



**Materia:** Tecnologías para la Web.  
**Tema:** JSON.

## PARTE I.

### ¿Qué es y para qué sirve JSON?

**JSON (JavaScript Object Notation)** es un formato para el intercambio de datos. JSON describe los datos con una sintaxis dedicada que se usa para identificar e intercambiar datos. JSON es una alternativa a XML, el fácil uso en JavaScript ha generado muchos seguidores y su mayor ventaja es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, se usa para el intercambio de información entre distintas tecnologías.

### **Ejemplo de JSON:**

En una frutería se desea obtener el nombre y la cantidad de fruta y verdura que se tienen. En un principio se tiene lo siguiente:

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

La sintaxis de JSON es:

**JSON Nombre/Par de Valores**

Para asignar un **Valor** a un **Nombre** se usan los dos puntos ':' este separador es el equivalente al igual ('=') de cualquier lenguaje, por ejemplo:

"Nombre" : "Geeky Theory"

### **Valores JSON**

Los tipos de valores que se encuentran en JSON son los siguientes:

- Un número (entero o flotante)
- Un string (entre comillas simples)
- Un boolean (true o false)
- Un arreglo (entre corchetes [])
- Un objeto (entre llaves {})
- Null

### **Objetos JSON**

Los objetos JSON se identifican entre llaves, un objeto puede ser en nuestro caso una fruta o una verdura.

```
{ "NombreFruta": "Manzana" , "Cantidad": 20 }
```

### **Arreglos con JSON.**

En un JSON se pueden incluir arreglos, y su contenido debe ir entre los corchetes [], por ejemplo:

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



Una vez explicado el funcionamiento de la sintaxis JSON, se aplica al ejemplo de la frutería.

```
{ "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 }  
      ]  
    }  
  ]  
}
```

Se ha creado un objeto llamado *Frutería* y, dentro de ese objeto, se ha almacenado un arreglo de dos elementos. El primer elemento del arreglo contiene un objeto llamado *Fruta* y el segundo elemento del arreglo contiene otro objeto llamado *Verdura*.

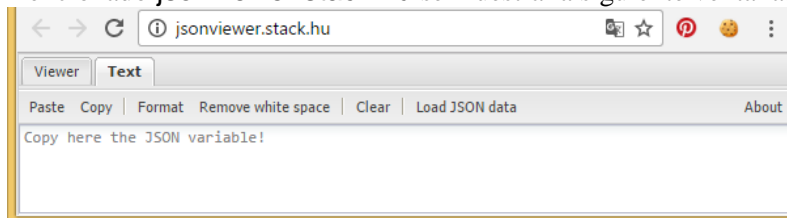
Estos objetos a su vez contienen un arreglo cuyo contenido es el nombre y la cantidad de cada fruta o verdura.

Si se deseara conocer la cantidad de manzanas que se tiene, entonces la ruta hacia el arreglo sería el siguiente:

Path: `json['Fruteria'][0]['Fruta'][0]['Cantidad']`

Observar que la cantidad de manzanas se almacena dentro del primer elemento del arreglo que contiene el objeto *Frutería*, y a su vez dentro del primer elemento del arreglo que contiene el objeto *Fruta*. La página [JSON Viewer](#) posee una herramienta para JSON. Si se introduce el ejemplo se observa lo siguiente:

- a. Al ingresar al sitio mencionado [jsonviewer.stack.hu](http://jsonviewer.stack.hu) se muestra la siguiente ventana:

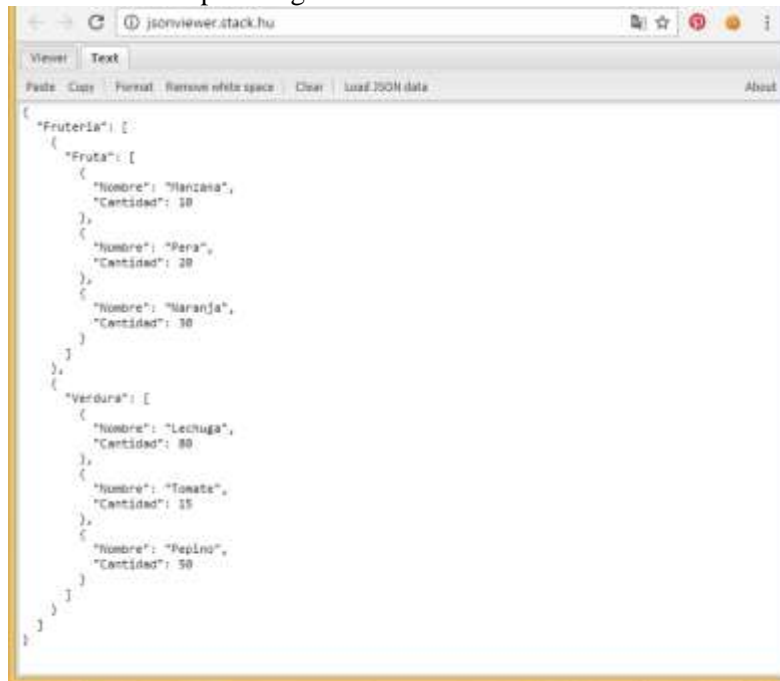


- b. Ingresar o arrastrar el texto del dato en formato JSON al área mostrada en la pestaña **Text**:




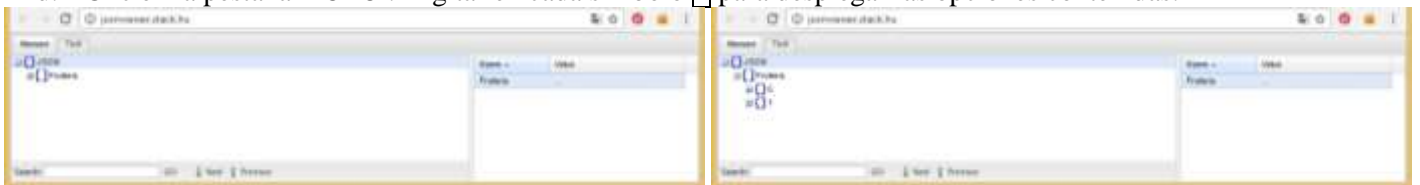


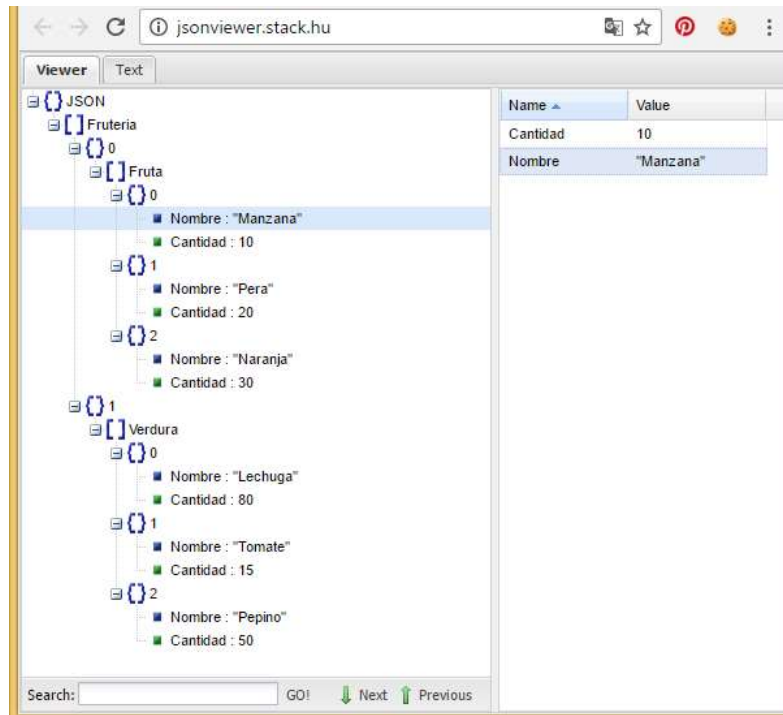
c. Clic en la opción **Format** del menú para asignar formato al texto:



```
{
  "Frutaria": [
    {
      "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
    }
  ]
}
```

d. Clic en la pestaña **Viewer**. Digitar en cada símbolo  para desplegar las opciones contenidas:





## PARTE II.

### Creación de un JSON a partir de una consulta en MySQL.

1. Generar una conexión a la base de datos.
2. Obtener un arreglo multidimensional de una consulta.
3. Generar JSON a partir del arreglo.

#### 1. Generar una conexión a la base de datos

Para generar una consulta en una base de datos primero se genera una conexión con ella. Para ello, en PHP se usan las funciones de **PHP5**, estas funciones ayudan a gestionar la información de la base de datos. Una vez realizada la consulta se debe eliminar la conexión a la base de datos.

Para conectarse a la base de datos se diseña la siguiente función:

```
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 inesperado en la conexión de la base de datos';
    }
    return $conexion;
}
```

Esta función devuelve en una variable la conexión a la base de datos, esta conexión será importante para realizar la consulta.

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 de datos se ha hecho satisfactoriamente';
    }else{
        echo 'Ha sucedido un error inesperado en la desconexión de la base de datos';
    }
}
```



```

    }
    return $close;
}

```

Esta función devuelve un parámetro booleano indicando si la desconexión ha tenido éxito o no.

## 2. Obtener un arreglo multidimensional de una consulta

Para realizar la consulta se diseña otra función que tiene como parámetro de entrada una instrucción SQL.

```

function getArraySQL($sql){
    //Crear la conexión con la función anterior
    $conexion = connectDB();
    //generar la consulta
    mysqli_set_charset($conexion, "utf8"); //formato de datos utf8
    if(!$result = mysqli_query($conexion, $sql)) die(); //si la conexión cancela programa
    $rawdata = array(); //creamos un array
    //guardar en un arreglo multidimensional todos los datos de la consulta
    $i=0;
    while($row = mysqli_fetch_array($result)){
        $rawdata[$i] = $row;
        $i++;
    }
    disconnectDB($conexion); //desconectar la base de datos
    return $rawdata; //devolvemos el array
}

```

Se observa que esta función devuelve una matriz. Por ejemplo: imaginar que se tiene una base de datos de una frutería, en una tabla de esta base de datos se tienen tres columnas, que son `id_fruta`, `nombre_fruta` y `cantidad`. Por ejemplo:

<code>id_fruta</code>	<code>nombre_fruta</code>	<code>cantidad</code>
1	Manzana	100
2	Plátano	167
3	Pera	820

Si se pasa la instrucción SQL que genera esta tabla en la función, se obtendrá un arreglo multidimensional. Si se desea obtener la cantidad de plátanos, se tendría que buscar en el arreglo con lo siguiente:

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

**Nota:** Los índices de un arreglo inician con cero.

## 3. Generar JSON a partir del arreglo.

Una vez que se tiene la matriz, se transforman los datos del arreglo a un formato JSON y se usa la función `json_encode($array)`.

Si se hace un echo del resultado de esta función se muestra algo similar a lo siguiente:

```

[{"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"}]

```

El ejemplo completo es el siguiente, donde se tendrá que rellenar las variables que aparecen al inicio del script.

```
sql = "SQL" //ejemplo frutería: SELECT id_fruta,nombre_fruta,cantidad FROM tabla_fruta;
```

```

function connectDB(){
    $server = "SERVER";
    $user = "USER";
}

```



```

    $pass = "PASS";
    $bd = "BD";
    $conexion = mysqli_connect($server, $user, $pass,$bd);
    if($conexion){
        echo 'La conexión de la base de datos se ha hecho satisfactoriamente';
    }else{
        echo 'Ha sucedido un error inesperado en la conexión de la base de datos';
    }
    return $conexion;
}

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

    $myArray = getArraySQL($sql);
    echo json_encode($myArray);
?>

```

## PARTE III.

### Gestionar JSON en PHP.

Para gestionar **JSON en PHP** se realizan los pasos siguientes:

1. Usar la función **json\_decode** para decodificar un JSON.
2. Entender cómo funciona la función **json\_decode**.
3. Ejemplos de uso.

#### **1. Usar la función json\_decode para decodificar un JSON.**

Partiendo del ejercicio anterior y suponiendo que se tiene un JSON generado como este:

```

[{"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"}]

```



Se usa la función `json_decode` sobre este JSON; suponer que el JSON es un `JSONArray`, es decir que es un arreglo de datos y el resultado se guardará en un arreglo al 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);
```

## 2. ¿Cómo funciona la función `json_decode`?

Revisar el primer ejercicio, en donde se observa que JSON se compone de JSON Objects y de JSON Arrays. La función `json_decode` decodifica el **String** que se le introduce 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 arreglo, que en este caso será la variable de salida: `$array`.

Observar que se ha generado un arreglo y se puede usar la función **`print_r`** sobre la variable `$array` de salida.

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

En este ejemplo se 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 ) )
```

Observar cómo se ha creado una **key** por cada campo del JSON; se puede obtener el dato directamente escribiendo el índice del arreglo o el nombre del mismo.

## 3. Ejemplos de uso.

- Obtener un dato directamente del Array:  

```
echo $array[0]->nombre_fruta;
```

El resultado es: **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 "\n";
}
```

El resultado del ciclo es el siguiente:

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

- Recorrer y recuperar valores de un objeto JSON con un ciclo `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 "  
}
```

El resultado es el mismo que tiene el apartado anterior

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

## PARTE IV.

### Ejemplo práctico de uso de JSON con OpenWeatherMap.

Para este ejemplo de uso de JSON se usa la API de OpenWeatherMap. Esta API proporciona información general sobre el tiempo (temperatura, localización, etc.) en función de una localización.

Para usar esta API se debe incluir el nombre del lugar de donde se desea obtener el tiempo al final del siguiente enlace:

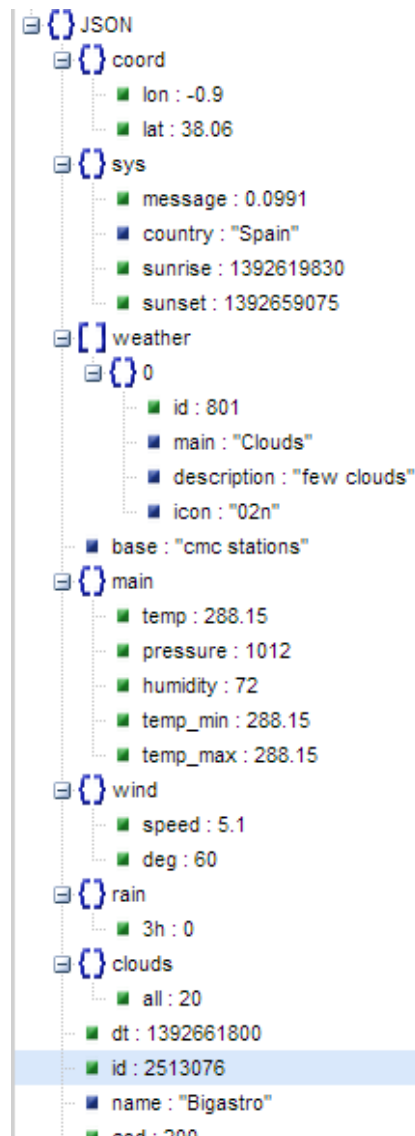
<http://api.openweathermap.org/data/2.5/weather?q=bigastro>

En este caso se busca la información del tiempo de un sitio, por ejemplo el pueblo Bigastro. Si se usa como ejemplo esta población se obtendrá un JSON como este:

```
{ "coord": { "lon": -  
0.9, "lat": 38.06 }, "sys": { "message": 0.0991, "country": "Spain", "sunrise": 1392619830, "sunset":  
1392659075 }, "weather": [ { "id": 801, "main": "Clouds", "description": "few  
clouds", "icon": "02n" } ], "base": "cmc  
stations", "main": { "temp": 288.15, "pressure": 1012, "humidity": 72, "temp_min": 288.15, "temp_max  
": 288.15 }, "wind": { "speed": 5.1, "deg": 60 }, "rain": { "3h": 0 }, "clouds": { "all": 20 }, "dt": 13926618  
00, "id": 2513076, "name": "Bigastro", "cod": 200 }
```

Usando el visualizador JSON Viewer, que se mencionó en el ejercicio anterior, se obtiene la siguiente información:





Como ejemplo, se crea una aplicación en PHP que lea el JSON obtenido a partir del servicio web de OpenWeatherMap, enseguida se extrae la información útil de este JSON. Los datos que se obtienen son los siguientes:

- Ciudad
- Latitud
- Longitud
- Temperatura
- Temperatura Máx.
- Temperatura Mín.
- Presión
- Humedad
- Velocidad del viento
- Estado del cielo
- Descripción del estado del cielo

Enseguida se realiza un script/app en PHP que lea el JSON generado por la API de OpenWeatherMap y que muestre la información.



Primero, se crea un formulario donde se introduzca la ciudad donde se desea obtener los datos del tiempo:

```
<meta charset="UTF-8">

<form method="get" action="">
  <label>
    Ciudad
  </label>
  <input type = "text" name="c">
  <input type = "submit" value="Mostrar el Tiempo">
</form>
```

Este formulario envía los datos de un input a sí misma con el método **GET**. Estos datos son procesados; en primer lugar los datos del input no vienen vacíos, por ello se escribe la siguiente instrucción:

```
if($_GET["c"]==null) die();
```

Seguidamente, si el campo no está vacío, se va a obtener el JSON que genera la API de OpenWeatherMap con la función `file_get_contents()`, el contenido de este JSON se almacenará en la variable `$html`:

```
$html = file_get_contents("http://api.openweathermap.org/data/2.5/weather?q=".$_GET["c"]);
```

Una vez obtenido todo el código `html`, es decir el JSON crudo que envía la API de OpenWeatherMap se decodifica el JSON con la función `json_decode`:

```
$json = json_decode($html);
```

Con lo almacenado en la variable `$json`, toda la información del JSON obtenida de la API se va a extraer la información de él; para ello, se va a almacenar en variables, con nombres significativos, la información del JSON:

```
$ciudad = $json->name;
$lat = $json->coord->lat;
$lon = $json->coord->lon;
$temp = $json->main->temp;
$tempmax = $json->main->temp_max;
$tempmin = $json->main->temp_min;
$presion = $json->main->pressure;
$humedad = $json->main->humidity;
$vel_viento = $json->main->temp;
$estado_cielo = $json->weather[0]->main;
$descripcion = $json->weather[0]->description;
```

Para mostrar la información con el uso de `echo`, el código es el siguiente:

```
echo "<h3>Ciudad: ".$ciudad." [lat = ".$lat." , lon = ".$lon." ]</h3>";
echo "<b>Estado del cielo: </b>".$estado_cielo."<br>";
echo "<b>Descripción: </b>".$descripcion."<br>";
echo "<br>";
echo "<b>Temperatura: </b>".$temp." K [Máx: ".$tempmax."K, Mín: ".$tempmin."K]<br>";
echo "<b>Presión: </b>".$presion."<br>";
echo "<b>Humedad: </b>".$humedad."<br>";
echo "<br><br><br><br><br>";
```

Una captura de lo que se genera para la población Bigastro es el siguiente:



Ciudad

**Ciudad: Bigastro [lat = 38.06, lon = -0.9 ]**

**Estado del cielo:** Clouds

**Descripción:** few clouds

**Temperatura:** 291.15 K [Máx: 291.15K, Mín: 291.15K]

**Presión:** 1009

**Humedad:** 48

Por último, se escribe el código completo:

```
<meta charset="UTF-8">
<form method="get" action="">
    <label>
        Ciudad
    </label>
    <input type = "text" name="c">
    <input type = "submit" value="Mostrar el Tiempo">
</form>
<?php
if($_GET["c"]==null) die();
$html = file_get_contents("http://api.openweathermap.org/data/2.5/weather?q=".$_GET["c"]);
$json = json_decode($html);
$ciudad = $json->name;
$lat = $json->coord->lat;
$lon = $json->coord->lon;
$temp = $json->main->temp;
$tempmax = $json->main->temp_max;
$tempmin = $json->main->temp_min;
$presion = $json->main->pressure;
$humedad = $json->main->humidity;
$vel_viento = $json->main->temp;
$estado_cielo = $json->weather[0]->main;
$descripcion = $json->weather[0]->description;
echo "<h3>Ciudad: ".$ciudad." [lat = ".$lat." , lon = ".$lon." ]</h3>";
echo "<b>Estado del cielo: </b>".$estado_cielo."<br>";
echo "<b>Descripción: </b>".$descripcion."<br>";
echo "<br>";
echo "<b>Temperatura: </b>".$temp." K [Máx: ".$tempmax."K, Mín: ".$tempmin."K]<br>";
echo "<b>Presión: </b>".$presion."<br>";
echo "<b>Humedad: </b>".$humedad."<br>";
echo "<br><br><br><br><br>";
?>
```

**Nota:** Tener en cuenta que el objetivo de JSON no es sólo proporcionar información con PHP, sino proporcionarla a todas las plataformas que se desee, por ejemplo a Android, Java, Python, etc.

**Nota:** Usando como base este ejemplo, diseñar una aplicación Web que busque el lugar donde se encuentre automáticamente y muestre la información del tiempo.