nombre EPS: LA RUMBERA 1

Estado: 1 2 Enviar 3



Nuevo

Nombre	Estado	Funcion	
SALUDCOOP	4	E	X
CAFESALUD	1	E	X
NUEVA EPS	1	E	X
ISS1	5	E	X
CAPRECOM	4	E	X
COOMEVA	1	E	X
COMPENSAR	1	E	X
COLMEDICA	1	E	X
HUMANA VIVIR	2	E	X
FAMISANAR	1	E	X
CRUZ BLANCA	3	E	X
SOLSALUD	1	E	X
SALUD TOTAL	5	E	X
SANITAS	3	E	X
SANTANDER	1	E	X
VALLE DEL CAUCAS	1	E	X
xxxxxx	1	E	X
LA RUMBERA	1	E	X

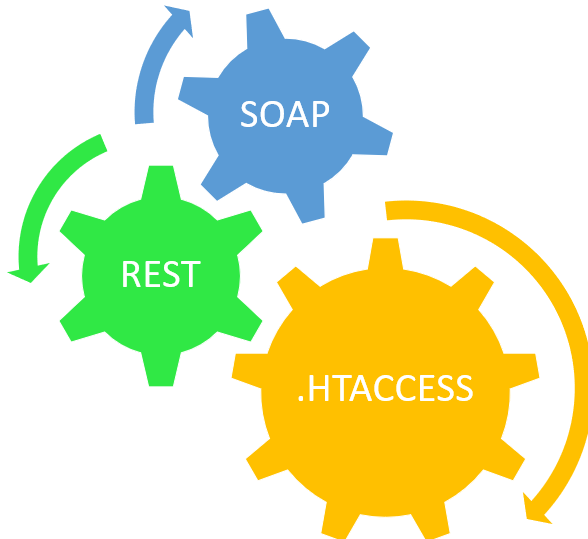


Nuevo

Nombre	Estado	Funcion	
SALUDCOOP	4	E	X
CAFESALUD	1	E	X
NUEVA EPS	1	E	X
ISS1	5	E	X
CAPRECOM	4	E	X
COOMEVA	1	E	X
COMPENSAR	1	E	X
COLMEDICA	1	E	X
HUMANA VIVIR	2	E	X
FAMISANAR	1	E	X
CRUZ BLANCA	3	E	X
SOLSALUD	1	E	X
SALUD TOTAL	5	E	X
SANITAS	3	E	X
SANTANDER	1	E	X
VALLE DEL CAUCAS	1	E	X
xxxxxx	1	E	X

<https://jsonlint.com/>

### 3. WEBSERVICES CON PHP



REST es una forma simple de organizar interacciones entre sistemas independientes. REST le 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.

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 métodos HTTP.

#### Proyecto para desarrollar paso a paso:

Descargar los archivos desde [GITHUB](https://github.com/fegasu/api) para la práctica o utilizando el comando: “`git clone https://github.com/fegasu/api`”, desde git bash o descargue [GITHUB-DESKTOP](#).

En htdocs, crear una carpeta llamada “SALUD”, y ubicar los archivos “cnx.php” y “SALUD.db”, luego crear un archivo llamado “.htaccess” con el siguiente contenido:

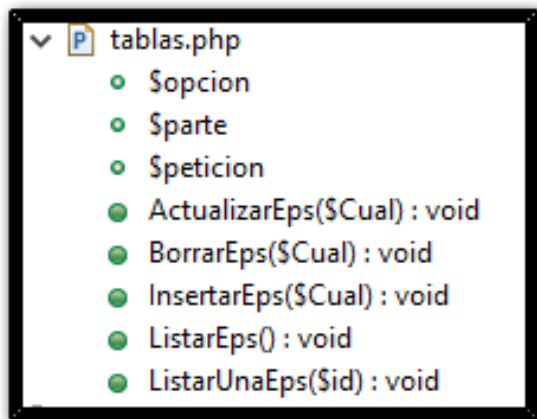
```
1 Options +FollowSymLinks -Multiviews
2 RewriteEngine on
3 RewriteRule ^([^\s]+)/([^\s]+)/([^\s]+) tablas.php?peticion=$1&opcion=$2&parte=$3 [L,QSA]
4
5
```

- 1- El archivo que contiene la API llamada “tablas.php”
- 2- Primer parámetro llamado “petición”
- 3- Segundo parámetro llamado “opción”
- 4- Tercer parámetro llamado “parte”



### 5- Reglas de construcción de la API.

PETICIÓN	OPCIÓN	PARTE	FUNCIÓN ASOCIADA	DESCRIPCIÓN
eps	0	0	ListarEps	Lista todos los registros
eps	0	{id}	ListarUnaEps(PARTE)	Lista un registro de acuerdo con el {id} pasado en PARTE.
eps	1	{v1;v2;...vn}	InsertarEps(PARTE)	Inserta un registro de acuerdo con los valores pasados en PARTE separados por punto y coma
eps	2	{id};{v1;v2;...vn}	ActualizarEps(PARTE)	Actualiza un registro de acuerdo con los valores pasados en PARTE separados por punto y coma y al {id} pasado
eps	3	{id}	BorrarEps(PARTE)	Borra un registro de acuerdo con el id pasado



Construir un archivo llamado “tablas.php”, que será la API a desarrollar, con el siguiente contenido:

```
1  <?php
2  $peticion=$_GET["peticion"];
3  $opcion=$_GET["opcion"];
4  $parte=$_GET["parte"];
5  echo "peticion=".$peticion."<br>";
6  echo "opcion=".$opcion."<br>";
7  echo "PARTE=".$parte."<br>";
```

Ejecute en el navegador la URL: “<http://localhost/SALUD/eps/0/0>” y la salida será:

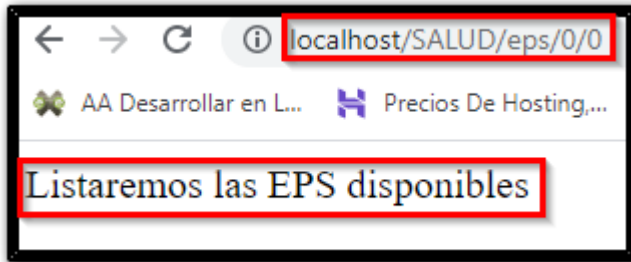
```
peticion=eps
opcion=0
PARTE=0
```

Seguidamente se debe colocar en comentarios las filas 5,6 y 7.  
Se debe incluir el archivo “cnx.php” en el archivo “tablas.php”.

Crear un función llamada “ListarEps”

```
9  function ListarEps(){
10     $cn=new SqlitePDO("SALUD.db");
11     echo "Listaremos las EPS disponibles";
12 }
13
14 if($peticion=="eps" && $opcion==0){
15     ListarEps();
16 }
```

Al ejecutar la salida será:



Ajustar la función ListarEps para que la salida sea un JSON:

```
function ListarEps(){  
    $cn=new SQLitePDO("SALUD.db");  
    $sql="select *from eps";  
    $cn->EjecutarJSON($sql);  
    echo $cn->CargarJSON();  
}
```

```
if($peticion=="eps" && $opcion==0 && $parte==0){  
    ListarEps();  
}
```

```
[{"ideps": "1", "nombre": "SALUDCOOP", "estadoeps": "4"}, {"ideps": "2", "nombre": "CAFESALUD", "estadoeps": "1"}, {"ideps": "3", "nombre": "NUEVA EPS", "estadoeps": "1"}, {"ideps": "4", "nombre": "ISS", "estadoeps": "5"}, {"ideps": "5", "nombre": "CAPRECOM", "estadoeps": "4"}, {"ideps": "6", "nombre": "COOMEVA", "estadoeps": "1"}, {"ideps": "7", "nombre": "COMPENSAR", "estadoeps": "1"}, {"ideps": "8", "nombre": "COLMEDICA", "estadoeps": "1"}, {"ideps": "9", "nombre": "HUMANA VIVIR", "estadoeps": "2"}, {"ideps": "10", "nombre": "FAMISANAR", "estadoeps": "1"}, {"ideps": "11", "nombre": "CRUZ BLANCA", "estadoeps": "3"}, {"ideps": "12", "nombre": "SOLSALUD", "estadoeps": "1"}, {"ideps": "13", "nombre": "SALUD TOTAL", "estadoeps": "5"}, {"ideps": "14", "nombre": "SANITAS", "estadoeps": "3"}]
```

Crear la función “ListarUnaEps”

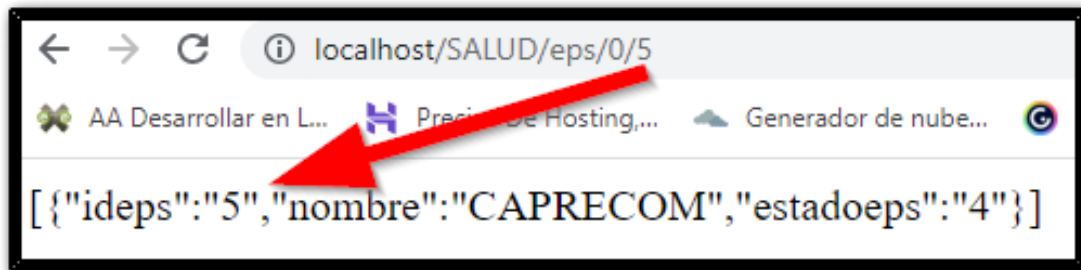
```
function ListarUnaEps($id){  
    $cn=new SqlitePDO("SALUD.db");  
    $sql="select *from eps where ideps=".$id;  
    $cn->EjecutarJSON($sql);  
    echo $cn->CargarJSON();  
}
```

Y realizar la llamada para buscar un id de una eps:

```
if($peticion=="eps" && $opcion==0 && $parte>0){  
    ListarUnaEps($parte);  
}
```

La salida es:





```
function InsertarEps($Cual){  
    $cn=new SqlitePDO("SALUD.db");  
    $a=explode(";", $Cual);  
    $sql="INSERT INTO EPS(NOMBRE,ESTADOEPS) VALUES('".$a[0]."', '".$a[1]."')";  
    $cn->Ejecutar($sql);  
}
```

```
if($peticion=="eps" && $opcion==1){  
    InsertarEps($parte);  
}
```



 localhost/SALUD/eps/1/La+Z;1

DB Browser for SQLite - C:\xampp\htdocs\SALUD\SALUD.db

Archivo Editar Ver Ayuda

Nueva base de datos Abrir base de datos Guardar

Estructura de la Base de datos Navegar Datos Editar Pragma

Tabla: EPS

	ideps	nombre	estadoeps
	Filtro	Filtro	Filtro
1	1	SALUDCOOP	4
2	2	CAFESALUD	1
3	3	NUEVA EPS	1
4	4	ISS	5
5	5	CAPRECOM	4
6	6	COOMEVA	1
7	7	COMPENSAR	1
8	8	COLMEDICA	1
9	9	HUMANA VIVIR	2
10	10	FAMISANAR	1
11	11	CRUZ BLANCA	3
12	12	SOLSALUD	1
13	13	SALUD TOTAL	5
14	14	SANITAS	3
15	16	La X	3
16	17	La Z	1

```
function ActualizarEps($Cual){  
    $cn=new SqlitePDO("SALUD.db");  
    $a=explode(";", $Cual);  
    $sql="UPDATE EPS SET NOMBRE='". $a[1]. "', ESTADOEPS='". $a[2]. "' WHERE IDEPS='". $a[0]. "'";  
    $cn->Ejecutar($sql);  
}
```

```
if($peticion=="eps" && $opcion==2){  
    ActualizarEps($parte);  
}
```



← → ↻ ⓘ localhost/SALUD/eps/2/5;caprecom1;5

DB Browser for SQLite - C:\xampp\htdocs\SALUD\SALUD.db

Archivo Editar Ver Ayuda

Nueva base de datos Abrir base de datos Guardar

Estructura de la Base de datos Navegar Datos Editar Pragma

Tabla: EPS

	ideps	nombre	estadoeps
	Filtro	Filtro	Filtro
1	1	SALUDCOOP	4
2	2	CAFESALUD	1
3	3	NUEVA EPS	1
4	4	ISS	5
5	5	caprecom1	5
6	6	COOMEVA	1
7	7	COMPENSAR	1
8	8	COLMEDICA	1
9	9	HUMANA VIVIR	2
10	10	FAMISANAR	1
11	11	CRUZ BLANCA	3
12	12	SOLSALUD	1
13	13	SALUD TOTAL	5
14	14	SANITAS	3
15	16	La X	3
16	17	La Z	1

```
function BorrarEps($Cual){  
    $cn=new SqlitePDO("SALUD.db");  
    $sql="DELETE FROM EPS WHERE IDEPS=".$Cual;  
    $cn->Ejecutar($sql);  
}
```

```
if($peticion=="eps" && $opcion==3){  
    BorrarEps($parte);  
}
```

← → ↻ ⓘ localhost/SALUD/eps/3/17

#### 4. FRONTEND EN PHP PARA CONSUMIR LA API DESDE EL BACKEND PHP

Se determina la variable Nivel que se utiliza y se construye en JavaScript dos funciones que permite: la primera redirecciona a un nivel específico y la segunda función permite ir a un nivel específico con un valor; finalmente se crea un botón “Nuevo” que al dar clic en él, se ira al nivel 1 donde presenta un formulario vacío para el ingreso del nuevo registro.

```
<?php
if(!isset($_GET["N"]))
$N=0;
else
$N=$_GET["N"];
?>

<script type="text/javascript">
function Ir(donde){
    location.href="<?php echo $_SERVER['PHP_SELF']?>?N="+donde;
}
function Ir1(donde,que){
    location.href="<?php echo $_SERVER['PHP_SELF']?>?N="+donde+"&donde="+que;
}
</script>
<button onclick="Ir(1)">Nuevo</button>
```

Crear la función “CnxAPI” en PHP con cuatro parámetros:

1. Carpeta donde se encuentra la API
2. Petición o tabla para gestionar
3. Opción que va a utilizar (ver tabla reglas)
4. Parte parámetros extra

Esta función permite cambiar los espacios por el símbolo más (+) para ser enviada por la URL como parámetros

```
function CnxAPI($purl,$peticion,$opcion,$parte){  
    $eps="";  
    $vparte=str_replace(' ','+', $parte);  
    $vurl="http://".$_SERVER["SERVER_NAME"]."/".  
    $purl."/". $peticion."/". $opcion."/". $vparte;  
    $JSON = file_get_contents($vurl,true);  
    if($opcion>0){  
        $eps=json_decode($JSON);  
    }else  
        $eps=$JSON;  
    return $eps;  
}
```

También crear la función “CSV2ARRAY” en PHP, que convierte una cadena de caracteres CSV a arreglo, utilizado solamente cuando la opción es cero.

```
function CSV2ARRAY($cadena){  
    $a=array();  
    $z=explode("|",$cadena);  
    for($i=0;$i<count($z)-1;$i++){  
        $w=explode(",",$z[$i]);  
        for($j=0;$j<count($w);$j++){  
            $a[$i][$j]=$w[$j];  
        }  
    }  
    return $a;  
}
```

Este nivel 0, construye la tabla principal con todas las EPS

```
if($N==0){  
    $a=CnxAPI("SALUD","eps",0,0);  
    $eps=CSV2ARRAY($a);  
    echo "<table border=1><tr><th>Nombre</th><th>Estado</th><th colspan=2>Funcion</th></tr>";  
    foreach($eps as $key => $dato){  
        echo "<tr><td>",$dato[1]."</td><td>",$dato[2]."  
        </td><td onclick='Ir1(3,\".$dato[0].\" )>E</td><td onclick='Ir1(5,\".$dato[0].\"  
        )>X</td></tr>";  
    }  
}
```

Se crea un formulario vacío para guardar la información de la nueva EPS

```
if($N==1){  
echo "Nivel 1";  
echo "<form name=mio method=gett >Nombre EPS:  
<input type=text name=nombre><input type=submit>  
<input type=hidden name=estadoeps value=1>  
<input type=hidden name=N value=2></form>";  
}
```

Este nivel 2, permite ingresar una nueva EPS con la información enviada en el formulario del nivel anterior

```
if($N==2){  
echo "Nivel 2<br>";  
$a=CnxAPI("SALUD","eps",1,$_GET["nombre"].".".$_GET["estadoeps"]);  
echo "<script>Ir(0)</script>";  
}
```

En este nivel 3, se consulta a la base de datos la EPS a editar y se incrusta en un formulario que permite hacer el cambio y al dar submit se envía al nivel 4.

```
if($N==3){  
echo "Nivel 3<br>";  
$eps=CnxAPI("SALUD","eps",0,$_GET["donde"]);  
$eps=explode(",",$eps);  
  
echo "<form name=mio method=get >  
Nombre EPS:<input type=text name=nombre value='".$eps[1]."'><br>  
Estado:<input type=number name=estadoeps value='".$eps[2]."'>  
<input type=hidden name=N value=4>  
<input type=hidden name=id value='".$_GET["donde"].">  
<input type=submit></form>";  
}
```

El nivel 4 recibe la información enviada desde el formulario del nivel 3

```
if($N==4){  
echo "Nivel 4<br>";  
$eps=CnxAPI("SALUD","eps",2,$_GET["id"]."."."  
.str_replace(' ','+',$_GET["nombre"]).".".$_GET["estadoeps"]);  
echo "<script>Ir(0)</script>";  
}
```





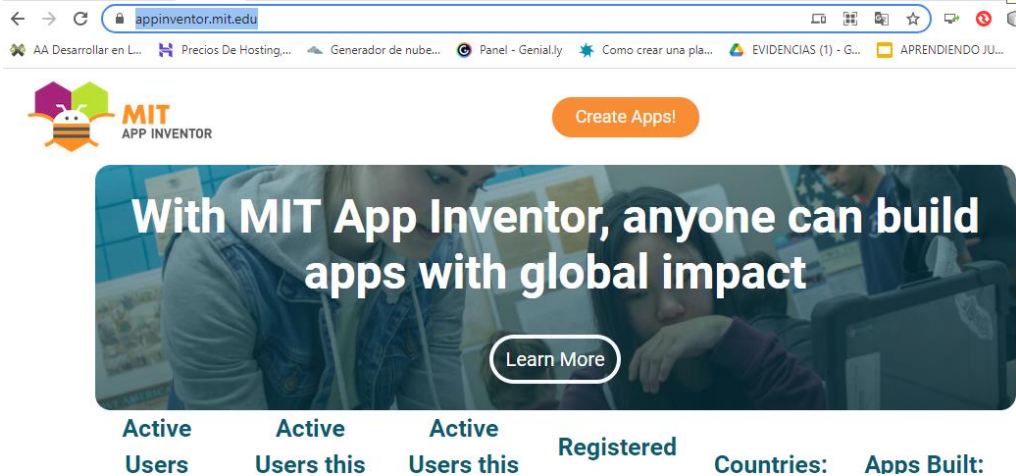
En el nivel 5, permite borrar una EPS

```
if($N==5){  
  echo "Nivel 5<br>";  
  $eps=CnxAPI("SALUD","eps",3,$_GET["donde"]);  
  echo "<script>Ir(0);</script>";  
}
```

## 5. FRONTEND APPINVENTOR (MÓVIL) CONSUMIENDO API REST PHP

Para realizar este laboratorio se debe descargar el proyecto [appinventor](#)

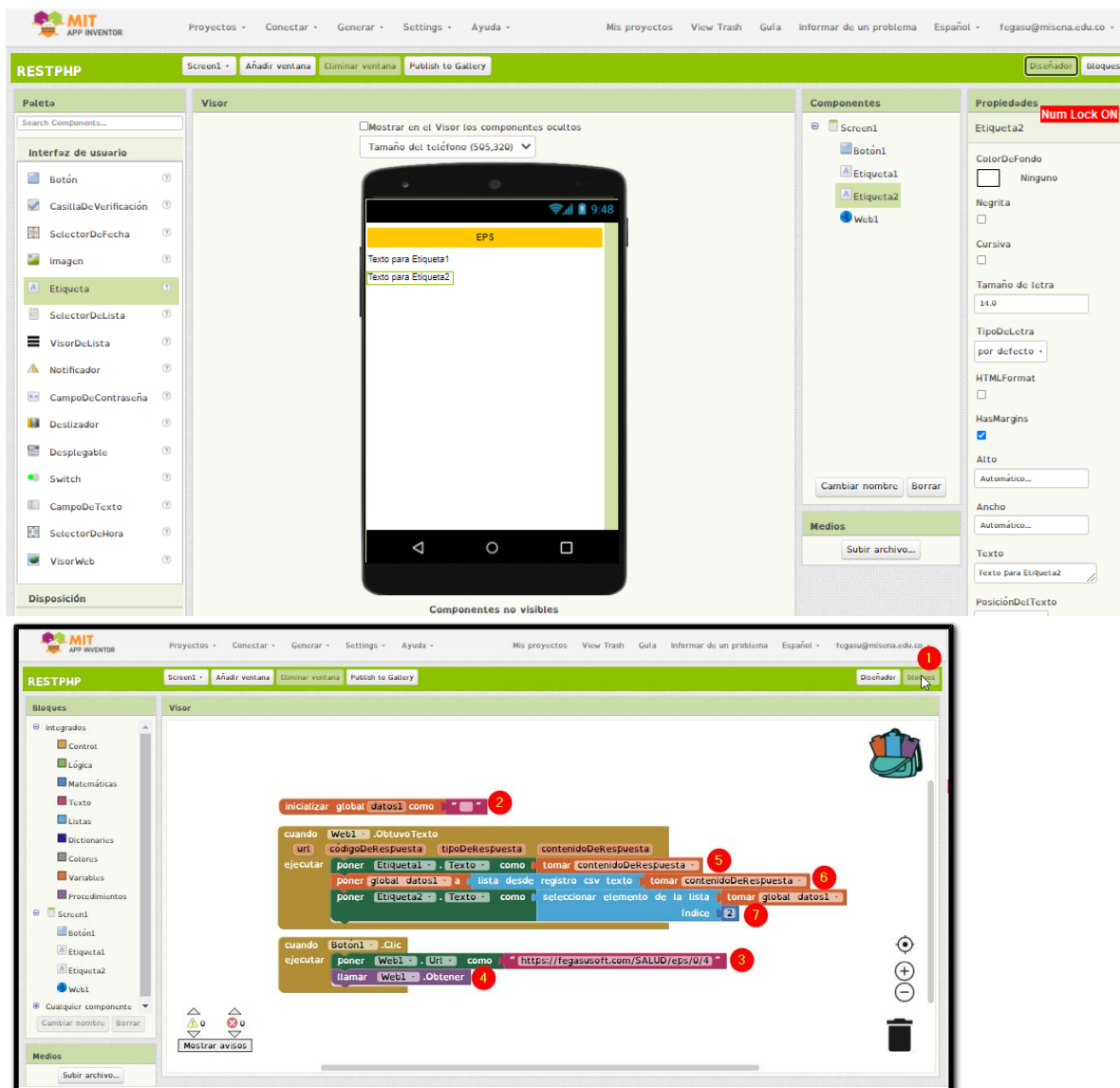
Ingresa [appinventor](#) y carga el proyecto descargado anteriormente así:







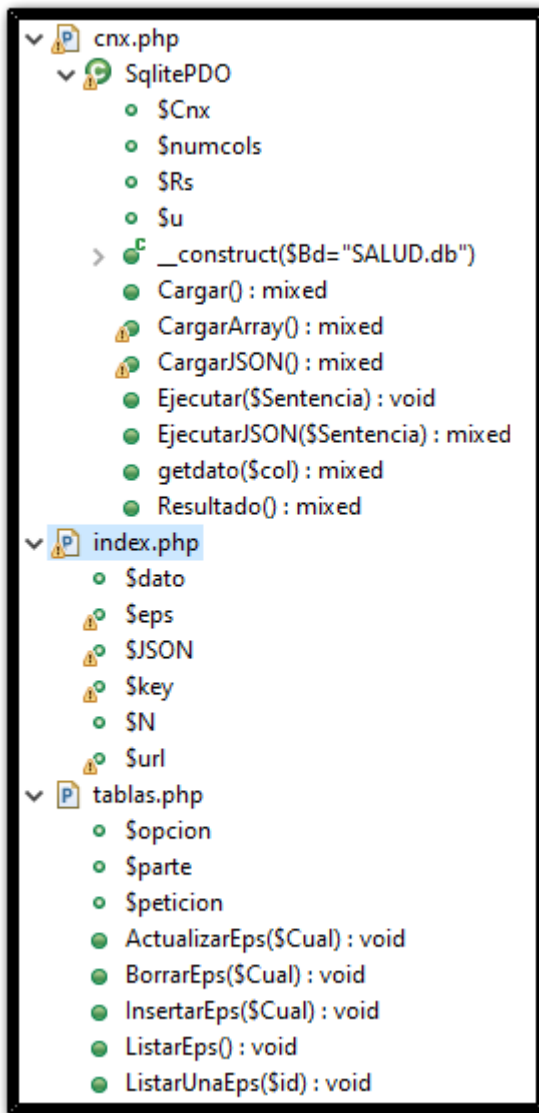
Servicio Nacional de Aprendizaje  
Formato Taller  
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.



Página 52 de 61

Línea de atención al ciudadano: 018000 910270  
Línea de atención al empresario: 018000 910682

  
@SENAcomunica [www.sena.edu.co](http://www.sena.edu.co)



Antes de realizar la practica se hacen unos cambios en “cnx.php”, se adiciona un metodo llamado resultado que entrega una cadena separada por comas:

```
<?php
class SqlitePDO{
var $Cnx;
var $Rs;
var $u;
public $numcols;

function Ejecutar($Sentencia){

    try{
        $this->Rs = $this->Cnx->prepare($Sentencia.';') or die(SQLITE_ERROR.' '.$Sentencia);
        $this->Rs->execute();
        $this->numcols=$this->Rs->columnCount();

    }catch (Exception $e){ die($e);
    }
}
```

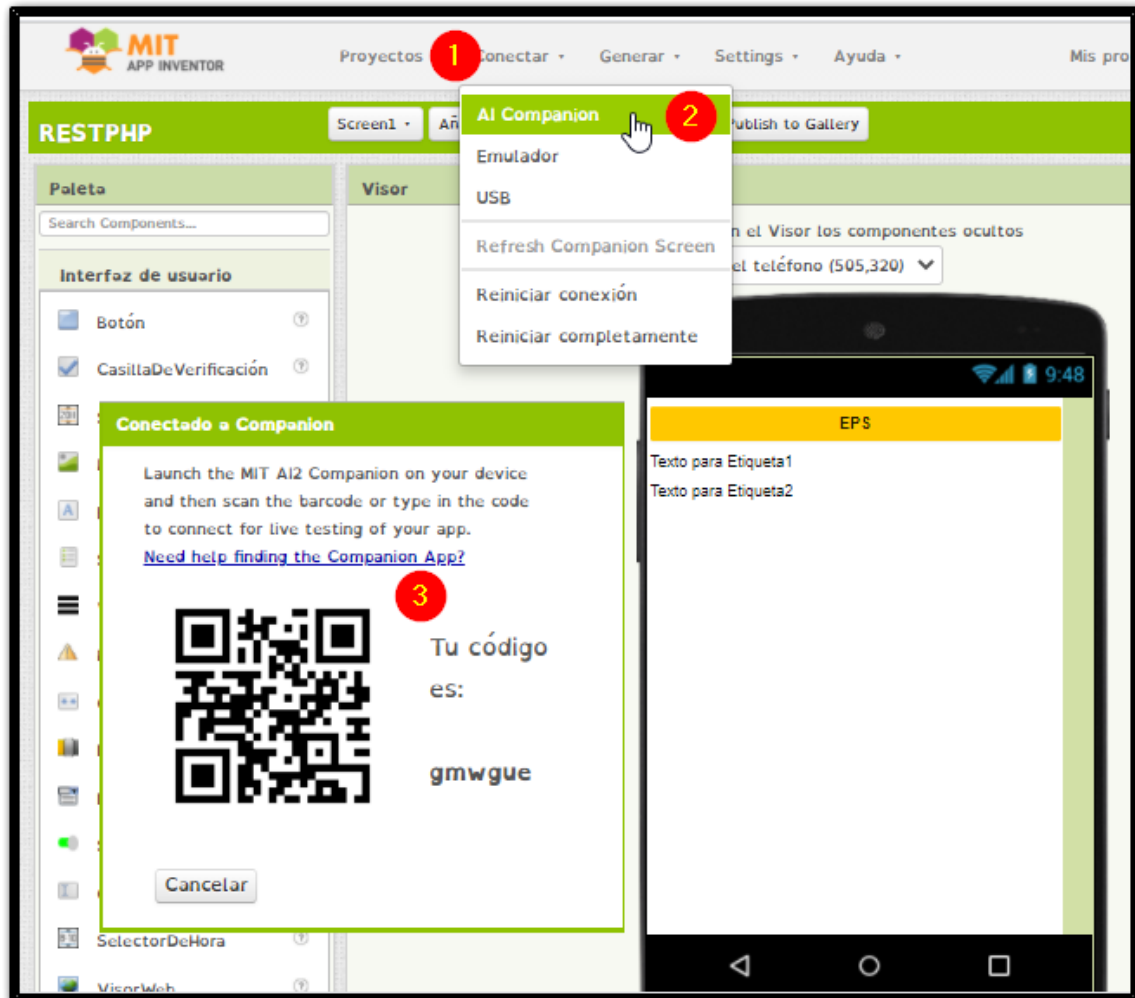
Se creo este nuevo metodo:

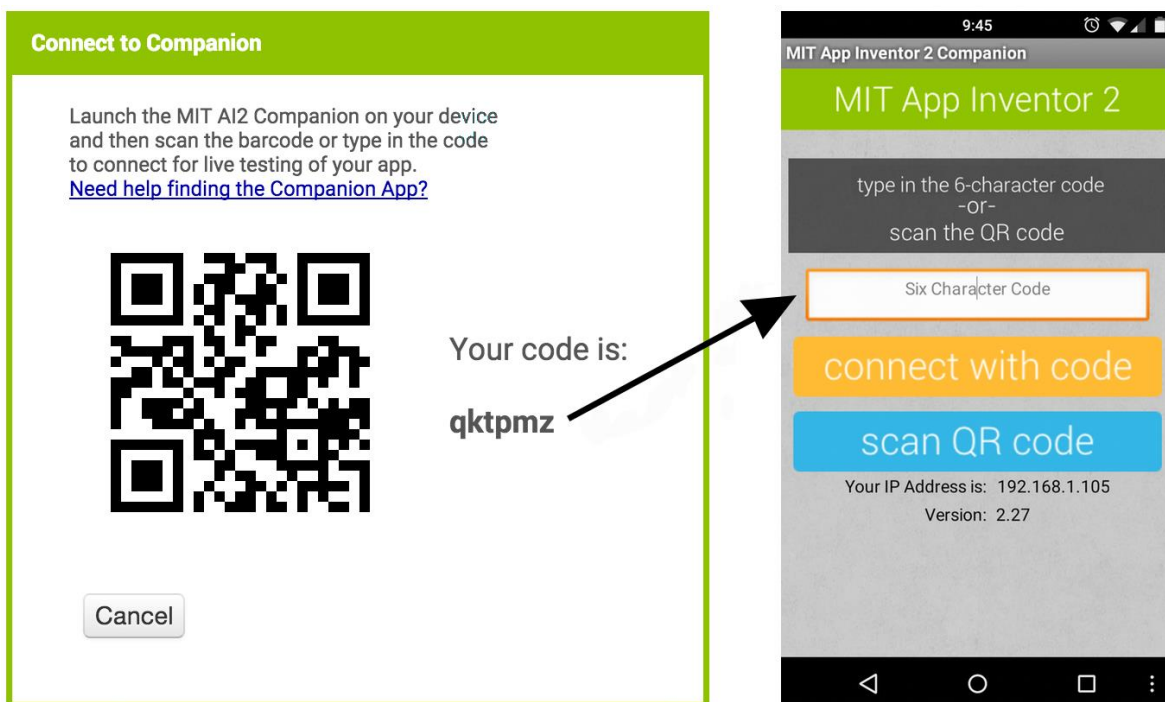
```
function Resultado(){
    $i=0;$a="";
    for($i=0; $i<$this->numcols;$i++){
        $a.=$this->getdato($i);
        if($i<$this->numcols-1)
            $a.=",";
    }
    return $a;
}
```

Se hizo cambios en el archivo "tablas.php":

```
function ListarUnaEps($id){
    $cn=new SqlitePDO("SALUD.db");
    $sql="select * from eps where ideps=".$id;
    $cn->Ejecutar($sql);
    //echo $cn->numcols."<br>";
    $cn->Cargar();
    echo $cn->Resultado();
}
```

Que sera utilizado desde AppInventor







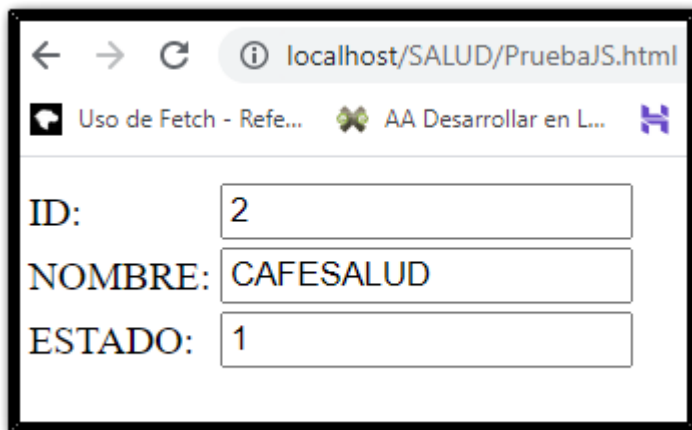
## 6. FRONTEND JAVASCRIPT CONSUMIENDO API REST PHP

```
<form>
<table>
  <td>ID:</td><td><input type=text id=a1 value=""></td><tr>
  <td>NOMBRE:</td><td><input type=text id=a2 value=""></td><tr>
  <td>ESTADO:</td><td><input type=text id=a3 value=""></td><tr>
</table>
</form>
<script>
var url = 'http://localhost/SALUD/eps/0/2';

async function file_get_contents(uri, callback) {
  let res = await fetch(uri),
      ret = await res.text();
  return callback ? callback(ret) : ret;
}

function Mio(x) {
  z=x.split(",");
  document.getElementById('a1').value=z[0];
  document.getElementById('a2').value=z[1];
  document.getElementById('a3').value=z[2];
}
file_get_contents(url, Mio);

</script>
```



localhost/SALUD/PruebaJS.html

Uso de Fetch - Refe... AA Desarrollar en L...

ID:	<input type="text" value="2"/>
NOMBRE:	<input type="text" value="CAFESALUD"/>
ESTADO:	<input type="text" value="1"/>

Para esto se debe adicionar el siguiente método en “cnx.php”:

```
function CargarCSV($sentencia){  
    $this->Ejecutar($sentencia);  
    $a="";  
    $j=0;  
    while($si=$this->Cargar()){  
        for($i=0; $i<$this->numcols;$i++){  
            $a.=$this->getdato($i);  
            if($i<$this->numcols-1)  
                $a.=",";  
        }  
        if(!$si)  
            $a.="";  
        else  
            $a.="|";  
    }  
    return $a;  
}
```

Modificar la API “tablas.php” en el método “ListarEps” así:

```
function ListarEps(){  
    $cn=new SQLitePDO("SALUD.db");  
    $sql="select *from eps";  
    //$cn->EjecutarJSON($sql);  
    echo $cn->CargarCSV($sql);  
}
```

Construir el archivo "Listar.html"

```
<script>
async function file_get_contents(uri, callback) {
  let res = await fetch(uri),
      ret = await res.text();
  return callback ? callback(ret) : ret;
}

function MiEstadoEps(x) {
  z=x.split("|");
  document.write("<table border=1><th>IdEps</th><th>Nombre Eps</th><th>Estado</th>");
  for(i=0;i<z.length-1;i++){
    w=z[i].split(",");
    document.write("<tr><td>"+w[0]+"</td><td>"+w[1]+"</td><td>"+w[2]+"</td></tr>");
  }
}

url = 'http://localhost/SALUD/eps/0/0';
file_get_contents(url, MiEstadoEps);

</script>
```

IdEps	Nombre Eps	Estado
1	SALUDCOOP	4
2	CAFESALUD	1
3	NUEVA EPS	1
4	ISS	5
5	CAPRECOM	4
6	COOMEVA	1
7	COMPENSAR	1
8	COLMEDICA	1
9	HUMANA VIVIR	2
10	FAMISANAR	1
11	CRUZ BLANCA	3
12	SOLSALUD	1
13	SALUD TOTAL	5
14	SANITAS	3
21	SOLOSALUD	1

Página 59 de 61

## RETO – EVIDENCIA



El reto consiste en convertir la **API-REST** y la **API-THACCESS**, que está conectada a **BACKEND-PHP-SQLITE**, ahora conectarla con **MYSQL** y utilizar el procedimiento **CRUD** llamado “**PREPS**” que se construyó en el anterior taller.

Se debe modificar los archivos “**cnx.php**”, “**MIAPI.php**”, “**tablas.php**” y el **FRONTEND-PHP** “**index.php**” para

conectar con el **BACKEND-PHP-MYSQL**.

Elabore un informe llamado “**InformeTallerReto.docx**”, donde se detalle los cambios necesarios que se realizaron para desarrollar el reto, es importante que inserte pantallazos donde se evidencia la salida al ejecutar los cambios, adjunte en la evidencia llamada “**MiRetoAPI.zip**”, los archivos utilizados en el desarrollo del reto; el archivo comprimido debe ser ZIP, no se aceptan otros formatos y no olvide guardar una copia en portafolio de evidencias del aprendiz.



Recuerde que esta evidencia debe ser presentada mínimo de cuatro(4) aprendices y debe ser expuesta por uno de ellos que será elegido por el instructor.

Responda el [cuestionario](#) sobre API para saber cuanto ha aprendido sobre el tema.





### RECURSOS ADICIONALES DISPONIBLES:

[https://jwt.io/#debugger-io?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImRhdG8iOiJkb3NlIn0.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.7O7hx3G4TxpUGAj4J9R4pS6z\\_TuhoEVemslqLdeByk](https://jwt.io/#debugger-io?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImRhdG8iOiJkb3NlIn0.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.7O7hx3G4TxpUGAj4J9R4pS6z_TuhoEVemslqLdeByk)

<https://living-sun.com/es/php/643701-how-to-decode-jwt-using-jwt-auth-in-laravel-php-laravel-jwt-jwt-auth.html>

