TEMA 7. PROGRAMACIÓN DE SERVICIOS WEB.

Objetivos

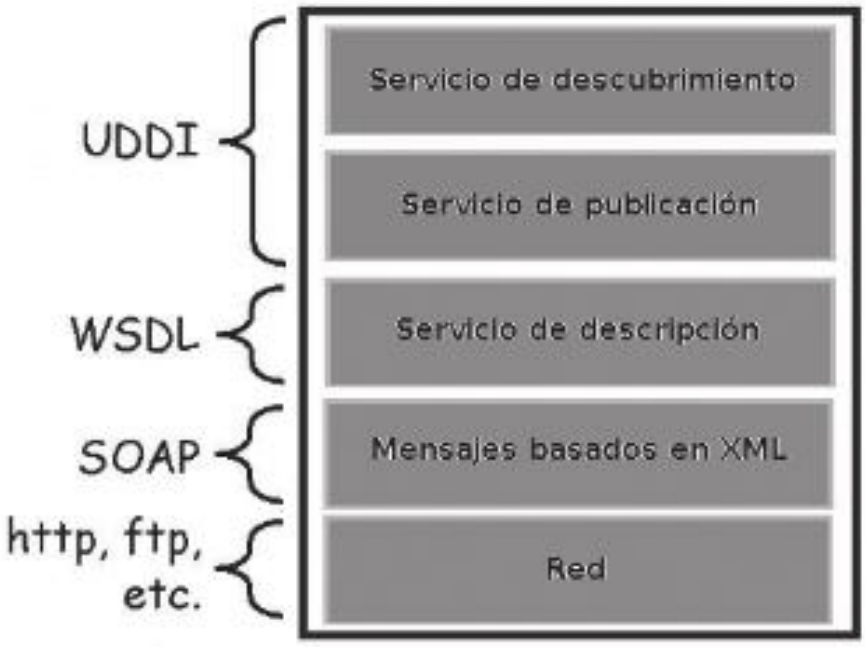
- Reconocer las características y ámbito de aplicación de servicios web.
- Reconocer las ventajas de utilizar servicios web para dar funcionalidad a la lógica de la capa de negocio de la aplicación.
- Identificar las tecnologías y protocolos para publicar y utilizar servicios web.
- Programar un servicio web u crear el documento de descripción del mismo.
- Verificar y consumir el funcionamiento de un servicio web.

7.1.- Servicios Web.

Un servicio Web es un conjunto de protocolos y estándares que permiten comunicar dos sistemas a través de una red aunque estén en plataformas diferentes. Habitualmente los servicios Web actúan para intercambiar datos entre dos aplicaciones aunque estén desarrolladas en lenguajes de programación distintos.

Un ejemplo sería el servicio de pago que una tienda online ofrece a sus clientes y que debe comunicarse con la aplicación de la entidad bancaria para formalizar el pago.

Los protocolos que intervienen en la comunicación y permiten intercambiar datos entre las aplicaciones se agrupan en un conjunto llamada Pila de protocolos de Servicios Web.



Servicio de transporte
Es el responsable del envío de mensajes entre las aplicaciones a través de la red. Se encarga de cómo se codifica la información, sin preocuparse de su formato. Generalmente utiliza el protocolo HTTP.

Servicio de mensajería
Es responsable de la codificación de los mensajes. En él se especifica que contienen los datos que se intercambian entre las maquinas. El lenguaje utilizado para los mensajes es XML.
Existen varios protocolos para este servicio, que interactúan con el lenguaje XML para ofrecer el servicio de mensajería: SOAP, XML-RPC, REST.

Servicio de descripción
Para comunicarse el cliente de un servicio Web y el propio servicio Web, tienen que llegar a un acuerdo. Se debe decidir los detalles del transporte de los mensajes y el contenido de los mismos a través de un documento(descripción del servicio) que se utiliza para describir la interfaz pública del Servicio Web. WSDL se encarga de publicar la interfaz pública de un servicio Web, es decir, especificar la sintaxis y los mecanismos para el intercambio de mensajes.

Servicio de descubrimiento
Es el que centraliza un registro común de servicios Web, de manera que las empresas que generan servicios Web, puedan publicar su localización y descripción. UDDI es un servicio de directorio donde las empresas pueden registrar y buscar servicios Web. A los documentos almacenados en sistemas UDDI se los denomina ficheros de registro y constan de las siguientes partes:

- **Paginas blancas**: especifica los datos personales de la empresa propietaria del servicio como la dirección, contactos, identificación de la empresa y otros identificadores conocidos.
- **Páginas amarillas**: se especifica la categoría industrial.
- **Páginas verdes**: describe la información técnica del servicio Web.

7.2.- Tratamiento de datos en formato JSON.

Para tratar datos en formato JSON disponemos de varias funciones que permiten recuperar los datos, codificarlos y decodificarlos.

La función file_get_contents() convierte la información devuelta por una URL a una cadena.

```
$cadena = file_get_contents(url);
```

La función json_decode() convierte una cadena en un array.

```
json_decode(cadena, true);
```

La función json_encode() convierte un array en una cadena.

```
$cadena = json_encode(array);
```

Ejemplo: Mostrar los datos de un empleado desde una URL transfiriendo los datos en formato JSON

json/consulta.php

```
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Consulta Empleado</title>
  <style>
    .tabla{
      border: 1px solid black;
      border-collapse: collapse;
      margin: 30px;
    }
    .cabecera {
      border: 1px solid black;
      background-color: red;
      color: white;
      width: 100px;
    }
    .celda {
      border: 1px solid black;
      width: 150px;
      text-align: right;
    }
  </style>
</head>
<body>
  <?php
  if(isset($_GET['codigo'])){
    $codigo=$_GET['codigo'];
    $url="http://localhost/json/busca.php?codigo=".$codigo;
    $json = file_get_contents($url);
    $array = json_decode($json, true);

    if(sizeof($array)>0){
      echo "<table class='tabla'>";
      foreach($array as $key=>$valor){
        echo "<tr><td class='cabecera'>".$key."</td><td
class='celda'>".$valor."</td></tr>";
      }
      echo "</table>";
    } else {
      echo "Empleado no Existe";
    }
  }
  ?>
  <form action="" method="GET">
```

```
    <label for="direccion">Introducir Código:</label>
    <input type="text" name="codigo">
    <button type="submit">Consultar</button>
  </form>
</body>
</html>
```

json/busca.php

```
<?php
$codigo=$_GET['codigo'];

$datos = array(    array("codigo"=>100, "nombre"=>"Luis Ros Sala",
                         "edad"=>30,"ciudad"=>"New York"),
                   array("codigo"=>200, "nombre"=>"Ana Molero Mas",
                         "edad"=>56,"ciudad"=>"Valencia"),
                   array("codigo"=>300, "nombre"=>"Roberto Bas Moreno",
                         "edad"=>45,"ciudad"=>"Barcelona"),
                   array("codigo"=>400, "nombre"=>"Sara Solis Bosch",
                         "edad"=>20,"ciudad"=>"Madrid"));
foreach($datos as $key=>$valor){
  if ($valor["codigo"]==$codigo){
    $clave=$key;
  }
}
$myJSON = json_encode($datos[$clave]);

echo $myJSON;
?>
```

7.3.- Tratamiento de datos en formato XML.

Para tratar datos en formato XML disponemos de varias funciones que permiten recuperar los datos, codificarlos y decodificarlos.

La función file_get_contents() convierte la información devuelta por una URL a una cadena.

```
$cadena = file_get_contents(url);
```

La función simplexml_load_string () convierte una cadena XML en un array.

```
$array = (array)simplexml_load_string(cadena);
```

La función asXML() del objeto SimpleXMLElement convierte XML a una cadena XML válida.

```
$xml_info = new SimpleXMLElement("<?xml version=\"1.0\"?><elemento></elemento>");
$result = $xml_info->asXML();
```

No existe una función que convierta directamente array a XML, por lo que debemos crear nuestra propia función recursiva que convierta.

Ejemplo: Mostrar los datos de un empleado desde una URL transfiriendo los datos en formato JSON

xml/consultaXML.php

```
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Consulta Empleado</title>
  <style>
    .tabla{
      border: 1px solid black;
      border-collapse: collapse;
      margin: 30px;
    }
    .cabecera {
      border: 1px solid black;
      background-color: red;
      color: white;
      width: 100px;
    }
    .celda {
      border: 1px solid black;
      width: 150px;
      text-align: right;
    }
  </style>
</head>
<body>
  <?php
  if(isset($_GET['codigo'])){
    $codigo=$_GET['codigo'];
    $url="http://localhost/xml/buscaXML.php?codigo=".$codigo;
    $xml = file_get_contents($url);
    $array = (array)simplexml_load_string($xml);
    if(sizeof($array)>0){
      echo "<table class='tabla'>";
      foreach($array as $key=>$valor){
        echo "<tr><td class='cabecera'>".$key."</td><td
class='celda'>".$valor."</td></tr>";
      }
      echo "</table>";
    } else {
      echo "Empleado no Existe";
    }
  }
  ?>
  <form action="" method="GET">
    <label for="direccion">Introducir Código:</label>
```

```
    <input type="text" name="codigo">
    <button type="submit">Consultar</button>
  </form>
</body>
</html>
```

xml/buscaXML.php

```
<?php
$codigo=$_GET['codigo'];

$datos = array(   array("codigo"=>100,   "nombre"=>"Luis   Ros   Sala",
                              "edad"=>30,"ciudad"=>"New York"),
          array("codigo"=>200,     "nombre"=>"Ana     Molero     Mas",
                              "edad"=>56,"ciudad"=>"Valencia"),
          array("codigo"=>300,    "nombre"=>"Roberto    Bas    Moreno",
                              "edad"=>45,"ciudad"=>"Barcelona"),
          array("codigo"=>400, "nombre"=>"Sara Solis Bosch",
                              "edad"=>20,"ciudad"=>"Madrid"));

function array_to_xml($array, &$xml_info, $codigo) {
  $encontrado=false;
  foreach($array as $key => $value) {
    if($value['codigo']==$codigo){
      $array = $value;
      $encontrado=true;
      break;
    }
  }
  if ($encontrado){
    foreach($array as $key => $value) {
      if(is_array($value)) {
        if(!is_numeric($key)){
          $subnode = $xml_info->addChild("$key");
          array_to_xml($value, $subnode, $codigo);
        }else{
          $subnode = $xml_info->addChild("item$key");
          array_to_xml($value, $subnode, $codigo);
        }
      }else {
        $xml_info->addChild("$key",htmlspecialchars("$value"));
      }
    }
  }
}
//Creamos un objeto del tipo SimpleXMLElement
$xml_info = new SimpleXMLElement("<?xml version=\"1.0\"?>
<empleado></empleado>");
```

```
// Llamamos a la función que convierte array en XML
array_to_xml($datos,$xml_info, $codigo);

// Convertimos a string XML válido
$result = $xml_info->asXML();

echo $result;
?>
```

7.4.- Acceso a un Web Service API con PHP y MySQL

¿Qué es un API?

Es un método generalizado utilizado para consumir un servicio. Creamos un API con el objetivo de que cualquier software pueda "consumir" nuestros recursos, por lo general datos, sin importar en que lenguaje o plataforma en que sea creado.

RESTful API es el sucesor de métodos anteriores como SOAP y WSDL cuya implementación y uso son un poco más complejos y requieren mayores recursos y especificaciones al ser usados.

¿Cómo funciona un API?

Un RESTful API, es un estándar para compartir información, en un sistema de Consulta y Respuesta (Request -> Response).

Al consultar una API se deben especificar parámetros de consulta para que el servicio sepa lo que queremos consultar, basado en una estructura previamente definida provista por el API por medio de una documentación.

La arquitectura REST (del inglés: Representational State Transfer) trabaja sobre el protocolo HTTP. Por consiguiente, los procedimientos o métodos de comunicación son los mismos que HTTP, siendo los principales: GET, POST, PUT, DELETE.

Otro componente de un RESTful API es el "HTTP Status Code", que le informa al cliente o consumidor del API que debe hacer con la respuesta recibida. Estos son una referencia universal de resultado, es decir, al momento de diseñar un RESTful API toma en cuenta utilizar el "Status Code" de forma correcta.

Por ejemplo, el código 200 significa OK, que la consulta ha recibido respuesta desde el servidor o proveedor del API. Los código de estado más utilizados son:

- 200 OK
- 201 Created (Creado)
- 304 Not Modified (No modificado)
- 400 Bad Request (Error de consulta)
- 401 Unauthorized (No autorizado)
- 403 Forbidden (Prohibido)
- 404 Not Found (No encontrado)

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

Por lo general la respuesta de un API es una estructura en formato jSON. Aunque también puede ser una estructura XML o cualquier otra estructura de datos de intercambio, incluso una personalizada. Sin embargo, como el objetivo es permitir que cualquier cliente pueda consumir el servicio de un API, lo ideal es mantener una estructura estándar, por lo que jSON es la mejor opción.
Puedes conocer los shares y comentarios de una URL de un website cualquiera con consultar el API de Facebook desde el navegador.

https://graph.facebook.com/?ids=http://www.nasa.gov

Y así tenemos una estructura que puede ser "consumida" desde cualquier plataforma o software sin importar el lenguaje que se utilice, debido a la simple, organizada y estandarizada forma de presentar los datos.

```
{
   "http://www.nasa.gov": {
      "share": {
         "comment_count": 2,
         "share_count": 113932
      },
      "og_object": {
         "id": "10150653814831879",
         "description": "NASA.gov brings you the latest news, images
and videos from America's space agency, pioneering the future in space
exploration, scientific discovery and aeronautics research.",
         "title": "National Aeronautics and Space Administration",
         "type": "website",
         "updated_time": "2018-01-06T05:30:39+0000"
      },
      "id": "http://www.nasa.gov"
   }
}
```

¿Qué es y para que sirve JSON?

JSON (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente. JSON nació como una alternativa a XML. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

Ejemplo de JSON:
Imaginemos que tenemos una frutería y que queremos obtener el nombre y la cantidad de fruta y verdura que tenemos. En un principio vamos a suponer que tenemos lo siguiente:

- Fruta:

    - 10 manzanas
    - 20 Peras
    - 30 Naranjas
- Verduras
    - 80 lechugas
    - 15 tomates
    - 50 pepinos

Sintaxis de Json:

Para asignar a un nombre un valor debemos usar los dos puntos ':' este separador es el equivalente al igual ('=') de cualquier lenguaje.

```
"Nombre" : "Aula Campus"
```

Valores Json

Los tipos de valores que podemos encontrar en Json son los siguientes:

- Un número (entero o float)
- Un string (entre comillas simples)
- Un booleano (true o false)
- Un array (entre corchetes [])
- Un objeto (entre llaves {})
- Null

Arrays JSON

En un Json puedes incluir arrays, para ellos el contenido del array debe ir entre corchetes []:

```
{
"Frutas": [
{ "NombreFruta":"Manzana" , "cantidad":10 },
{ "NombreFruta":"Pera" , "cantidad":20 },
{ "NombreFruta":"Naranja" , "cantidad":30 }
]
}
```

En el caso de la Frutería el ejemplo sería el siguiente:

```
{"Fruteria":
 [
  {"Fruta":
   [
    {"Nombre":"Manzana","Cantidad":10},
    {"Nombre":"Pera","Cantidad":20},
    {"Nombre":"Naranja","Cantidad":30}
   ]
  },
  {"Verdura":
```

```
        [
         {"Nombre":"Lechuga","Cantidad":80},
         {"Nombre":"Tomate","Cantidad":15},
         {"Nombre":"Pepino","Cantidad":50}
        ]
       }
      ]
     }
```

Como podemos observar, hemos creado un objeto llamado frutería y, dentro de ese objeto hemos almacenado un array de dos elementos. El primer elemento del array contiene un objeto llamado fruta y el segundo elemento del array contiene otro objeto llamado verdura. Estos objetos a su vez contienen un array cuyo contenido es el nombre y la cantidad de cada fruta o verdura.

Si queremos saber la cantidad de manzanas que tenemos. utilizaremos

```
['Fruteria'][0]['Fruta'][0]['Cantidad']
```

Observamos que la cantidad de manzanas se almacena dentro del primer elemento del array que contiene el objeto Frutería, y a su vez dentro del primer elemento del array que contiene el objeto Fruta.

Existen herramientas online que ayudan a visualizar mejor un JSON.

```
JSON Viewer. http://jsonviewer.stack.hu/
```

Si introducimos el ejemplo de la frutería obtenemos el siguiente resultado:

Creación de un JSON a partir de una consulta en MySQL

Los pasos que vamos a realizar para poder obtener los datos de una consulta en formato Json son:

- Generar una conexión a la base de datos.
- Obtener un array multidimensional de una consulta.
- Generar JSON a partir del array.

Generar una conexión a la base de datos

Para generar una consulta en una base de datos primero tenemos que generar una conexión con esta, para hacer esto en PHP podemos usar las funciones que vienen por defecto en PHP5, estas funciones ayudan mucho a la hora de gestionar la información de la base de datos. Una vez realizada la consulta debemos de eliminar la conexión a la base de datos.

Para conectarnos a la base de datos podemos usar:

```
function connectDB(){
   $conexion = mysqli_connect("SERVER", "USER", "PASS", "BASEDEDATOS");
   if($conexion){
      echo 'La conexión de la base de datos se ha hecho satisfactoriamente';
   } else{
      echo 'Ha sucedido un error en la conexión de la base de datos';
   }
   return $conexion;
}
```

Esta función devuelve en una variable la conexión a la base de datos.

Para la desconexión de la base de datos usamos la siguiente función:

```
function disconnectDB($conexion){
    $close = mysqli_close($conexion);
    if($close){
        echo 'La desconexión de la base datos se ha realizado';
    }else{
        echo 'Ha sucedido un error en la desconexión de la base de datos';
    }
    return $close;
}
```

La función devuelve un parámetro booleano indicando si la desconexión a tenido éxito o no.

Obtener un array multidimensional de una consulta

Para obtener un array multidimensional con el resultado de una consulta debemos realizar la consulta.

```
function getArraySQL($sql){
   //Creamos la conexión con la función anterior
   $conexion = connectDB();
   //generamos la consulta
```

```
      mysqli_set_charset($conexion, "utf8"); //formato de datos utf8
   if(!$result = mysqli_query($conexion, $sql)) die();
   //si la conexión cancelar programa
   $rawdata = array(); //creamos un array
   //guardamos en un array multidimensional todos los datos de la consulta
   $i=0;
   while($row = mysqli_fetch_array($result)) {
      $rawdata[$i] = $row;
      $i++;
   }
   disconnectDB($conexion); //desconectamos la base de datos
   return $rawdata; //devolvemos el array
}
```

La función devuelve un array multidimensional, es decir una matriz.

En el ejemplo de la Frutería tenemos una base de datos y en una tabla de esta base de datos tenemos 3 columnas, que son id_fruta, nombre_fruta y cantidad. Introducimos los siguientes datos:

id_fruta	nombre_fruta	cantidad
1	Manzana	100
2	Platano	167
3	Pera	820

Así pues, la función getArraySQL() obtendremos un array multidimensional. Si por ejemplo queremos obtener la cantidad de platanos tan sólo tendriamos que buscar en el array de la siguiente forma:

```
array[1][2];
```

Generar JSON a partir del array

Una vez que ya tenemos nuestro array multidimensional tan solo nos queda transformar los datos de nuestro array a un formato JSON. Para ello usaremos la función json_encode($array).

Si mostramos el resultado de esta función nos muestra:
[{"0":"1","id_fruta":"1","1":"Manzana","nombre_fruta":"Manzana","2":"100","cantidad":"100"},{"0":"2","id_fruta":"2","1":"Platano","nombre_fruta":"Platano","2":"167","cantidad":"167"},{"0":"3","id_fruta":"3","1":"Pera","nombre_fruta":"Pera","2":"820","cantidad":"820"}]

Podemos observar más gráficamente este JSON con la herramienta de visualización de JSON llamada JSONviewer.

Ejemplo completo:

```
<?php
```

```
function connectDB(){
   $server = "SERVER";
   $user = "USER";
   $pass = "PASS";
   $bd = "BD";
   $conexion = mysqli_connect($server, $user, $pass,$bd);
   if($conexion){
      echo 'La conexion de la base de datos se ha hecho satisfactoriamente';
   }else{
      echo 'Ha sucedido un error en la conexion de la base de datos';
   }
   return $conexion;
}
function disconnectDB($conexion){
   $close = mysqli_close($conexion);
   if($close){
      echo 'La desconexion de la base de datos se ha realizado';
   }else{
      echo 'Ha sucedido un error en la desconexion de la base de datos';
   }
   return $close;
}
function getArraySQL($sql){
   //Creamos la conexión con la función anterior
   $conexion = connectDB();
   //generamos la consulta
   mysqli_set_charset($conexion, "utf8"); //formato de datos utf8
   if(!$result = mysqli_query($conexion, $sql)) die();
   //si la conexión cancelar programa
   $rawdata = array(); //creamos un array
   //guardamos en un array multidimensional todos los datos de la consulta
   $i=0;
   while($row = mysqli_fetch_array($result)) {
      $rawdata[$i] = $row;
      $i++;
   }
   disconnectDB($conexion); //desconectamos la base de datos
   return $rawdata; //devolvemos el array
}

// Generar Json con el resultado de una Consulta
$sql = "SELECT id_fruta,nombre_fruta,cantidad FROM tabla_fruta";
$myArray = getArraySQL($sql);
echo json_encode($myArray);

?>
```

Usar la función json_decode para decodificar un JSON.

Suponiendo que tenemos el siguiente JSON generado:

[{"0":"1","id_fruta":"1","1":"Manzana","nombre_fruta":"Manzana","2":"100","cantidad":"100"},{"0":"2","id_fruta":"2","1":"Platano","nombre_fruta":"Platano","2":"167","cantidad":"167"},{"0":"3","id_fruta":"3","1":"Pera","nombre_fruta":"Pera","2":"820","cantidad":"820"}]

Usaremos la función json_decode sobre este JSON, para guardar en un array el resultado de decodificar el JSON.

```
$json='[{"0":"1","id_fruta":"1","1":"Manzana","nombre_fruta":"Manzana","2":"100","cantidad":"100"},{"0":"2","id_fruta":"2","1":"Platano","nombre_fruta":"Platano","2":"167","cantidad":"167"},{"0":"3","id_fruta":"3","1":"Pera","nombre_fruta":"Pera","2":"820","cantidad":"820"}]';
$array = json_decode($json);
```

La funciona la función json_decode.

JSON se compone de JSON Objects y de JSON Arrays. Así pues la función json_decode decodifica el String que le introducimos y separa los JSON Objects y los JSON Arrays de tal forma que una vez que se han separado (virtualmente) se vuelven a componer en un Array que será la variable de salida $array.

Para observar que se ha generado un Array se puede usar la función "print_r" sobre la variable $array de salida.

```
print_r($array);
```

Mostrará lo siguiente:
Array ([0] => stdClass Object ([0] => 1 [id_fruta] => 1 [1] => Manzana [nombre_fruta] => Manzana [2] => 100 [cantidad] => 100) [1] => stdClass Object ([0] => 2 [id_fruta] => 2 [1] => Platano [nombre_fruta] => Platano [2] => 167 [cantidad] => 167) [2] => stdClass Object ([0] => 3 [id_fruta] => 3 [1] => Pera [nombre_fruta] => Pera [2] => 820 [cantidad] => 820))

Se ha creado una key por cada campo de nuestro JSON, es decir podemos obtener el dato directamente escribiendo el índice del array o el nombre del mismo.

Ejemplos de uso.

Obtener un dato directamente del Array.

```
echo $array[0]->nombre_fruta;
//Obtenemos "Manzana"
```

Recorrer y recuperar valores de un objeto JSON con foreach.

```
foreach($array as $obj){
        $id_fruta = $obj->id_fruta;
        $nombre_fruta = $obj->nombre_fruta;
        $cantidad = $obj->cantidad;
        echo $id_fruta." ".$nombre_fruta." ".$cantidad;
        echo "<br>";
}
```

El resultado de este bucle es el siguiente:

```
1 Manzana 100
2 Plátano 167
3 Pera 820
```

Recorrer y recuperar valores de un objeto JSON con un bucle for.

```
for($i=0;$i<count($array);$i++){
      $id_fruta="$array[$i]->id_fruta;
      $nombre_fruta = $array[$i]->nombre_fruta;
      $cantidad = $array[$i]->cantidad;
      echo $id_fruta." ".$nombre_fruta." ".$cantidad;
      echo "<br> ";
}
```

El resultado es el mismo:

```
1 Manzana 100
2 Plátano 167
3 Pera 820
```

Ejemplo práctico de uso de JSON con OpenWeatherMap.

Para ver un ejemplo de uso de JSON vamos a usar la API de OpenWeatherMap, esta API proporciona información general sobre el tiempo (temperatura, localización, etc) en función de una localización.

Para usar esta API debemos registrarnos en la web y nos proporcionan por correo una clave para poder utilizar la API. Una vez obtenida la clave de la API podemos utilizarla mediante la llamada a la URL con los parámetros de la ciudad, la unidad de medida de la temperatura y la clave API.

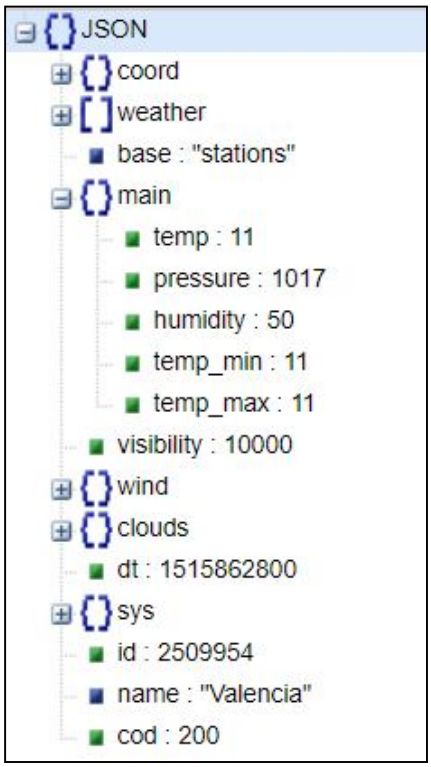
```
$url="http://api.openweathermap.org/data/2.5/weather?q=Valencia,es&units=metric&APPID=a77a241542b9ed8ac5b8e533f10c2f0a";
```

En este caso buscamos la información del tiempo de Valencia con unidad de medida metric que muestra la información de temperatura en grados Celsius.

Obtendremos el siguiente JSON:


```
{
  "coord": {
    "lon": -0.38,
    "lat": 39.47
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 11,
    "pressure": 1017,
    "humidity": 50,
    "temp_min": 11,
    "temp_max": 11
  },
  "visibility": 10000,
  "wind": {
    "speed": 1
  },
  "clouds": {
    "all": 0
  },
  "dt": 1515862800,
  "sys": {
    "type": 1,
    "id": 5503,
    "message": 0.0025,
    "country": "ES",
    "sunrise": 1515828035,
    "sunset": 1515862838
  },
  "id": 2509954,
  "name": "Valencia",
  "cod": 200
}
```


Usando el visualizador JSONViewer, obtenemos la siguiente información:



Para crear una aplicación en PHP que lea el JSON obtenido a partir del servicio web de OpenWeatherMap. Los datos que queremos obtener son los siguientes:

- Ciudad
- Temperatura
- Temperatura Máx
- Temperatura Mín
- Humedad

Para realizar un script/app en PHP que lea el JSON generado por la API de OpenWeatherMap y muestre la información útil por pantalla realizaremos el siguiente código.

En primer lugar, crearemos un formulario donde introduzcamos la ciudad donde queremos obtener los datos del tiempo y al pulsar un botón Mostrar Datos envía por GET la ciudad. La ciudad introducida en el formulario y enviada es almacenada en una variable y montamos la URL que permite acceder a la API y devolver los valores en JSON.

La función file_get_contents obtiene la información de la API en formato JSON y lo almacenamos en la variable $json. A continuación vamos a decodificar en un array el JSON en 'bruto' que nos envía la API de OpenWeatherMap con la función json_decode.

Una vez que tenemos almacenado en la variable toda la información del JSON obtenido de la API vamos a extraer la información útil de él, para ello vamos a almacenar en variables con nombres orientativos la información del JSON:

```
$ciudad=$array['name'];
$temp_max=$array['main']['temp_max'];
$temp_min=$array['main']['temp_min'];
$humedad=$array['main']['humidity'];
```

Lo último mostraremos la información con echos.

El código completo es el siguiente:

```
<?php
    if(isset($_POST['enviar'])){
        $ciudad=$_POST['ciudad'];

    $url="http://api.openweathermap.org/data/2.5/weather?q=".$ciudad."
&units=metric&APPID=a77a241542b9ed8ac5b8e533f10c2f0a";
        $json = file_get_contents($url);
        $array = json_decode($json, true);
        $ciudad=$array['name'];
        $temp_max=$array['main']['temp_max'];
        $temp_min=$array['main']['temp_min'];
        $humedad=$array['main']['humidity'];
        echo "Ciudad: ".$ciudad."<br>";
        echo "Temp. Máxima: ".$temp_max."<br>";
        echo "Temp. Mínima: ".$temp_min."<br>";
        echo "Humedad: ".$humedad."<hr>";
    }
?>
<form action="" method="POST">
Ciudad: <input type="text" id="ciudad" name="ciudad">
<input    type="submit"    value="Mostrar    Información"    id="enviar"
name="enviar">
</form>
```

JSON proporcionar información a todas las plataformas que queramos como por ejemplo PHP, Android, Java, Python, etc...

Actividad.

Crear una página que permita introducir en un formulario una URL y nos devuelva la URL, la descripción de la página y en número de seguidores de facebook utilizando su API desde la URL:

```
https://graph.facebook.com/?ids=direccionURL
```