

Programación Server-Side

Introducción a PHP

Arquitectura Cliente - Servidor

Las aplicaciones web funcionan sobre un esquema **cliente-servidor**.

En este tipo de interacción, el usuario (**cliente**) realiza **peticiones (request)** a un programa remoto (**servidor**), quien le devolverá a cambio una **respuesta (response)**.

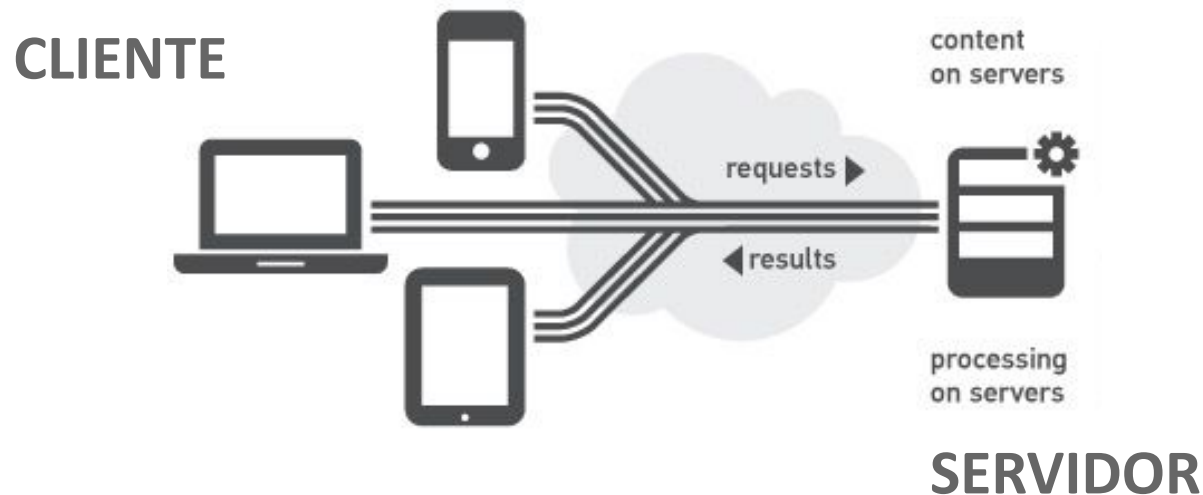
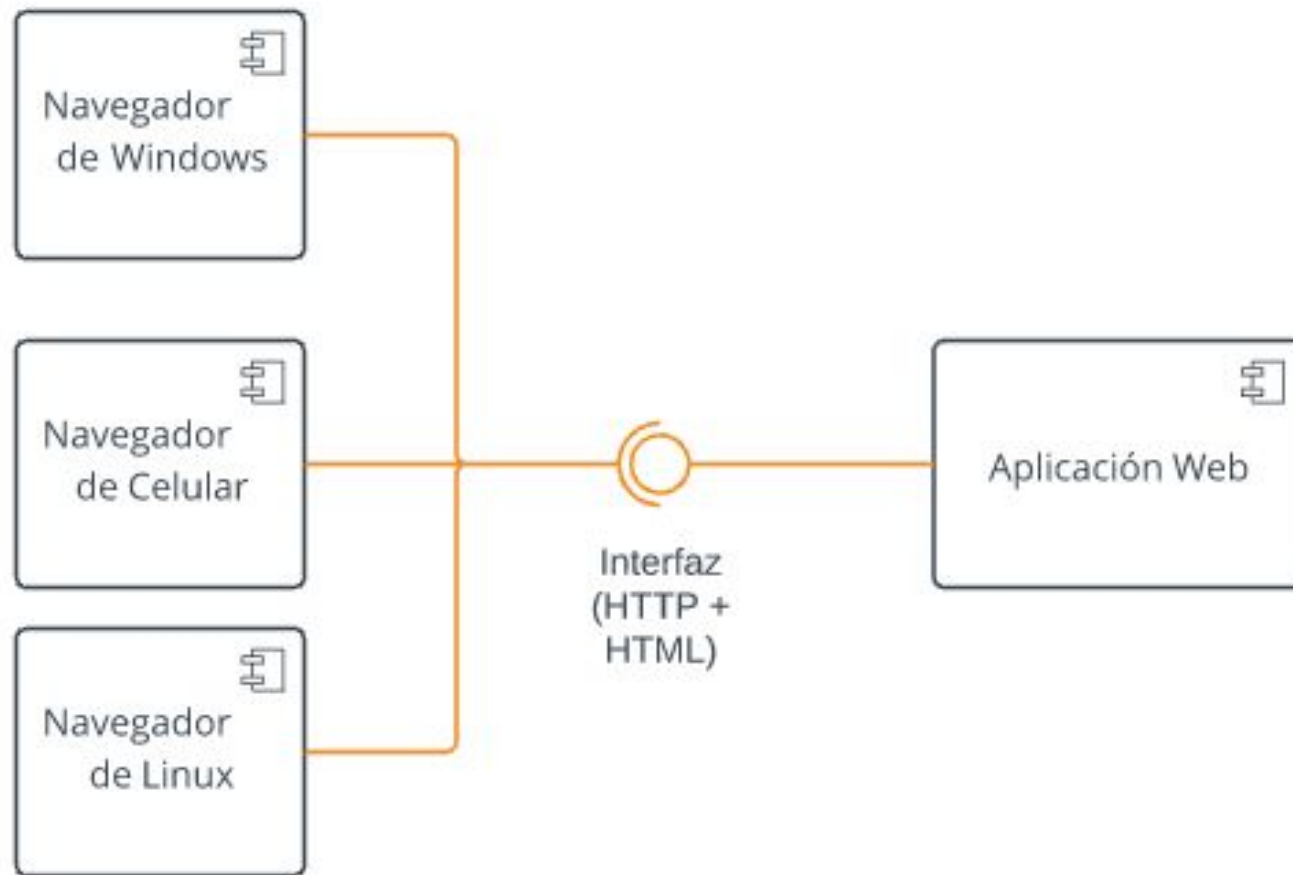


Diagrama de componentes

REPASO



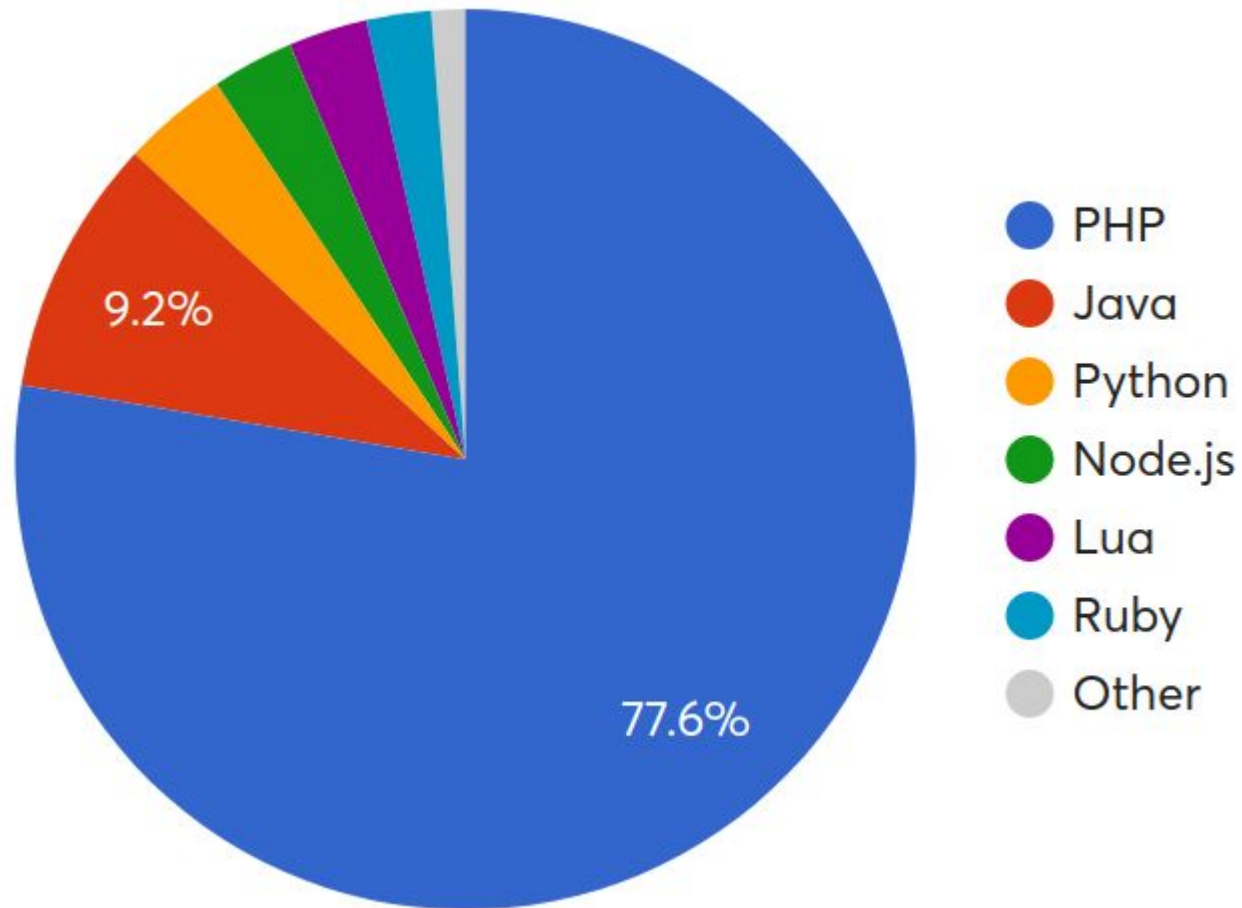
Programación Server-Side

La programación **server-side** es el componente principal de los *sitios dinámicos*, donde un servidor recibe las solicitudes de los clientes, para luego procesarlas y devolver una respuesta.

- El cliente recibe HTML/JSON pero no sabe cómo fue producido.
- Requiere instalación en el servidor.
- No requiere instalación ni complementos adicionales en el cliente.



Lenguajes server-side (Market Leaders)



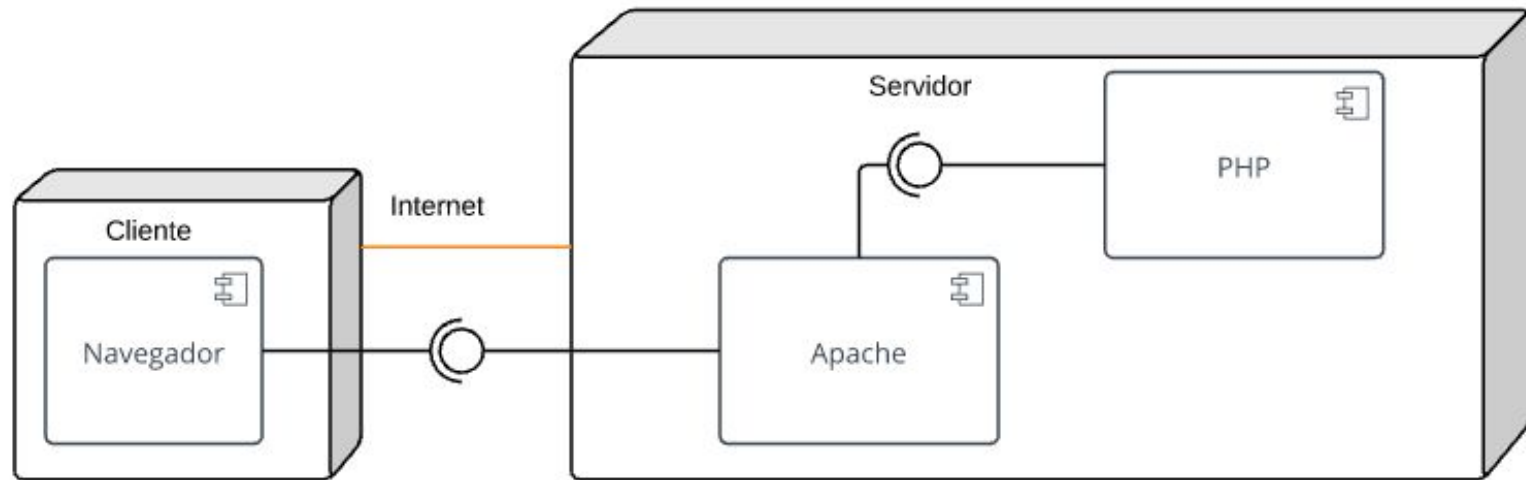
PHP

PHP (Hypertext Preprocessor)

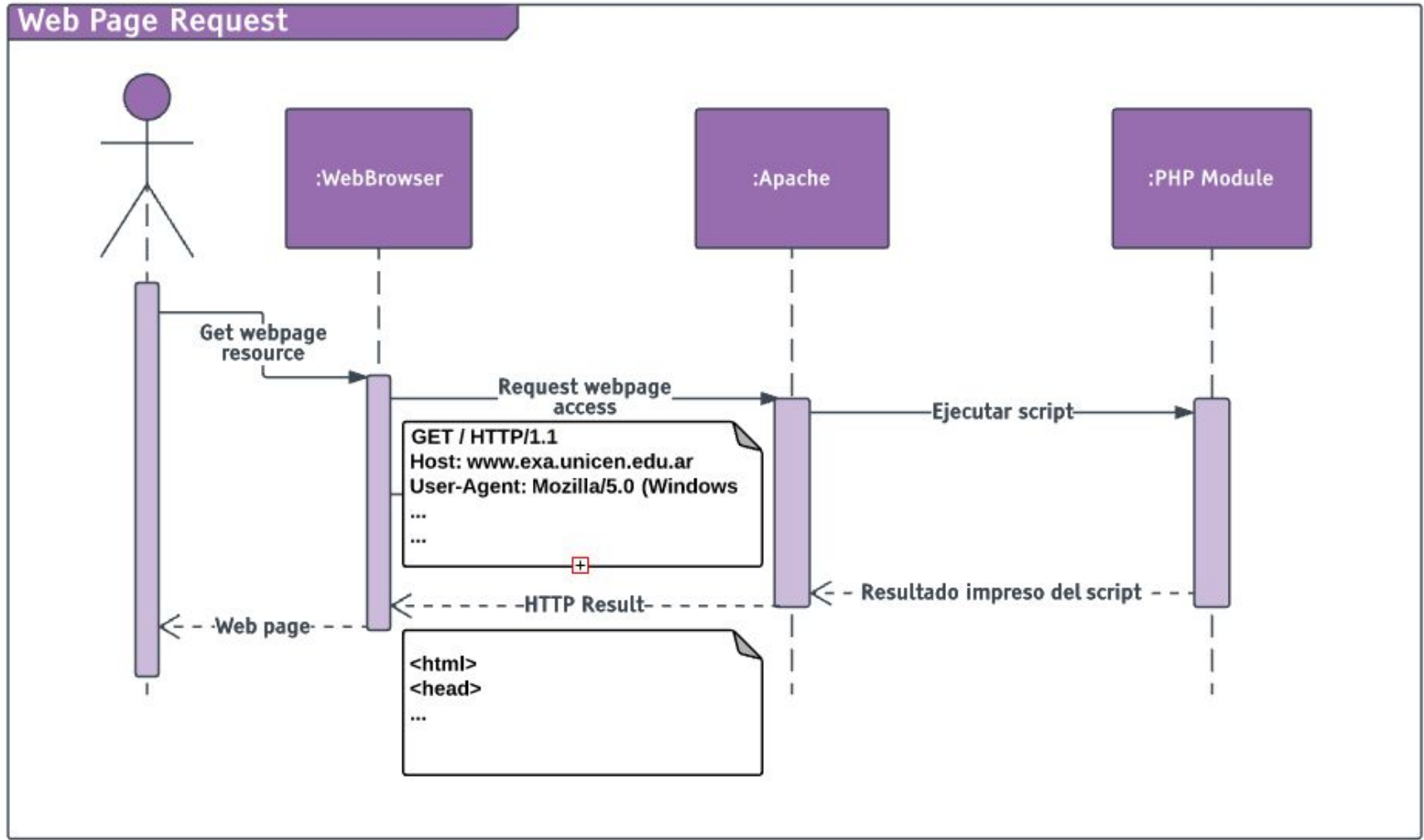
Lenguaje server-side de código abierto muy popular especialmente adecuado para el desarrollo web.

- Lenguaje de programación interpretado.
- Diseñado para producir sitios y aplicaciones web dinámicas.
- Soporte para múltiples plataformas.
- El código es procesado por el intérprete PHP que genera la página web resultante.

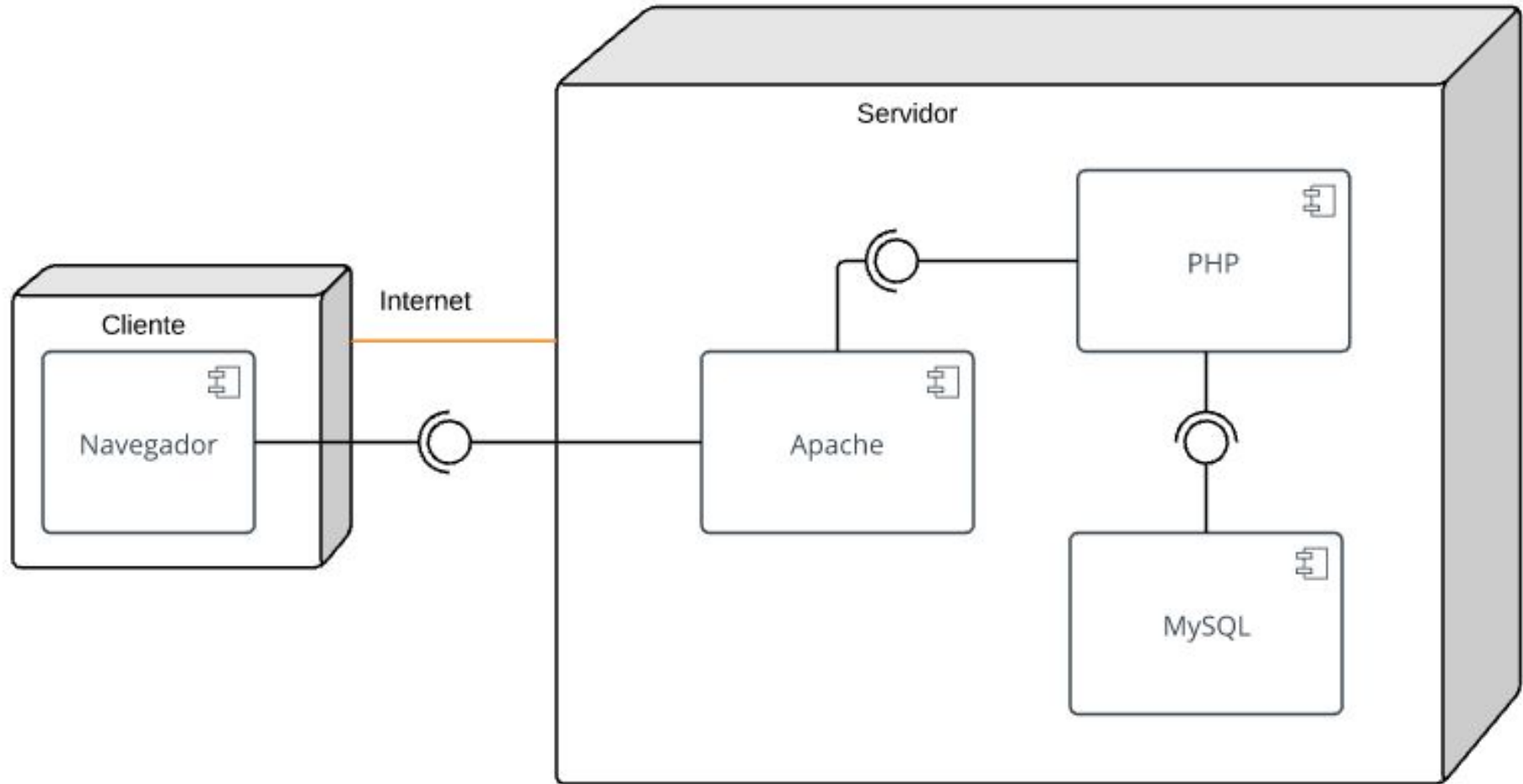
Ejecución de código PHP



Ejecución de código PHP

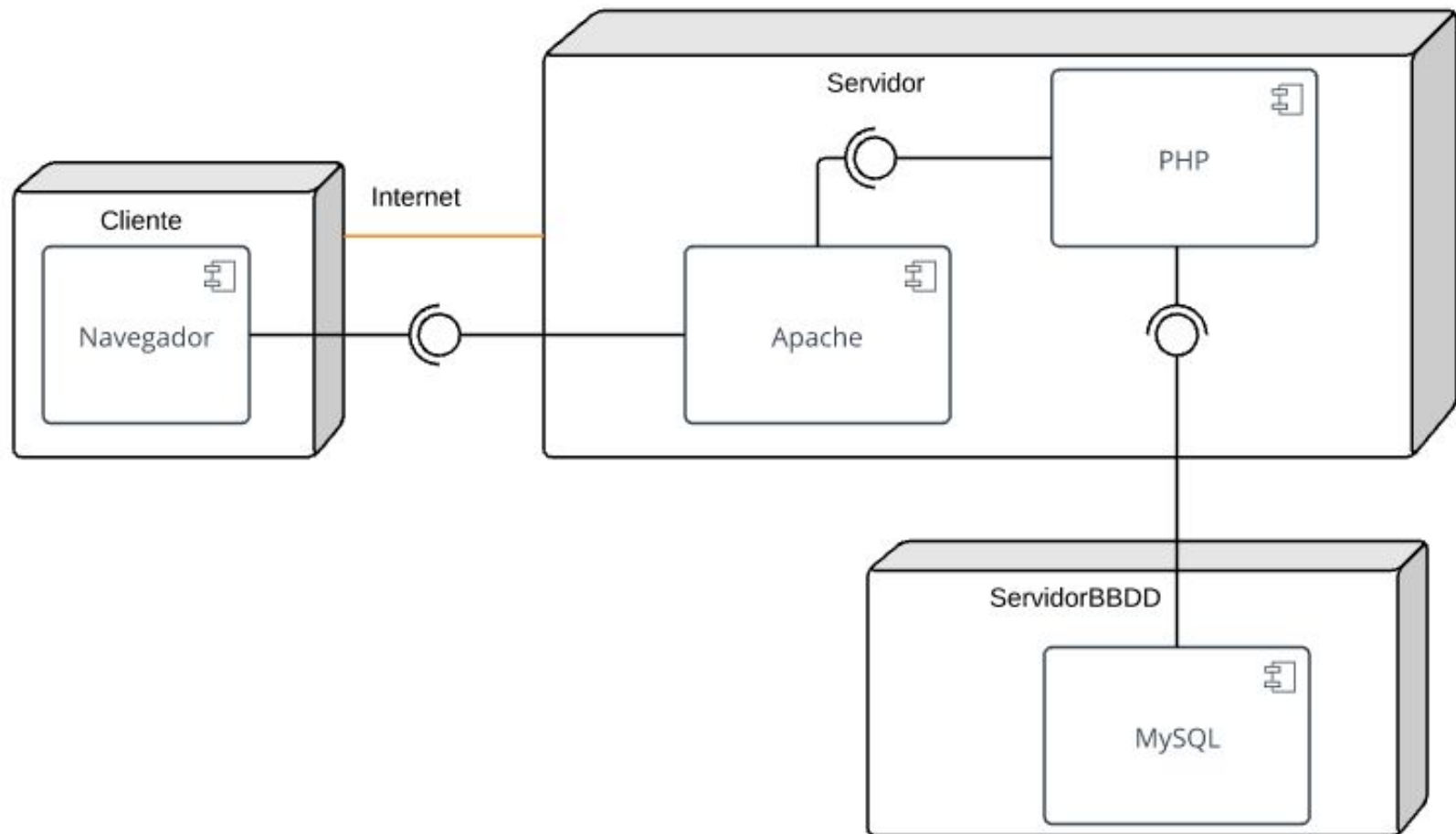


Más adelante habrá una BBDD



Diferentes despliegues

El servidor podría estar desplegado (deployado) en más de un servidor. Por ejemplo, la BBDD en otra máquina.



PHP: ¿Como se empieza?

CREAR UN ARCHIVO PHP

.php

EL ARCHIVO DEBE TENER
LA EXTENSIÓN PHP

INCLUIR LOS TAGS DE INICIO Y CIERRE

<?php

...

?>

EL CÓDIGO DEBE ESTAR
ENTRE LOS TAGS PHP

PHP: Sintaxis

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Hello PHP!</title>
</head>
<body>
  <?php
    $titulo = "Hello World!!!";
    echo "<h1>" . $titulo . "<h1>";
  ?>
</body>
</html>
```

AL CLIENTE LE LLEGA

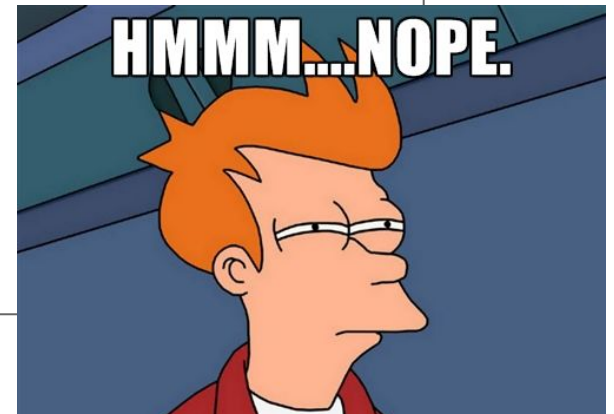
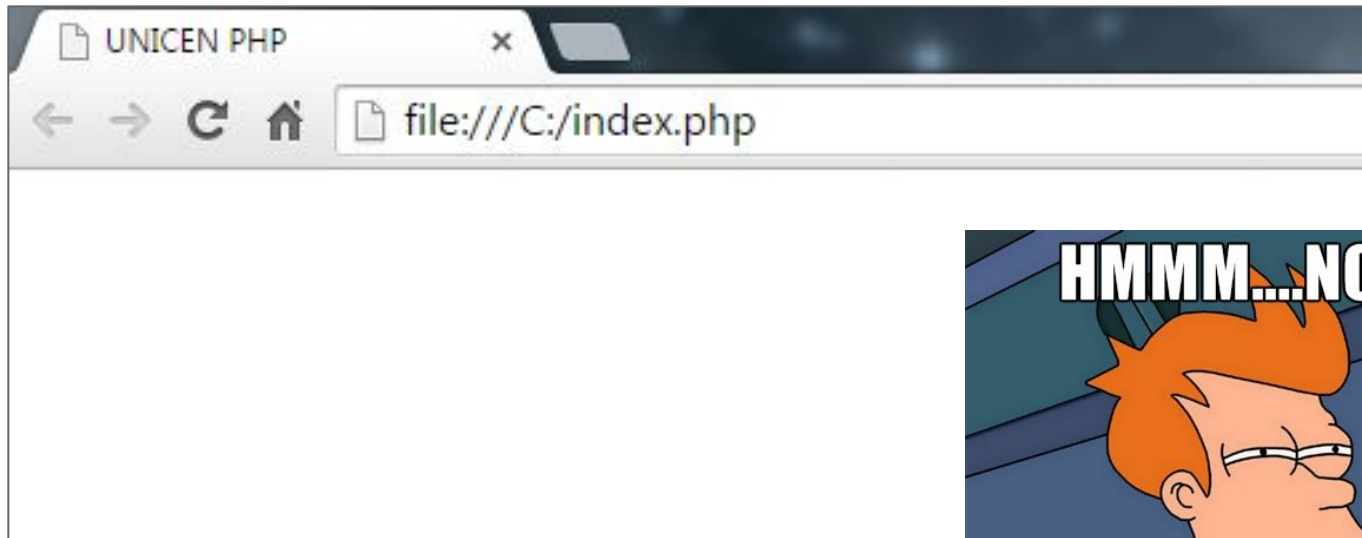
```
...
<body>
  <h1> Hello World!!! </h1>
</body>
...
```

**¿Como hacemos
“andar” nuestro
archivo (script) PHP?**



PHP

Si creo un archivo de extension php y lo abro en el navegador con doble click. ¿Qué pasa?



Error común

Estudiante: No me anda la página!!!

Docente: Tenes un PHP y lo estas abriendo directamente en el browser. Servilo desde Apache para que el módulo PHP lo interprete.



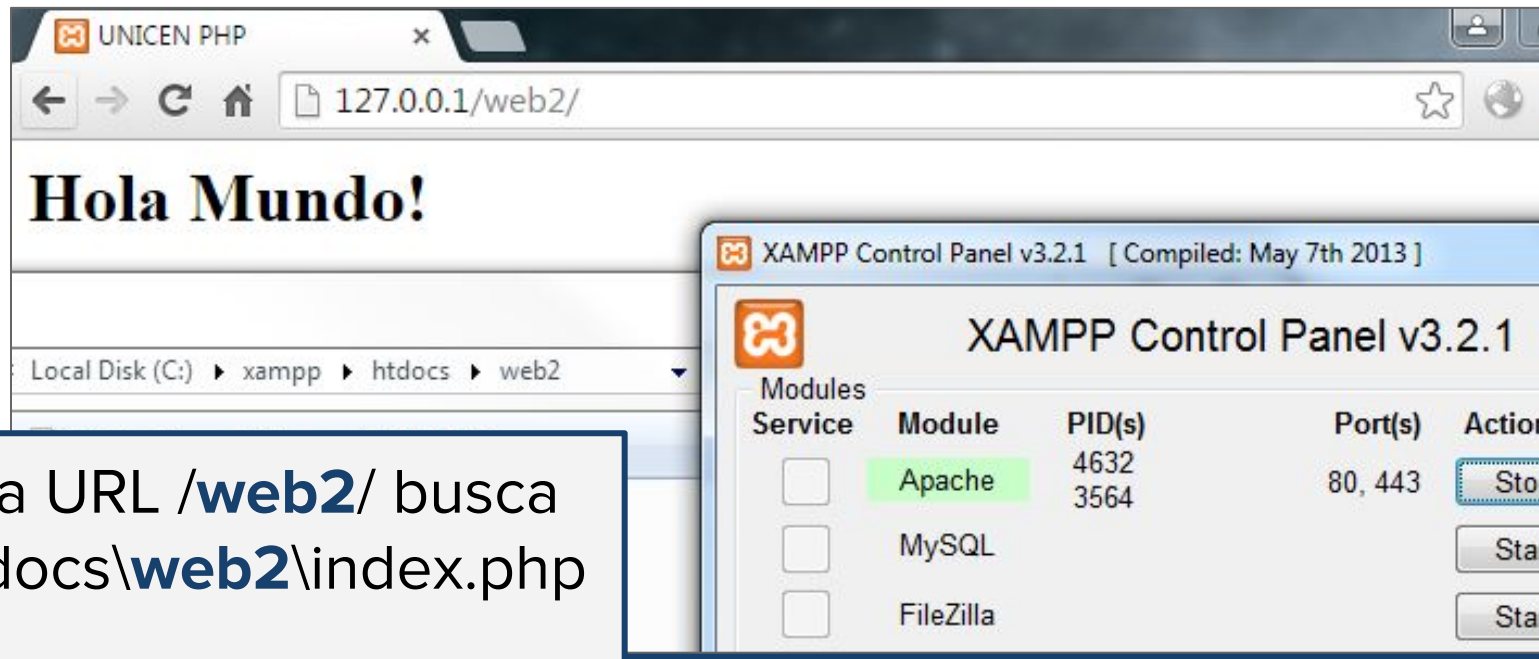
OH REALLY?

**HOW
INTERESTING**

Servidor WEB

Tenemos que “servir” nuestros archivos usando un servidor web:

- XAMPP - Apache usa por default la carpeta **‘htdocs’** para servir archivos (en windows c:\xampp\htdocs).
- Ponemos los archivos donde el servidor pueda verlos y usamos nuestra dirección local (localhost/127.0.0.1). Apache traduce la ruta relativa de la URL a la estructura de carpetas que tenemos en el servidor.



Al poner en la URL **/web2/** busca
C:\xampp\htdocs**web2**\index.php

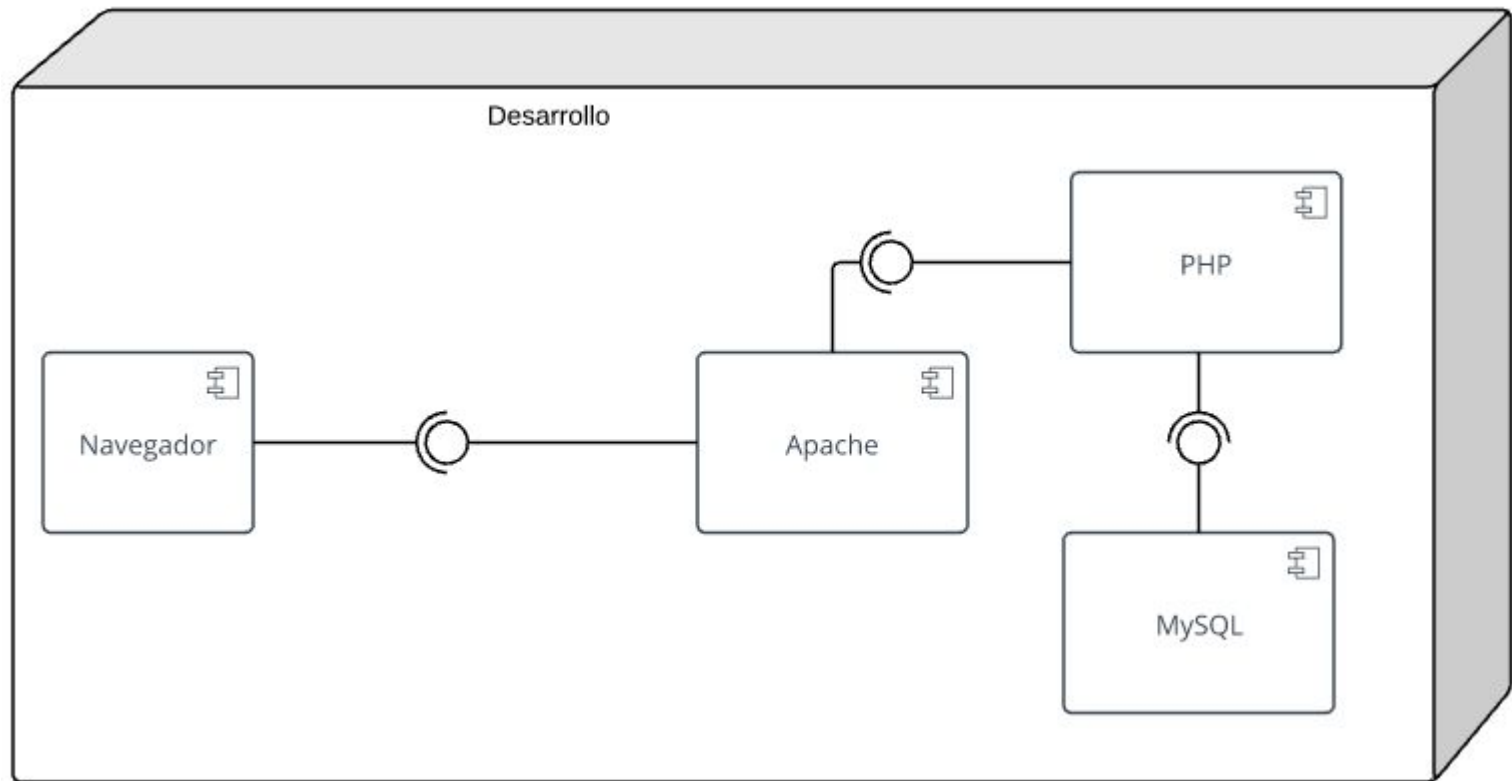
Funciona!

- Si el archivo es un PHP y el módulo de PHP está instalado, Apache ejecuta el script PHP automáticamente
- Lo que genera el script es lo que se envía al navegador



Pero... ¿no era que iba a un servidor?

Cuando estoy desarrollando, todos los componentes están en mi PC.



PHP: El lenguaje

PHP: Variables

VARIABLES

- PHP es un lenguaje **no tipado**: el tipo se define por el contexto en el que es usada
- Se definen **implícitamente**: no hay que declararlas
- El nombre siempre empieza con **\$**

```
<?php
```

```
$aBool = true;    // boolean
```

```
$name  = "Juan";  // string
```

```
$lastName = "Perez"; // string
```

```
$cont = 12;      // integer
```

```
echo "$name, $lastName"; // outputs "Juan, Perez"
```

```
?>
```

PHP: Constantes

CONSTANTES

Como el nombre sugiere, este valor no puede variar durante la ejecución del script.

- Se usa el método *define(nombre, valor)*
- Para leerlas, usamos el nombre sin '\$'

```
<?php
    define("SALUDO", "Hello world.");
    echo SALUDO; // outputs "Hello world."
?>
```

PHP: Arreglos

Arreglos indexados

Pueden ser creados usando el constructor `array()`.

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    // Asignación manual
    $cars[0] = "Volvo";
    $cars[1] = "BMW";
    $cars[2] = "Toyota";
    $count = count($cars); // 3 (int)
?>
```


PHP: Arreglos

Arreglos asociativos

Se construyen indicando pares *clave => valor* separados por coma como argumento.

```
<?php
    $edades = array(
        "juan" => 35,
        "nico" => 17,
        "julia" => 23
    );
    echo $edades["juan"]; //imprime 35
    echo $edades["julia"]; //imprime 23
?>
```

Ejercicio

Hagamos una página que nos muestre la edad de un usuario en particular.

- En un arreglo asociativo tenemos usuario y edad.

Ejemplo:

`$edades["juan"] = 31`

- En una constante tenemos el nombre del usuario que queremos mostrar.



PHP: Arreglos

Operaciones sobre **arreglos**

- **Insertar** un elemento al final
 - `array_push($arreglo, $valor)`
- **Extraer** el último elemento del array
 - `array_pop($arreglo)`
- **Invertir** orden de los elementos
 - `array_reverse($arreglo)` (Devuelve un arreglo nuevo)
- Operaciones **aritméticas** sobre arreglos
 - `array_sum($arr)` Calcula la suma de los valores
 - `array_product($arr)` Calcula el producto de los valores

Todas las operaciones en: <http://php.net/manual/es/ref.array.php>

PHP: Arreglos

Operadores para arreglos

Ejemplo	Efecto	Resultado
<code>\$a + \$b</code>	Unión	Unión de \$a y \$b.
<code>\$a == \$b</code>	Igualdad	TRUE si \$a y \$b tienen las mismas parejas clave/valor.
<code>\$a === \$b</code>	Identidad	TRUE si \$a y \$b tienen las mismas parejas clave/valor en el mismo orden y de los mismos tipos.
<code>\$a != \$b</code>	Desigualdad	TRUE si \$a no es igual a \$b.
<code>\$a <> \$b</code>	Desigualdad	TRUE si \$a no es igual a \$b.
<code>\$a !== \$b</code>	No Identidad	TRUE si \$a no es idéntica a \$b.

PHP: Funciones

Una función puede ser definida empleando una sintaxis como la siguiente:

```
<?php
    /**
    * Calcular el promedio de los valores de un arreglo
    **/
    function promedioEdad($edades){
        $promedio = array_sum($edades) / count($edades);
        return $promedio;
    }
?>
```

<https://www.php.net/manual/es/functions.user-defined.php>

PHP: Ámbitos

```
<?php
$a = 1; /* ámbito global */

function test() {
    echo $a; /* referencia a una variable del ámbito local */
}

test();
?>
```

¿Qué pasa cuando ejecuto este script?

¿Cómo usar las globales? ¿Es una buena práctica?

PHP: Ámbitos (solución)

```
<?php
$a = 1; /* ámbito global */
function test() {
    GLOBAL $a;
    echo $a;
}
test();
?>
```

¿Cómo usar las globales? ¿Es una buena práctica?

- Se puede indicar que `$a` es global, con `"global $a"` o accediendo directamente con `"$GLOBALS['a']"`.
- Usar variables globales es mala práctica!

PHP: Estructuras de control

- if, else, elseif, switch
- while, do-while, for
- foreach
- break, continue

IF

```
<?php
if($a > $b){
    echo "a es mayor que b";
} else {
    echo "a NO es mayor que b";
}
if($a > $b) {
    echo "a es mayor que b";
} elseif ($a == $b) {
    echo "a is igual a b";
} else {
    echo "a menor que b";
}
?>
```

SWITCH

```
<?php
$i = 3;
switch($i){
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual 1";
        break;
    default:
        echo "i es distinto a 0 y 1";
        break;
}??>
```

PHP: Estructuras de control

FOR

```
for ($i = 0; $i < $count; $i++) {  
    echo "<li>" . $cars[$i] . "</li>";  
}
```

WHILE

```
$i = 0;  
while ($i < $count) {  
    echo "<li>" . $cars[$i] . "</li>";  
    $i++;  
}
```

// Arreglo indexado

```
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";  
$count = count($cars);
```

FOREACH

```
foreach ($cars as $car) {  
    echo "<li>".$car."</li>";  
}
```

PHP: Estructuras de control

FOR

```
for ($i = 0; $i < $count; $i++) {  
    echo "<li>" . $cars[$i] . "</li>";  
}
```

WHILE

```
$i = 0;  
while ($i < $count) {  
    echo "<li>" . $cars[$i] . "</li>";  
    $i++;  
}
```

FOREACH

```
foreach ($cars as $car) {  
    echo "<li>". $car . "</li>";  
}
```



MALAS PRÁCTICAS

Las resolveremos
más adelante

PHP: Strings

Operadores para **cadenas de texto**

- `.` → concatena strings
- `.=` → concatena strings al final

```
<?php
    $bar = "Mundo";
    $foo = "Hola " . $bar;
    $foo .= "!";
    echo $foo; //outputs Hola Mundo!
?>
```

PHP: Strings

Funciones para **cadena de textos**

```
<?php
```

```
    echo strlen("Hello world!"); // outputs 12
    echo str_word_count("Hello world!"); // outputs 2
    echo strrev("Hello world!"); // outputs !dlrow olleH
    echo strpos("Hello world!", "world"); // outputs 6
    echo str_replace("world", "Dolly", "Hello world!"); //
outputs Hello Dolly!
```

```
?>
```

Una aplicación web: muchos archivos

Ejecución de archivos

index.php

Es el archivo con el script PHP que **Apache** elige ejecutar por defecto.

- Si se quiere ejecutar otro archivo desde el browser, cada uno tiene su propia URL.
- Los archivos se pueden incluir entre sí para organizar el código.

PHP: Include

`include 'file_name.php'`

Incluye el archivo especificado. Todas las funciones y variables definidas en el archivo incluido tienen **alcance global**.

localidades.php

```
<?php
$prov = 'BA';
$ciudad = 'Tandil';
?>
```

test.php

```
<?php
// Todavía no se incluyó el archivo
echo "Origen $ciudad $prov"; // Origen
include 'localidades.php';
// Ahora sí, las variables toman valor
echo "Origen $ciudad $prov"; //Origen Tandil-BA
?>
```


PHP: Require

`require 'file_name.php'`

Es igual a `include` excepto que en caso de fallo producirá un error fatal de nivel `E_COMPILE_ERROR`.

- Se detiene el script mientras que `include` sólo emitirá una advertencia (`E_WARNING`) lo cual permite continuar el script.
- Algunas posibles causas de fallos son:
 - El archivo no existe
 - El archivo no es accesible (permisos)

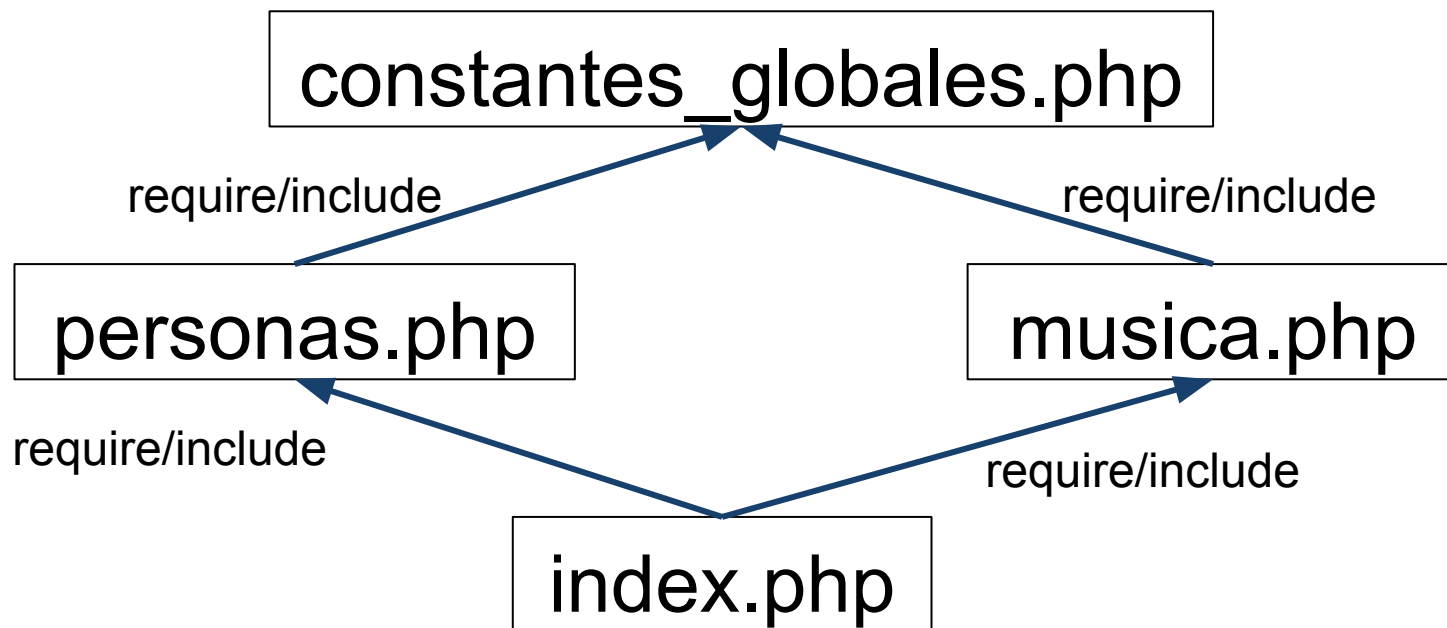
PHP include_once, require_once

`require_once 'file_name.php'`

Es idéntica a require salvo que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye de nuevo.

Sucede lo mismo con la sentencia `include_once`.

require_once

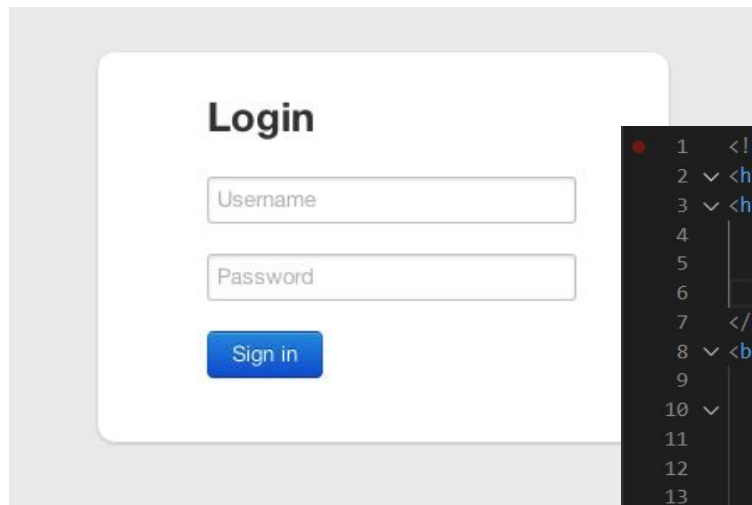


Enviar datos desde un cliente al
servidor

Enviar datos al servidor

Uno de los mecanismos más naturales para enviar datos al servidor es mediante un **FORMULARIO HTML**

- Son uno de los principales puntos de interacción entre un usuario y un sitio web o aplicación.
- Permiten a los usuarios la introducción de datos, que generalmente se envían a un servidor web para su procesamiento y almacenamiento



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>SignIn</title>
7 </head>
8 <body>
9
10  <form action="login.php" method="POST">
11    <input type="text" name="username" placeholder="Username">
12    <input type="password" name="password" placeholder="Password">
13
14    <button type="submit">Sign In</button>
15  </form>
16 </body>
17 </html>
```

Partes de un Formulario Web

El atributo **action** define la ubicación (URL) donde se envían los datos del formulario..

El atributo **method** define con qué método HTTP se envían los datos (generalmente get o post).

```
</head>
<body>

  <form action="login.php" method="POST">
    <input type="text" name="username" placeholder="Username">
    <input type="password" name="password" placeholder="Password">

    <button type="submit">Sign In</button>
  </form>
</body>
</html>
```

El formulario debe tener al menos un elemento **type="submit"** que enviará el formulario entero a la url correspondiente.

El atributo **name** identificará, en el lado servidor, a los datos ingresados por el usuario en el control. Se envía siempre un par clave/valor por cada elemento

Métodos HTTP

Dos de los métodos HTTP más utilizados con los que el navegador puede enviar información al servidor desde un formulario son:

- Método GET

```
<form method="GET">
```

El **método GET** envía la información codificada del usuario en **HTTP request**, directamente en la **URL**.

```
localhost/index.php?variable1=valor1&variable2=valor2&...
```

- Método POST

```
<form method="POST">
```

Con el método **HTTP POST** también se codifica la información, pero ésta se envía a través del **body** del **HTTP Request**, por lo que no aparece en la URL.

PHP: Variables `$_GET` y `$_POST`

¿Cómo accedemos a los valores enviados desde el cliente?

En PHP existen arreglos asociativos que se utilizan para recuperar la información enviada al servidor:

- `$_GET` se usa para recuperar datos enviados por GET

- `$_POST` se usa para recuperar datos enviados por POST

PHP: Ejemplo \$_GET

- Cuando se envía el formulario, la URL contiene los datos enviados.
- El request se verá como:

`http://localhost/web2/ejemploget.php?nombre=Carolina&edad=24`

formulario.html

```
<html>
  <form action="ejemploget.php" method="get">
    <input type="text" name="nombre" />
    <input type="text" name="edad" />
    <input type="submit">
  </form>
```

ejemploget.php

```
<?php
if(isset($_GET['nombre'])) {
    $usuario = $_GET['nombre'];
    $edad = $_GET['edad'];
    echo "<p>Usuario: " . $usuario . "</p>";
    echo "<p>Edad: " . $edad . "</p>";
}
?>
```


PHP: Ejemplo \$_POST

- Cuando se envía el formulario, la URL NO CONTIENE los datos enviados. Estos se envían con el Request HTTP.
- El request se verá como:

<http://localhost/web2/ejemplopost.php>

formulario.html

```
<html>
  <form action="ejemplopost.php" method="post">
    <input type="text" name="nombre" />
    <input type="text" name="edad" />
    <input type="submit">
  </form>
```

ejemplopost.php

```
<?php
if(isset($_POST['nombre'])) {
    $usuario = $_POST['nombre'];
    $edad = $_POST['edad'];
    echo "<p>Usuario: " . $usuario . "</p>";
    echo "<p>Edad: " . $edad . "</p>";
}
?>
```

Ejercicio

Hagamos una página que nos muestre la edad de un usuario en particular.



¿Cómo se verán las URL?

POR GET

<http://localhost/web2/edad.php?usuario=juan>

POR POST

<http://localhost/web2/edad.php>

PHP: Variables \$_REQUEST

- Existe tambien \$_REQUEST
- Contiene todo el contenido de \$_POST y \$_GET
- Se puede usar cuando esperamos parámetros por ambos métodos.

ejemploRequest.php

```
<?php
if(isset($_REQUEST['nombre'])){
    $nombre = $_REQUEST['nombre'];
    echo "Nombre: " . $nombre . "<br/>";
}
?>
```

Links



Se puede utilizar el método GET para armar URL's dinámicas que obtienen información específica según ciertos parámetros. Son conocidas como **query string** y se pueden armar sin usar un formulario.

`John Doe ----->` link al perfil del usuario con el id 20

`Página 2 ----->` link a la página dos de la lista de productos

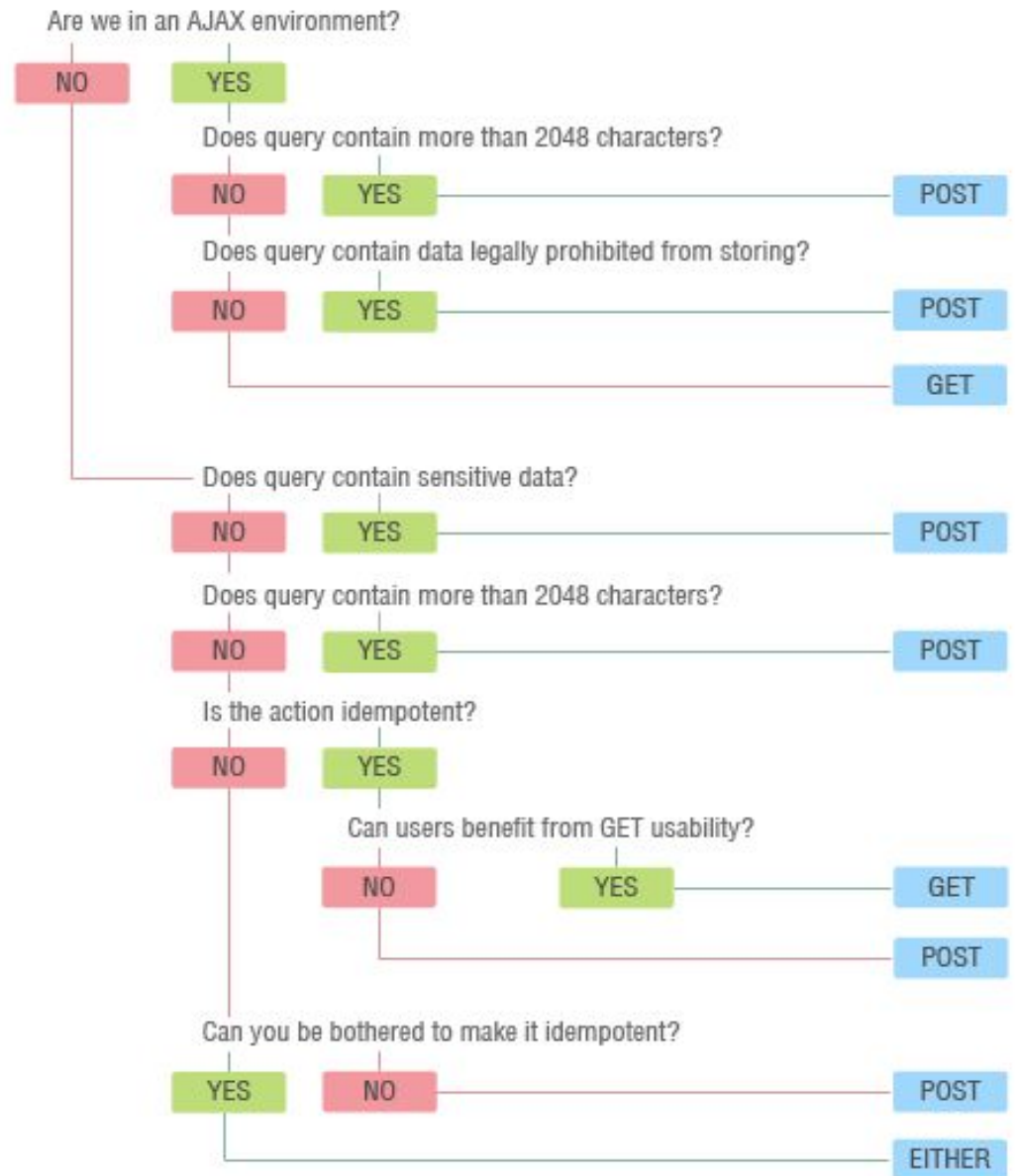
Pregunta rápida

¿Qué método usarían para enviar una contraseña?



PHP: GET vs POST

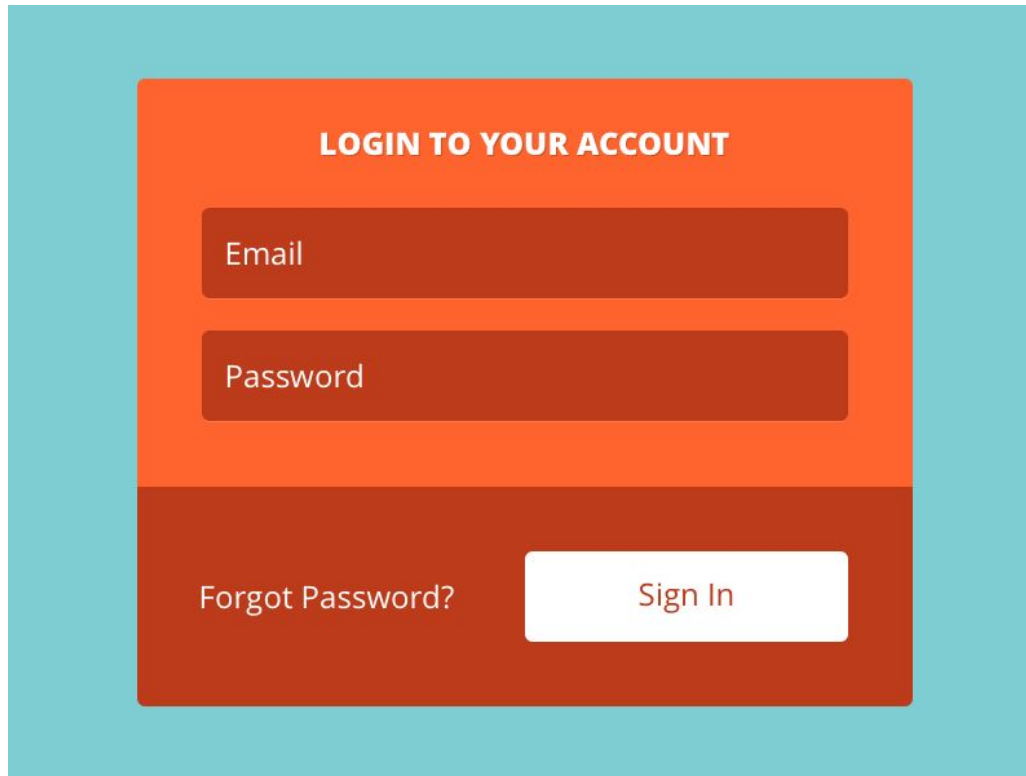
¿Cuándo usar uno u otro?



Formularios con AJAX

Formularios con Ajax

¿Es posible enviar un formulario al servidor sin recargar toda la página?



LOGIN TO YOUR ACCOUNT

Email

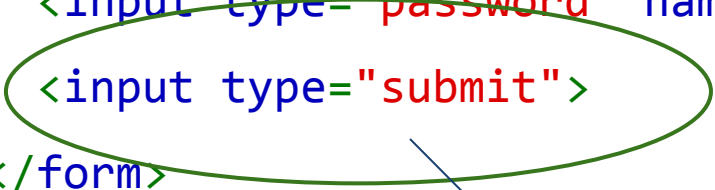
Password

[Forgot Password?](#)

Formularios con Ajax

Tenemos que realizar un llamado asincrónico desde JS para enviar el formulario, en vez de que lo envíe automáticamente el browser.

```
<form id="formulario" >  
  <input type="text" name="user" placeholder="Usuario" />  
  <input type="password" name="pass" placeholder="Clave" />  
  <input type="submit">  
</form>
```



...

IMPORTANTE

Controlar el envío del FORM mediante un *input type="submit"* o *button type="submit"*.

Formularios

JAVASCRIPT

1. Asignar un **Event Handler** para enviar el formulario vía Ajax:

```
document.querySelector('input[type=submit]').addEventListener('click', function(e) {  
...  
});
```

MALA PRACTICA!!!!

ESA MALA PRÁCTICA



```
document.querySelector('#formulario').addEventListener('submit', function(e) {  
    e.preventDefault(); //detiene el envío normal  
    ...  
});
```



Formularios

JAVASCRIPT

2. Obtener la información del formulario:

```
let user = document.querySelector("input[name=user]").value;  
let pass = document.querySelector("input[name=password]").value;
```

y si tenemos **gran cantidad** de datos??? **MALA PRACTICA!**

Usamos `FormData`

Los objetos `FormData` permiten compilar un conjunto de pares clave/valor para enviar mediante XMLHttpRequest (fetch).

```
const data = new URLSearchParams(new FormData(this));
```

Dentro de la función “handler” hace referencia al **formulario**.
CUIDADO, si usamos arrow function “this” no es el formulario.

Formularios

JAVASCRIPT

3. Enviar la información con **fetch()**

```
document.querySelector('#formulario').addEventListener('submit', function(e) {  
    e.preventDefault();  
    const data = new URLSearchParams(new FormData(this));  
  
    fetch('ejemplo.php', {  
        method: 'post',  
        body: data,  
    })  
    .then(response => response.text()) // el servidor nos devuelve HTML  
    .then(html => {  
        document.querySelector('#container').innerHTML = html;  
    })  
    .catch(error => console.log(error));  
});
```

Tambien podemos usar
this.action y **this.method**
para tomar los valores del form

Como hacer Debug en el servidor

- Configurar Debugger
 - Más adelante
- Usar echo
 - Para variables simples

```
echo "Valor de la variable var: $var";
```

- **Usar var_dump:**
 - Para variables complejas como arreglos, objetos, etc

```
var_dump($var);
```

Más Información

- PHP - www.php.net
- Manual de PHP - <https://secure.php.net/manual/es/>
- XAMPP - <https://www.apachefriends.org/es/index.html>
- PHP The Right Way - <http://www.phptherightway.com/>
- PHP 5 Power Programming - http://ptgmedia.pearsoncmg.com/images/013147149X/downloads/013147149X_book.pdf