Desarrollo Web en Entorno Servidor

Programación en PHP

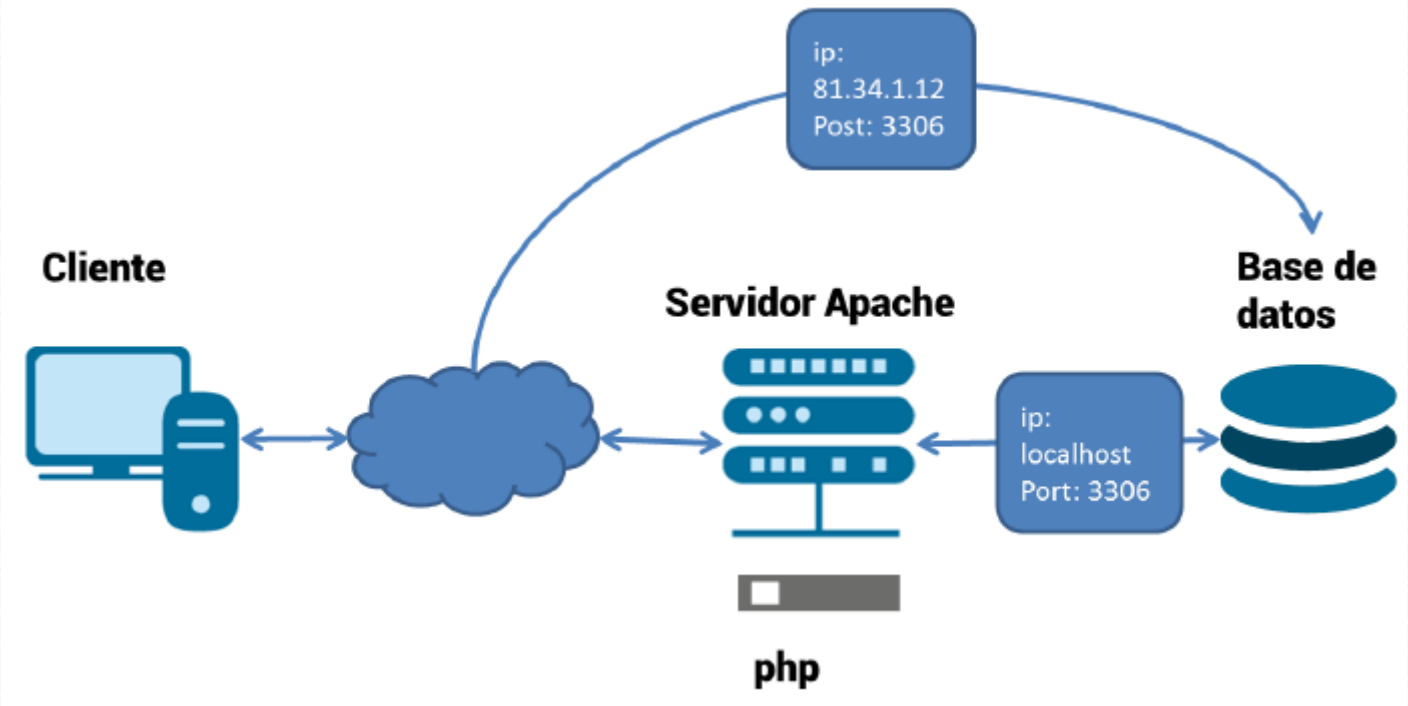
José A. Lara

API con PHP. JSON

- Del inglés Interfaz de Programación de Aplicaciones, el API de una aplicación es un **conjunto** de clases, algoritmos y funcionalidades que nos proporcionan principalmente un acceso a información, en bases de datos normalmente, controlando y gestionando los permisos, accesos y niveles de seguridad para un pull de clientes.
- Si lo aterrizamos en el entorno de programación, resulta que es una aplicación que nos va a permitir realizar una serie de acciones y de actividades de una forma sencilla y transparente para el usuario o máquina externa que utiliza ese API.

API con PHP. JSON

- Recordemos cómo accedíamos a una base de datos MySQL:



API con PHP. JSON

- El cliente accedía por dos mecanismos a la información de la base de datos, bien a través del servidor Apache y de una aplicación realizada, en nuestro caso con PHP, o bien directamente.
- ¿Qué ocurre si el cliente no es un ser humano, sino un móvil o cualquier otro dispositivo automático?
- ¿Qué ocurre si quien accede no necesita disponer de una visualización de datos en formato HTML, es más en ese formato no sabría interpretar la información?

API con PHP. JSON

- Podemos dar como respuesta que podríamos utilizar el mecanismo de la conexión directa con la base de datos, sin embargo ese mecanismo tiene dos problemas:
  - Es muy **dependiente del tipo de la base de datos** y por lo tanto no estándar necesitando un conector para cualquier transacción. Por ejemplo en dispositivos móviles no es una alternativa la utilización de conectores para el acceso a las bases de datos.
  - El acceso a la base de datos de forma directa es **muy peligroso** a nivel de seguridad ya que deja el acceso directo a la base de datos.

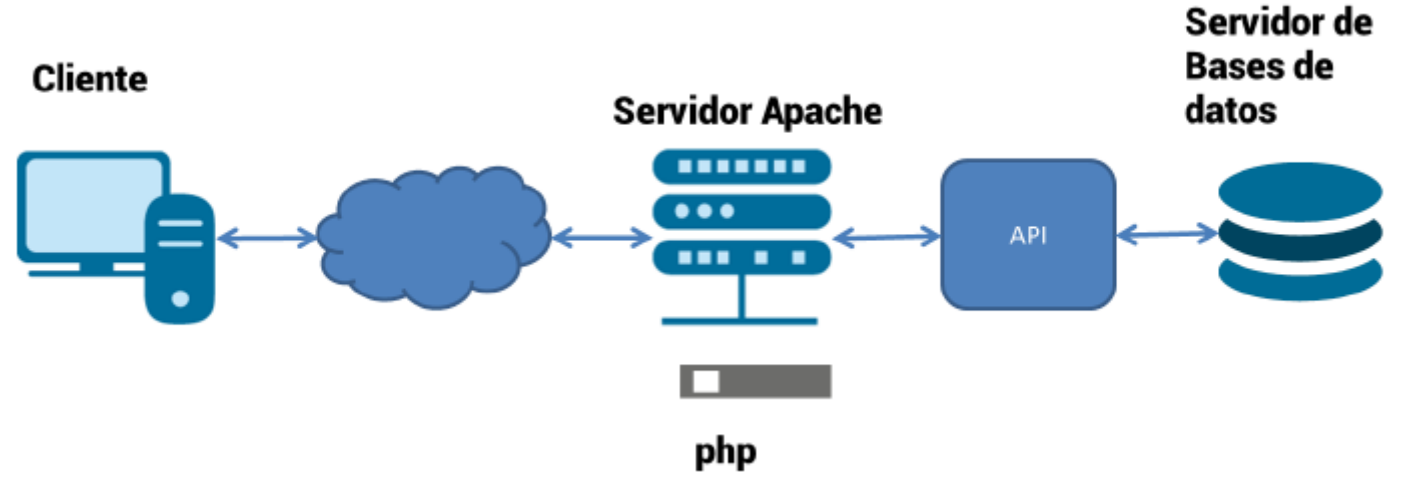
API con PHP. JSON

API de datos

- La respuesta a las anteriores preguntas es crear un API, una app intermedia que acceda a los datos de una base de datos y que devuelva la información. Esto da solución a ambas problemáticas:
  - El API utiliza un mecanismo estándar de comunicación como es el HTTP y por lo tanto no necesita de conectores privador, es el API el encargado de solucionar esta problemática
  - El API se encarga de la seguridad

API con PHP. JSON

API de datos



API con PHP. JSON

Construcción de un API sencillo

- Vamos a construir un sencillo API, puesto que ya tenemos todas las herramientas y conocimientos para poder realizar un desarrollo que acceda a una serie de datos en unas tablas de la base de datos y dependiendo del método utilizado, devuelva información o bien interactúe con las tablas.

API con PHP. JSON

Construcción de un API sencillo. Paso 1 - MVC

- El primer paso será plantear la estructura siguiendo el Patrón MVC, así si lo que queremos es interactuar contra la tabla usuarios, deberemos crear la siguiente estructura:

```
modelo
|_____ db.php
       usuario.php
```

API con PHP. JSON

Construcción de un API sencillo. Paso 1 - MVC

- Si recordamos lo visto en la UD8:
  - El fichero db.php será el encargado de la conexión contra la base de datos.
  - El fichero usuario.php será el encargado de realizar todas las transacciones de información.

API con PHP. JSON

```php
/** * Permitir la conexion contra la base de datos */
class db {
//Atributos necesarios para la conexion
    private $host="localhost";
    private $user="root";
    private $pass="";
    private $db_name="usuarios";
    //Conector
    private $conexion;
    //Propiedades para controlar errores
    private $error=false;
    private $error_msj="";
    function __construct() {
        $this->conexion = new mysqli($this->host, $this->user,
            $this->pass, $this->db_name);
        if ($this->conexion->connect_errno) {
            $this->error=true; $this->error_msj="No se ha podido
            realizar la conexion a la bd. Revisar base de datos
            o parámetros";
        }
    }
```

```php
    //Funcion para saber si hay error en la conexion
    function hayError(){
        return $this->error;
    }
    //Funcion que devuelve mensaje de error
    function msjError(){
        return $this->error_msj;
    }
    //Metodo para la realización de consultas a la bd
    public function realizarConsulta($consulta){
        if($this->error==false){
            $resultado = $this->conexion->query($consulta);
            return $resultado;
        }else{
            $this->error_msj="Imposible realizar la consulta: ".$consulta;
            return null;
        }
    }
}
```

API con PHP. JSON

```php
include "db.php";
/** * */
class Usuario extends db {
    function __construct() {
    //De esta forma realizamos la conexion a la base de datos
        parent::__construct();
    }
    //Devolvemos todos los usuarios
    function devolverUsuarios(){
    //Construimos la consulta
        $sql="SELECT * from usuario";
        //Realizamos la consulta
        $resultado=$this->realizarConsulta($sql);
        if($resultado!=null){ //Montamos la tabla de resultados
            $tabla=[];
            while($fila=$resultado->fetch_assoc()){
                $tabla[]=$fila;
            }
            return $tabla;
        }else{
            return null;
        }
    }
}
```

API con PHP. JSON

Ejercicio

- Con lo visto y aprendido en Unidades anteriores y la actual podemos crear un API para una aplicación de un restaurante:
  1. Crearemos una estructura de directorios y ficheros idéntica a la propuesta en la teoría:
     a. Fichero padre db.php encargado de la conexión
     b. Fichero hijo restaurante.php encargado de la interacción con una base de datos
  2. Generaremos el contenido del fichero db.php de acuerdo al ejemplo anterior

API con PHP. JSON

CRUD

- Deberemos crear el CRUD completo contra la base de datos y tablas contra las que queremos realizar la gestión de información. Esto lo realizaremos dentro del fichero usuario.php descrito en el apartado anterior.

API con PHP. JSON

CRUD

- Sería por lo tanto en este punto en el que desarrollaríamos todas las funciones necesarias para interactuar con nuestra base de datos:
  - Funciones de lectura, tantas como necesidad tengamos en la aplicación. Como mínimo deberíamos tener tantas como tablas tengamos en nuestra aplicación.
  - Funciones de escritura, para realizar la inserción de información en las tablas
  - Funciones de actualización, para realizar la actualización de la información
  - Funciones de borrado de información

API con PHP. JSON

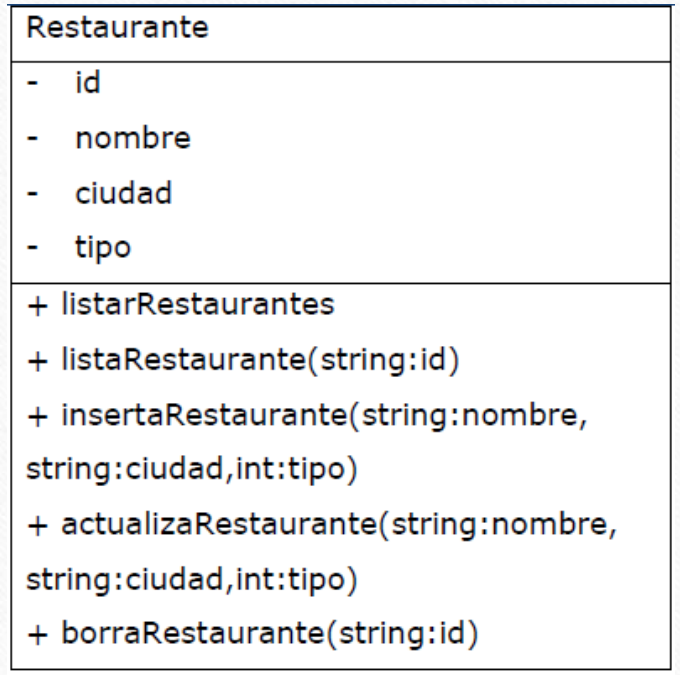
CRUD

- En nuestros ejemplos no nos vamos a preocupar de la seguridad de nuestro API, pero ya podemos imaginar que uno de los problemas a atender más importante dentro del desarrollo del API es justamente limitar y/o permitir con unos determinados niveles de seguridad a la información justa.

API con PHP. JSON

Ejercicio

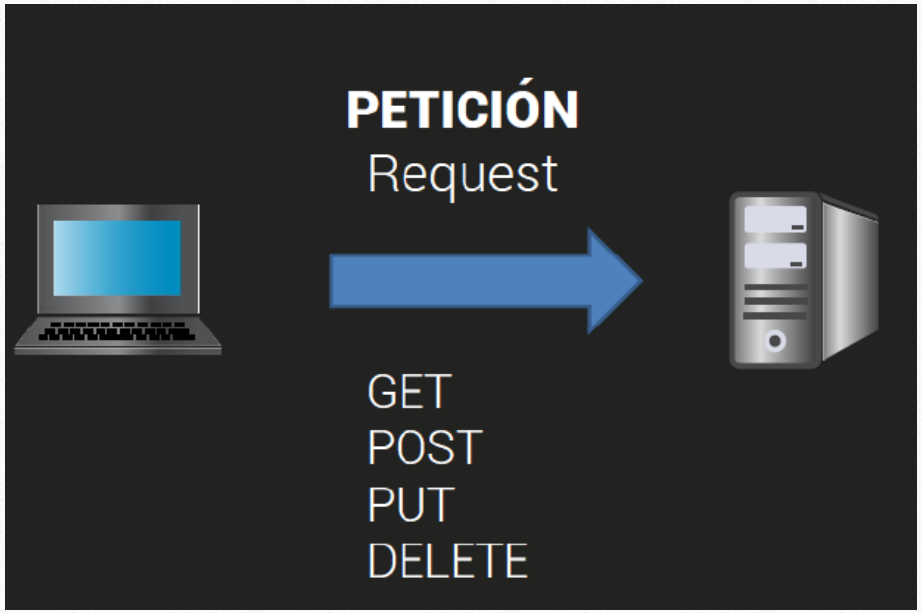
- Siguiendo con el ejemplo anterior:
  1. Crearemos la tabla de la base de datos con los siguientes campos
  2. Generaremos las funciones dentro de restaurante.php para poder acceder a la base de datos
  3. Crearemos un fichero de prueba, debugRestaurante.php para realizar las diferentes pruebas contra el API generado



API con PHP. JSON

API. Request METHOD

- Cuando vimos Formularios vimos los diferentes tipos de métodos request que podíamos utilizar:



API con PHP. JSON

API. Request METHOD

- En ese momento nos centrábamos únicamente en el uso de Formularios para la interacción con nuestras aplicaciones, pero en el momento que deja de ser un usuario final quien interactúa con nuestra aplicación y pasa a ser dispositivos que no pueden utilizar formularios, comienza a tener sentido el uso de los diferentes métodos de petición para poder discernir qué acción vamos a realizar con nuestra base de datos:
  - Con el método GET realizaremos un SELECT de la base de datos
  - Con el método POST realizaremos un INSERT
  - Con el método DELETE realizaremos un DELETE
  - Con el método PUT realizaremos un UPDATE

API con PHP. JSON

API. Request METHOD

- A través de la superglobal $_SERVER podemos conocer cuál es el método

en la petición por parte del cliente, el código es muy sencillo.

- Según la documentación de php.net:

'REQUEST_METHOD'
Which request method was used to access the page; i.e. 'GET', 'HEAD', 'POST', 'PUT'. `$_SERVER['REQUEST_METHOD']`

API con PHP. JSON

JSON

- No es la intención ni la misión de este curso entrar en detalle del uso de JSON, pero para acabar de formar nuestro API, es imprescindible conocer generalidades del mismo.
- JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

API con PHP. JSON

JSON

- Se ha convertido en un estándar para la comunicación de información entre procesos, y sobre todo en la utilización de APIs. Veamos un ejemplo:

```
{
    "glossary":
    {
        "title": "example glossary",
        "GlossDiv":
        {
            "title": "S",
            "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                    "SortAs": "SGML",
                    "GlossTerm": "Standard Generalized Markup Language",
                    "Acronym": "SGML",
                    "Abbrev": "ISO 8879:1986",
                    "GlossDef": {
                        "para": "A meta-markup language, used to create
                        markup languages such as DocBook.",
                        "GlossSeeAlso": ["GML", "XML"]
                    },
                "GlossSee": "markup"
                }
            }
        }
    }
}
```

API con PHP. JSON

JSON

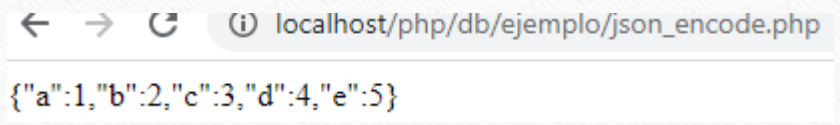
- Como podemos observar a diferencia con XML u otros Lenguajes de Marcas, el nivel de "burocracia" es mínima, ya que consiste en una estructura jerárquica de datos.

API con PHP. JSON

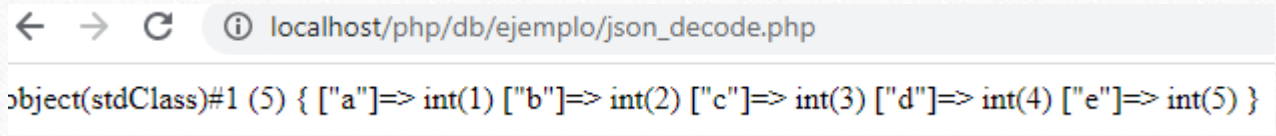
JSON

- Cómo codificar y decodificar json con php va a ser bien sencillo también, ya que a partir de arrays asociativos podemos generar una cadena json fácilmente:

```php
<?php
    $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
    echo json_encode($arr);
?>
```




```php
<?php
    $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
    var_dump(json_decode($json));
?>
```



API con PHP. JSON

Estructura final

- Una vez visto tanto el modelo como la vista (en este caso la salida es un JSON), la estructura final quedaría de la siguiente forma:

```
modelo
|_________   db.php
             restaurante.php
api.php
```

API con PHP. JSON

Estructura final

- Donde api.php realizaría las siguientes labores:
  - Recibe las peticiones por parte del cliente
  - Distingue entre los métodos solicitados
  - Recoge los datos si son necesarios a través de la superglobal $_REQUEST
  - Realiza la llamada al método solicitado
  - Devuelve la información a través de json_encode

  En este caso la salida de datos no se realiza embebiendo el echo dentro de etiquetas <html> sino directamente con un echo del json_encode

API con PHP. JSON

Estructura final

```php
<?php
    include "./modelo/restaurante.php";
    //distinguimos el tipo de peticion
    $requestMode=$_SERVER['REQUEST_METHOD'];
    if($requestMode=="GET"){
        ...
        echo json_encode($result);
    } else if ($requestMode=="POST") {
        ...
        echo json_encode($result);
    } else if ($requestMode=="PUT"){
        ...
        echo json_encode($result);
    } else if ($requestMode=="DELETE"){
        ...
        echo json_encode($result);
    } else {
        echo json_encode(["resultado"=>"Fallo"]);
    }
?>
```

API con PHP. JSON

Ejercicio

- Siguiendo con el ejemplo anterior:
  1. Crearemos un nuevo fichero, api.php, que será el encargado de recibir y devolver la información solicitada.
  2. En dicho fichero crearemos una estructura sencilla con un switch o if y dependiendo del método usado, utilizaremos la función creada en restaurante.php
  3. Incluiremos el código necesario para poder recibir/devolver la información con JSON