



UTN.BA FACULTAD
REGIONAL
BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

**Centro de
e-Learning**

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL AVANZADO



www.sceu.frba.utn.edu.ar/e-learning



Módulo II

CONTINUAMOS CON PHP AVANZADO



PHP Y JSON.



Presentación de la Unidad:

En esta unidad aprenderán que es el formato JSON para intercambio de datos entre distintas plataformas.

Aprenderán a leer desde PHP la API de Youtube V3 que genera sus respuestas en dicho formato.



Objetivos:

- ❖ **Aprender el formato JSON de comunicación entre las distintas plataformas.**
- ❖ **Ver un ejemplo de uso de la API de Youtube V3 para acceder a alguno de los feeds que nos provee.**



Temario:

- *JSON*
- *¿Qué es JSON?*
- *¿Por qué JSON?*
- *Sintaxis de JSON*
- *JSON Uso de Sintaxis JavaScript*
- *Archivos JSON*
- *Como utilizar JSON*
- *Usar eval()*
- *JSON Http Request*
- *JSON Function Files*
- *Crear un JSON desde PHP*
- *Consejos para crear un JSON correctamente*
- *Crear objetos PHP a partir de una cadena JSON*
- *Ejercicio práctico Uso de la API v3 de youtube*

CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.



Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

JSON

JSON es una notación Javascript para escribir objetos que se ha hecho bastante popular en el mundo del desarrollo de webs y que se utiliza en diversos lenguajes de programación, componentes (habitualmente Ajax), etc. Su éxito se debe a que es una excelente forma para almacenar información que deseamos compartir entre distintos componentes o lenguajes de las aplicaciones web. Nos daremos cuenta que si trabajamos con Ajax y JQuery o alguno de los otros frameworks Javascript existentes, JSON nos será de gran utilidad.

JSON, cuyas siglas significan JavaScript Object Notation (en español Notación de Objetos de JavaScript), es un formato ligero, fácil de escribir o codificar, así como también es fácil de leer por los seres humanos. Desde Javascript podemos procesar directamente cualquier objeto JSON y existen librerías para la mayoría de los lenguajes de programación que tienen funciones para interpretar este formato. Por ello se ha adoptado universalmente. Para más información podemos visitar el sitio web de JSON en <http://www.json.org/>.

Esto quiere decir que con JSON podemos comunicar datos fácilmente entre scripts Javascript y scripts PHP. Por ejemplo, pensemos en una validación de formulario que se desea hacer con Ajax. Los datos del formulario se pueden enviar a PHP por medio de POST y luego podríamos desde PHP enviar a Javascript el resultado de validar esos datos en el servidor. Como la validación puede ser positiva o negativa, así como puede tener más o menos códigos de error y acciones a realizar dependiendo de la información procesada, el script PHP tiene que mandar una respuesta más o menos elaborada al script Javascript y una posibilidad es enviar esos datos desde PHP utilizando la notación JSON.

Como dijimos JSON es un formato de intercambio de datos ligero que se utiliza para comunicarse entre distintas plataformas y lenguajes de programación.

Es comparable con el lenguaje de etiquetas XML ya que cumple el mismo propósito. A continuación vemos un ejemplo de cada uno de ellos y luego hacemos una pequeña comparación.

El ejemplo siguiente define un objeto JSON de empleados, con una serie de 3 registros de empleados:

```
{"employees":  
  {"firstName":"John", "lastName":"Doe"},  
  {"firstName":"Anna", "lastName":"Smith"},  
  {"firstName":"Peter", "lastName":"Jones"}  
}]
```

El ejemplo siguiente de XML también define un objeto empleados con 3 registros de

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

empleados:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

¿QUÉ ES JSON?

- JSON significa Java Script Object Notation
- JSON es un formato de intercambio de datos ligero
- JSON es independiente del lenguaje *
- JSON es "auto-descriptivo" y fácil de entender

* JSON utiliza la sintaxis de JavaScript, pero el formato JSON es sólo texto, al igual que XML.



El texto puede ser leído y utilizado como un formato de datos por cualquier lenguaje de programación.

El formato JSON es sintácticamente idéntico al código para la creación de objetos de JavaScript.

Debido a esta similitud, en lugar de utilizar un programa de análisis (como hace XML), un programa de JavaScript puede utilizar las funciones estándar de JavaScript para convertir datos JSON en objetos nativos de JavaScript.

Código del ejemplo json1.html

```
<!DOCTYPE html>
<html>
<body>

<h2>JSON Object Creation in JavaScript</h2>

<p id="demo"></p>

<script>
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



```
var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}';

var obj = JSON.parse(text);

document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.street + "<br>" +
obj.phone;
</script>

</body>
</html>
```

Parecido a XML porque:

- Tanto JSON y XML es perfectamente legible
- Tanto JSON y XML es hierarchichal (valores dentro de los valores)
- Tanto JSON y XML se pueden analizar y ser utilizados por una gran cantidad de lenguajes de programación
- Tanto JSON y XML se pueden recuperar con un XMLHttpRequest

Diferente a XML porque:

- JSON no utiliza etiqueta final
- JSON es más corto
- JSON es más rápido de leer y escribir
- JSON puede utilizar matrices

La mayor diferencia es la siguiente:

Para analizar un XML hay que utilizar un analizador XML, JSON puede ser analizado por una función estándar de JavaScript.

¿POR QUÉ JSON?

Para aplicaciones AJAX, JSON es más rápido y más fácil que XML ya que XML requiere:

- Recuperar un documento XML
- Utilizar el DOM XML para recorrer el documento
- Extraer y almacenar los valores en las variables

SINTAXIS DE JSON

La sintaxis JSON es un subconjunto de la sintaxis de JavaScript.

Estas son sus características principales:

- Los datos son pares nombre / valor
- Los datos son separados por comas
- Las llaves tienen objetos
- Los corchetes tienen matrices

Los datos JSON se escriben como pares nombre / valor.

Un par nombre / valor consiste en un nombre de campo (entre comillas), seguido de dos puntos, seguido de un valor:

Ejemplo:

`"firstName":"John"`



Los nombres en JSON requieren comillas dobles. Los nombres de JavaScript no.

Los valores de JSON pueden ser:

- Un número (entero o punto flotante)
- Una cadena (entre comillas dobles)
- Booleano (verdadero o falso)
- Una matriz (entre corchetes)
- Un objeto (entre llaves)
- nulo

Objetos JSON

Los objetos JSON se escriben dentro de llaves.

Al igual que en JavaScript, los objetos JSON pueden contener múltiples pares nombre / valor:

Ejemplo:

`{"firstName":"John", "lastName":"Doe"}`

JSON Arrays

Los arrays en JSON se escriben entre corchetes.

Al igual que JavaScript, una matriz JSON puede contener varios objetos:

Ejemplo:

```
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
```

En el ejemplo anterior, el objeto "empleados" es una matriz que contiene tres objetos. Cada objeto es un registro de una persona (con un nombre y un apellido).

JSON USO DE SINTAXIS JAVASCRIPT

Debido a que la sintaxis JSON se deriva de la notación de objetos de JavaScript, no se necesita software adicional para trabajar con JSON en JavaScript.

Con JavaScript se puede crear una matriz de objetos y asignar los datos a ella, así:

Ejemplo:

```
var employees = [
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
];
```

La primera entrada de la tabla de objeto JavaScript se puede acceder de esta manera:

```
// returns John Doe
employees[0].firstName + " " + employees[0].lastName;
```

También se puede acceder de esta manera:

```
// returns John Doe
employees[0]["firstName"] + " " + employees[0]["lastName"];
```

Los datos pueden ser modificados de esta manera:

```
employees[0].firstName = "Gilbert";
```

También se puede modificar así:

```
employees[0]["firstName"] = "Gilbert";
```

ARCHIVOS JSON

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

- El tipo de archivo para los archivos JSON es ".json"
- El tipo MIME para el texto JSON es "application / json"

COMO UTILIZAR JSON

Ahora vamos a ver cómo convertir un texto JSON a un objeto JavaScript.

Un uso común de JSON es leer datos de un servidor web, y mostrar los datos en una página web.

Por simplicidad, este se puede demostrar mediante el uso de una cadena como de entrada (en lugar de un archivo).

Creamos un string en Javascript con sintaxis JSON:

```
var text = '{ "employees" : [' +  
'{ "firstName":"John" , "lastName":"Doe" },' +  
'{ "firstName":"Anna" , "lastName":"Smith" },' +  
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

La sintaxis de JSON es un subconjunto de la sintaxis de JavaScript.

La función de JavaScript `JSON.parse (text)` se puede utilizar para convertir un texto JSON en un objeto de JavaScript:

```
var obj = JSON.parse(text);
```

Ahora podemos usar en nuestra página el nuevo objeto Javascript de esta forma:

```
<p id="demo"></p>
```

```
<script>  
document.getElementById("demo").innerHTML =  
obj.employees[1].firstName + " " + obj.employees[1].lastName;  
</script>
```

Pueden ver este ejemplo completo en [json2.html](#)

USAR EVAL()

Los navegadores más antiguos, sin el apoyo de la función JavaScript `JSON.parse ()` pueden utilizar la función `eval ()` para convertir un texto JSON en un objeto de JavaScript:

```
var obj = eval("(" + text + ")");
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



La función `eval()` puede compilar y ejecutar cualquier JavaScript. Esto representa un potencial problema de seguridad. **Tenemos que tratar de evitarla.**

Es más seguro utilizar un analizador JSON para convertir un texto JSON a un objeto JavaScript.

Un analizador JSON reconocerá únicamente el texto JSON y no compilará scripts.

En los navegadores que proporcionan apoyo JSON nativo, los analizadores JSON son también más rápidos.

Apoyo JSON nativo se incluye en todos los principales navegadores y en la última de ECMAScript (JavaScript) estándar:

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

Para navegadores mas viejos hay disponible una librería en <https://github.com/douglascrockford/JSON-js>.

JSON HTTP REQUEST

Como dijimos anteriormente un uso común de JSON es leer datos de un servidor web, y mostrar los datos en una página web.

Veremos, en 4 sencillos pasos, cómo leer datos JSON, utilizando XMLHttpRequest.

EJEMPLO JSON3.HTML

Este ejemplo lee un menú desde `myTutorials.txt`, y muestra el menú en una página web:

```
<div id="id01"></div>
```

```
<script>
```

```
var xmlhttp = new XMLHttpRequest();
```

```
var url = "myTutorials.txt";
```

```
xmlhttp.onreadystatechange = function() {
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



```
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {  
    var myArr = JSON.parse(xmlhttp.responseText);  
    myFunction(myArr);  
}  
  
xmlhttp.open("GET", url, true);  
xmlhttp.send();  
  
function myFunction(arr) {  
    var out = "";  
    var i;  
    for(i = 0; i < arr.length; i++) {  
        out += '<a href="' + arr[i].url + '">' +  
            arr[i].display + '</a><br>';  
    }  
    document.getElementById("id01").innerHTML = out;  
}  
</script>
```

Explicación del ejemplo

1 – Declaramos un array de objetos. Le damos a cada objeto un dos propiedades: display y url. Llamamos a este array myArray

```
var myArray = [  
    {  
        "display": "JavaScript Tutorial",  
        "url": http://www.w3schools.com/js/default.asp  
    },  
    {  
        "display": "HTML Tutorial",  
        "url": http://www.w3schools.com/html/default.asp  
    },  
    {  
        "display": "CSS Tutorial",  
        "url": http://www.w3schools.com/css/default.asp  
    }  
]
```

2- Creamos una función Javascript que recorre el array de objetos y muestra el contenido como vínculos HTML.

```
function myFunction(arr) {  
    var out = "";  
    var i;  
    for(i = 0; i < arr.length; i++) {
```



```

        out += '<a href="' + arr[i].url + '"' + '>' + arr[i].display + '</a><br>';
    }
    document.getElementById("id01").innerHTML = out;

}

```

Llamamos a la función pasándole como parámetro el array de objetos.

```
myFunction(myArray);
```

Ponemos el array en un archivo de texto myTutorials.txt

```

[
{
"display": "JavaScript Tutorial",
"url": http://www.w3schools.com/js/default.asp
},
{
"display": "HTML Tutorial",
"url": http://www.w3schools.com/html/default.asp
},
{
"display": "CSS Tutorial",
"url": http://www.w3schools.com/css/default.asp
}
]

```

4- Leemos el archivo de texto con un XMLHttpRequest y usamos myFunction(myArray) para mostrar el array

```

var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    var myArr = JSON.parse(xmlhttp.responseText);
    myFunction(myArr);
}
}

xmlhttp.open("GET", url, true);
xmlhttp.send();

```

JSON FUNCTION FILES

Ahora veremos el mismo ejemplo pero utilizando un archivo de funciones.

EJEMPLO JSON4.HTML

```
<div id="id01"></div>
```

```
<script>
```

```
function myFunction(arr) {
```

```
    var out = "";
```

```
    var i;
```

```
    for(i = 0; i<arr.length; i++) {
```

```
        out += '<a href="' + arr[i].url + '"' + arr[i].display + '</a><br>';
```

```
    }
```

```
    document.getElementById("id01").innerHTML = out;
```

```
}
```

```
</script>
```

```
<script src="myTutorials.js"></script>
```

Explicación del ejemplo

1 – El paso 1 es el mismo, creamos array de objetos. Le damos a cada objeto un dos propiedades: display y url. Llamamos a este array myArray

```
var myArray = [
```

```
{
```

```
    "display": "JavaScript Tutorial",
```

```
    "url": http://www.w3schools.com/js/default.asp
```

```
},
```

```
{
```

```
    "display": "HTML Tutorial",
```

```
    "url": http://www.w3schools.com/html/default.asp
```

```
},
```

```
{
```

```
    "display": "CSS Tutorial",
```

```
    "url": http://www.w3schools.com/css/default.asp
```

```
}
```

```
]
```

2- Creamos una función Javascript que recorre el array de objetos y muestra el contenido como vínculos HTML.

```
function myFunction(arr) {
```

```
    var out = "";
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

```
var i;  
for(i = 0; i < arr.length; i++) {  
    out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';  
}  
document.getElementById("id01").innerHTML = out;  
}
```

Llamamos a la función pasándole como parámetro el array de objetos, en este caso pasamos la variable en forma literal.

```
myFunction([  
    {  
        "display": "JavaScript Tutorial",  
        "url": "http://www.w3schools.com/js/default.asp"  
    },  
    {  
        "display": "HTML Tutorial",  
        "url": "http://www.w3schools.com/html/default.asp"  
    },  
    {  
        "display": "CSS Tutorial",  
        "url": "http://www.w3schools.com/css/default.asp"  
    }  
]);
```

El llamado a la función lo ponemos en un script externo llamado myTutorial.js e incluimos en nuestro html ese js externo de la siguiente manera:

```
<script src="myTutorials.js"></script>
```

EJEMPLO JSON5.HTML

El último ejemplo toma los datos de un script PHP que recupera la información en una tabla sql y genera un JSON. En el archivo Customers_MYSQLI.php es el script del lado del servidor que lee los datos de la tabla sql y arma un JSON. Los datos devueltos son procesados en json5.html

CREAR UN JSON DESDE PHP

Para crear una cadena para expresar un objeto u otro tipo de variable con JSON en PHP se dispone de una función llamada json_encode(), que recibe lo que deseamos convertir en notación JSON y devuelve una cadena de texto con el JSON producido.



Podemos convertir en JSON cualquier cosa que necesitemos, como una cadena, una variable numérica, un array -normal o asociativo- y objetos con todo tipo de datos dentro. Veremos varios ejemplos que ilustrarán este proceso.

```
$mivariable = "hola";  
print_r(json_encode($mivariable));
```

Esto devuelve:

```
"hola"
```

Ahora convertimos un array normal:

```
$miArray = array(1,4,6,8,3,34.8,9,43);  
print_r(json_encode($miArray));
```

Que devuelve:

```
[1,4,6,8,3,34.8,9,43]
```

Estos ejemplos anteriores son sin duda quizás demasiado simples, pero la cosa empieza a tener sentido cuando se convierte un array asociativo en JSON, como se puede ver a continuación.

```
$miArray = array("manzana"=>"verde", "uva"=>"Morada", "fresa"=>"roja");  
print_r(json_encode($miArray));
```

```
{"manzana":"verde","uva":"Morada","fresa":"roja"}
```

Este resultado ya tiene una forma parecida a lo que estamos acostumbrados a ver en JSON. Podríamos fijarnos que la cadena que devuelve no tiene saltos de línea e indentación de los elementos, con lo que la lectura se dificulta un poco. No obstante, podríamos utilizar esa cadena para crear una variable Javascript y por supuesto podremos acceder a cualquier valor del JSON, de una manera parecida a esta:

```
<script>  
  
var obj = '<?php echo json_encode($miArray);?>';  
  
JSONFrutas = JSON.parse(obj);  
  
alert(JSONFrutas.manzana);  
  
</script>
```

Luego de utilizar JSON.parse podemos acceder a los componentes del JSON como lo hacemos con cualquier objeto, variable, operador punto y luego el nombre de la

propiedad a acceder.

El código antes descripto se encuentra en el script json6.php dentro de la carpeta JSON.

CONSEJOS PARA CREAR UN JSON CORRECTAMENTE

La función `json_decode()` de PHP requiere unas normas un poco más estrictas a la hora de construir un JSON. Si no atendemos esas reglas no obtendremos un mensaje de error, sino que la función devolverá un valor nulo: `null`. Por tanto, algo que Javascript puede interpretar correctamente puede que no sea así por PHP, así que hay que prestar especial atención sobre una serie de puntos.

1) En la cadena JSON, tanto el nombre de una propiedad como el valor deben estar entre comillas, menos los valores numéricos, o palabras como `null`, que pueden estar sin comillas.

Incorrecto:

```
$json = json_decode('{nombre: "cualquier valor"}');
```

Correcto:

```
$json = json_decode('{"nombre": "cualquier valor"}');
```

2) Se tienen que utilizar comillas dobles. No se permiten comillas simples para los nombres de variables y sus valores.

Incorrecto:

```
$json = json_decode('{frutas: ['pera', 'uva']}');
```

Correcto:

```
$json = json_decode('{"frutas": ["pera", "uva"]}');
```

3) No podemos dejarnos una coma que después no tenga nada, así como un corchete o llave de más o de menos.

Incorrecto:

```
$json = json_decode('{"nombre": "valor", }');
```

Correcto:

```
$json = json_decode('{"nombre": "valor"}');
```

4) Tenemos que enviar los datos a `json_decode()` en UTF-8. Si estamos trabajando en iso-8859-1, habría que codificar la cadena en UTF-8 previamente:

Incorrecto:

```
$json = json_decode('{"mañana": "miércoles"}');
```

Correcto:

```
$json = json_decode(utf8_encode('{"mañana": "miércoles"}'));
```

CREAR OBJETOS PHP A PARTIR DE UNA CADENA JSON

A continuación veremos cómo crear variables u objetos PHP a partir del contenido de cadenas JSON.

PHP puede interpretar datos provenientes de una cadena con notación JSON, de modo que se puedan guardar en variables y luego utilizarlas en los scripts del lado del servidor. En este artículo veremos cómo hacerlo y además destacaremos una serie de consejos para que los JSON estén bien formados y consigamos una correcta interpretación.

Primero vamos a mostrar un objeto definido con JSON mediante Javascript, para hacer unas comprobaciones y saber si está bien construido.

```
<script>
var objJson = {
  nombre: "Juan",
  idCliente: 22,
  observaciones: null,
  producto: {
    id: 5,
    nombre: "Smartphone LG"
  }
}
alert(objJson.nombre)
alert(objJson.producto.nombre)
</script>
```

Este script Javascript, ejecutado en un navegador, nos mostraría dos mensajes de alerta con valores que se han colocado en el objeto definido con notación JSON. Esto no tiene nada que ver con lo que vamos a ver sobre PHP, pero al menos tenemos claro que nuestro JSON es correcto.

Ahora vamos a crear una variable PHP con la cadena para hacer este objeto y lo cargaremos en una variable PHP interpretando el JSON. Para ello PHP dispone de una función llamada `json_decode()` que recibe la cadena con notación JSON y devuelve un objeto, o cualquier otro tipo de variable, que estuviera representada en el JSON.

```
<?php
$str_obj_json = '{
  "nombre": "Juan",
```

```
"idCliente": 22,  
"observaciones": null,  
"producto": {  
    "id": 5,  
    "nombre": "smatphone LG"  
}  
};  
$obj_php = json_decode($str_obj_json);  
?>
```

Ahora podríamos mostrar el contenido de esa variable con la función de PHP `print_r()`, que muestra el contenido de variables de manera que se pueda entender por un humano.

```
stdClass Object  
(  
    [nombre] => Juan  
    [idCliente] => 22  
    [observaciones] =>  
    [producto] => stdClass Object  
    (  
        [id] => 5  
        [nombre] => Smartphone LG  
    )  
)
```

Por último podemos intentar acceder a una de las propiedades de este objeto, de esta manera:

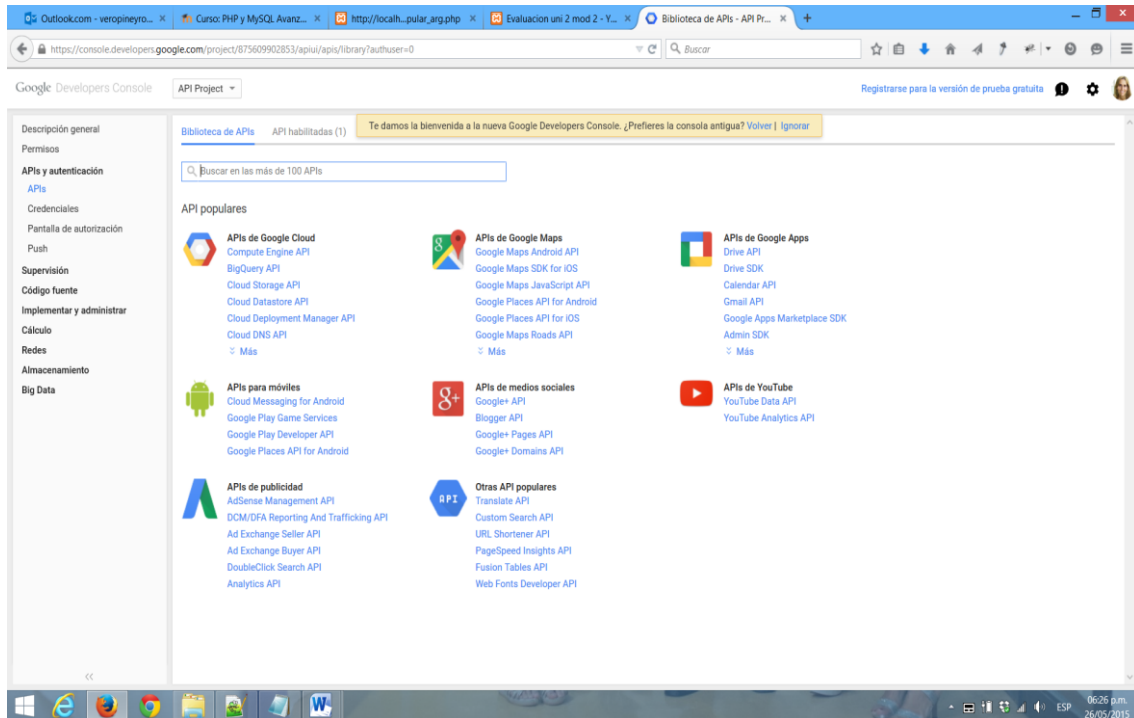
```
echo "Una propiedad cualquiera:" . $obj_php->nombre;
```

El código de este ejemplo se encuentra en `json7.php` dentro de la carpeta JSON.

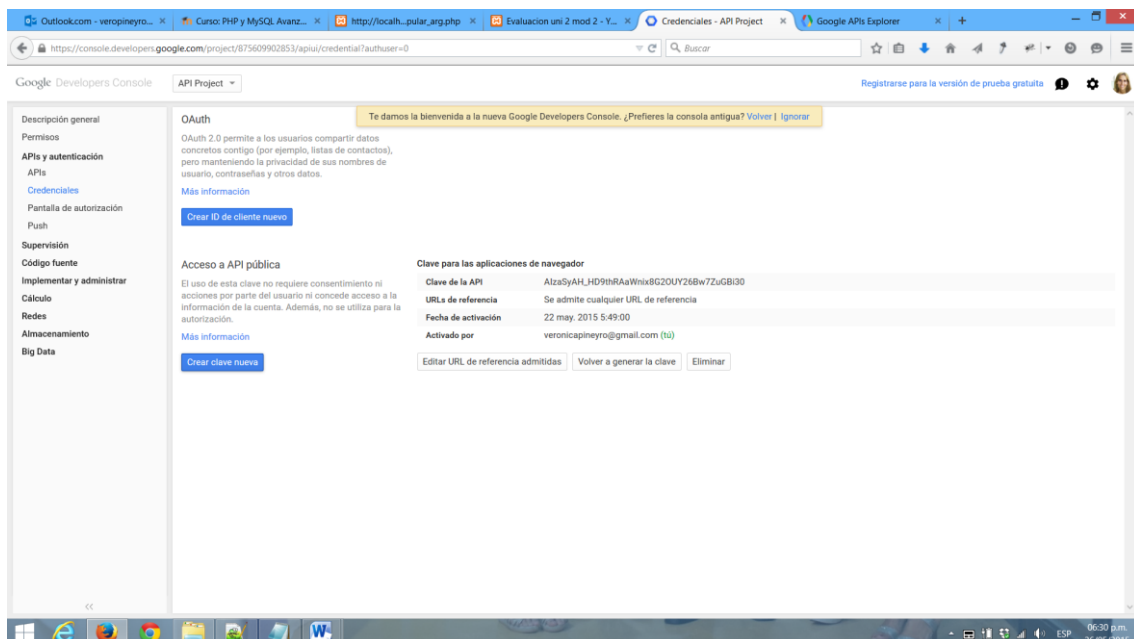
EJERCICIO PRÁCTICO USO DE LA API V3 DE YOUTUBE

La V2 de la API de Youtube estaba desarrollada en formato XML, la misma fue discontinuada y a partir de ahora Youtube nos obliga a trabajar con la V3 de la API que utiliza JSON como formato de intercambio de información.

Para poder utilizar la V3 de la API debemos obtener una API Key en la consola de Google. Para ello debemos ir a <https://console.developers.google.com/> una vez que ingresamos con nuestro usuario de Google vamos a ver una pantalla como esta:



Habilitamos la Youtube Data API. Y vamos a la opción Credenciales de nuestro menú de la izquierda en la siguiente pantalla es donde debemos crear nuestra API Key para poder usar la API de Youtube V3



Pueden seguir estos pasos:

www.youtube.com/watch?v=JbWnRhHfTDA

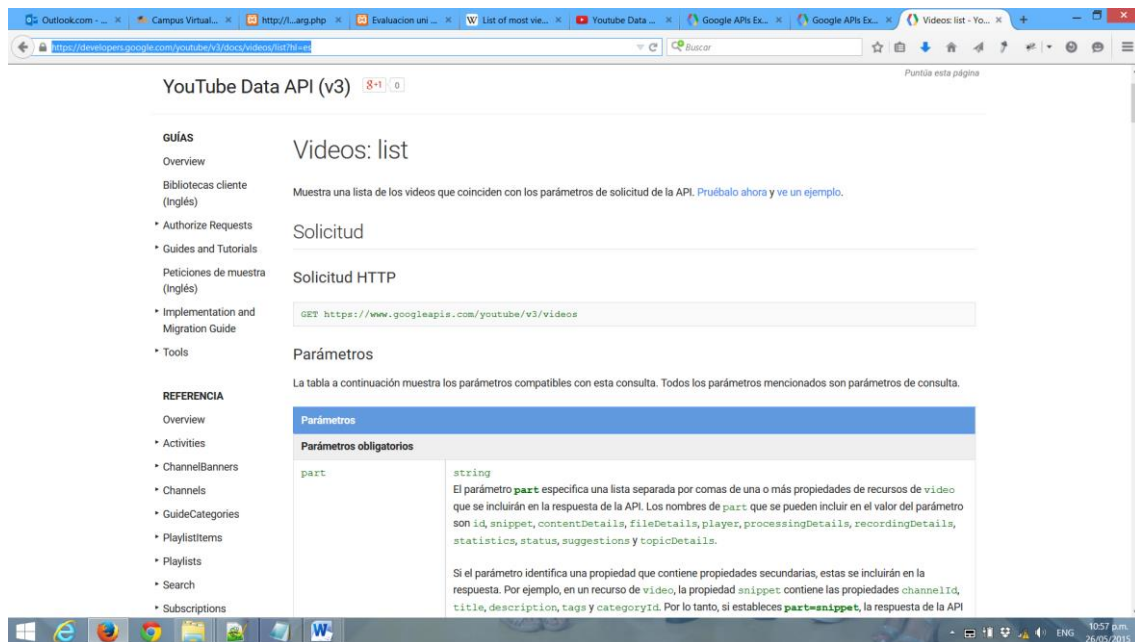
Una vez que tenemos nuestra API Key estamos en condiciones de pedirle los datos que necesitamos. Para pedirle esos datos tenemos que setear una serie de parámetros. Aquí tenemos la lista de videos con la explicación de cada uno de los parámetros:

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

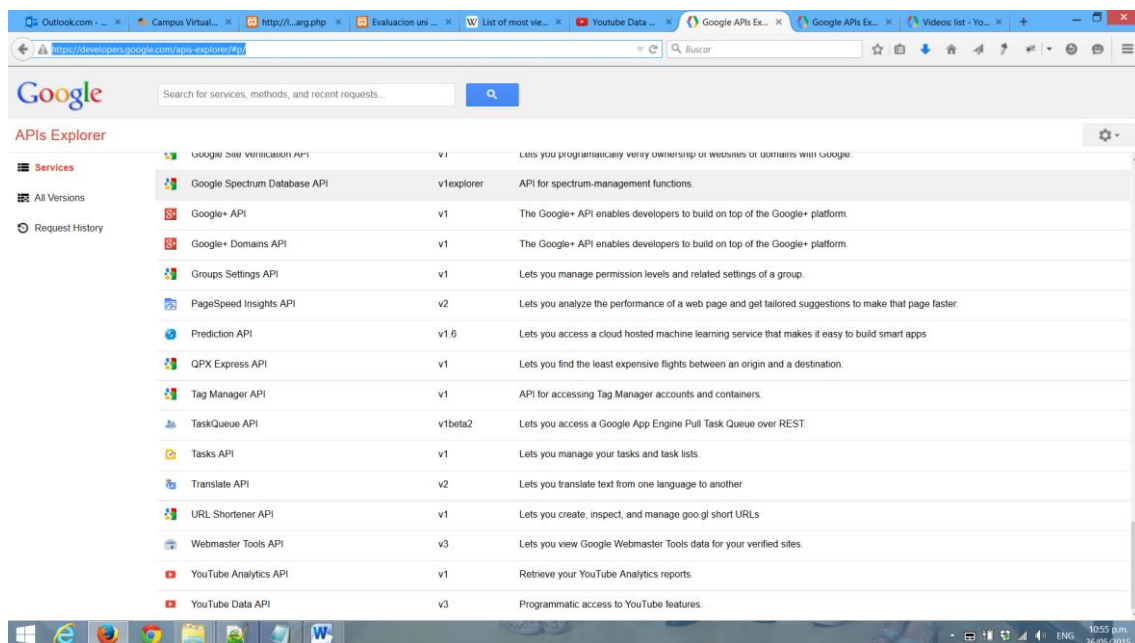
UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

<https://developers.google.com/youtube/v3/docs/videos/list?hl=es>



Nos vamos al explorador de las APIs de Google, <https://developers.google.com/apis-explorer/#p/>



Y seleccionamos Youtube Data API.

De la lista que se nos muestra a continuación vamos a seleccionar youtube.videos.list que contiene la lista de videos que nos será devuelta por la API según los parámetros que seteemos.

Una vez que ingresamos nos muestra unos campos de input que tenemos que completar según los datos que queramos obtener. Por ejemplo vamos a completar:

Part: contentDetails

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar

Chart: mostPopular
maxResults: 10
regionCode: AR

Una vez que ingresamos los presionamos el botón celeste al final de los campos que dice Execute. Eso nos va a generar la url que acabamos de solicitar según los parámetros establecidos y la respuesta de la API:

Request:

GET
https://www.googleapis.com/youtube/v3/videos?part=contentDetails&chart=mostPopular&maxResults=10®ionCode=AR&key={YOUR_API_KEY}

Response:

```
{
  "kind": "youtube#videoListResponse",
  "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/wabCAMEfCQ0yeolCBE2sdmAQygM\"",
  "nextPageToken": "CAoQAA",
  "pageInfo": {
    "totalResults": 30,
    "resultsPerPage": 10
  },
  "items": [
    {
      "kind": "youtube#video",
      "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/jzo6btnInbIqt6NviETJsQHpxPM\"",
      "id": "Hcyn4EBJcrc",
      "contentDetails": {
        "duration": "PT9M39S",
        "dimension": "2d",
        "definition": "hd",
        "caption": "false",
        "licensedContent": true
      }
    },
    {
      "kind": "youtube#video",
      "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/4SJH3aNbIsIVvwunR3h16QqlxdA\"",
      "id": "QKC_Ga_XRak",
      "contentDetails": {
        "duration": "PT5M14S",
        "dimension": "2d",
        "definition": "hd",
        "caption": "false",
        "licensedContent": true
      }
    },
    {
      "kind": "youtube#video",
      "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/IHDH6lCfO3JASKxeN6DNNu97UH8\"",
      "id":
```

Centro de Formación, Investigación y Desarrollo de Soluciones de e-Learning.

UTN - FRBA. Secretaría de Cultura y Extensión Universitaria

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148 // e-learning@sceu.frba.utn.edu.ar



```
"ZA1kRoCGeBI",
  "contentDetails": {
    "duration": "PT7M49S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": true
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/9mRBCYgm2ojmPQjmMNI4ZV8Rsz8\"",
  "id": "k3FXT1fOmpc",
  "contentDetails": {
    "duration": "PT13M57S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": true
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/v3iY8Dypx5MHMGsYEO721AFJBtU\"",
  "id": "QcIy9NiNbmo",
  "contentDetails": {
    "duration": "PT4M5S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": true
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/7fjVZ8cWiLBii7UOMX8V6x1S7eA\"",
  "id": "jdOfzucSkzE",
  "contentDetails": {
    "duration": "PT8M52S",
    "dimension": "2d",
    "definition": "sd",
    "caption": "false",
    "licensedContent": false
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuD438sF4hb-VcBBREXc/LQN-06je5tqIncBii92ldJznmvc\"",
  "id": "QK2FM2iyM_M",
  "contentDetails": {
    "duration": "PT11M45S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": true
  }
},
},
```



```
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuZD438sF4hb-
VcBBREXc/OK5oArMqT2mGBzM83T7w0UGpEiU\"",
  "id": "ceJod3fXU1M",
  "contentDetails": {
    "duration": "PT29M24S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": true
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuZD438sF4hb-
VcBBREXc/SnW0IyYUI4usNFNYWgPzWRytG9A\"",
  "id": "2OrnISO_lhc",
  "contentDetails": {
    "duration": "PT1M18S",
    "dimension": "2d",
    "definition": "hd",
    "caption": "false",
    "licensedContent": false
  }
},
{
  "kind": "youtube#video",
  "etag": "\"xmg9xJZuZD438sF4hb-
VcBBREXc/RbcY57R50fj79us8I7znOjHPInQ\"",
  "id": "Ar08IfAjb4",
  "contentDetails": {
    "duration": "PT46S",
    "dimension": "2d",
    "definition": "sd",
    "caption": "false",
    "licensedContent": false
  }
}
]
```

Estos son los valores devueltos por la API, que como se puede ver es un objeto JSON.

Ese request lo podemos hacer desde un PHP que obtenga la respuesta de la API de Youtube y que utilice esa información para mostrarla en nuestra pagina.

En la carpeta yt_v3_json_ tienen el acceso a la url generada desde PHP para explorar su contenido y la impresión de la respuesta devuelta por la API.

Si quieren seguir investigando como imprimir la lista de videos de un determinado canal les dejo el siguiente video de GoogleDevelopers que los puede orientar en como hacerlo. <https://www.youtube.com/watch?v=RjUImco7v2M>