PHP

Año 2023

Índice de contenido

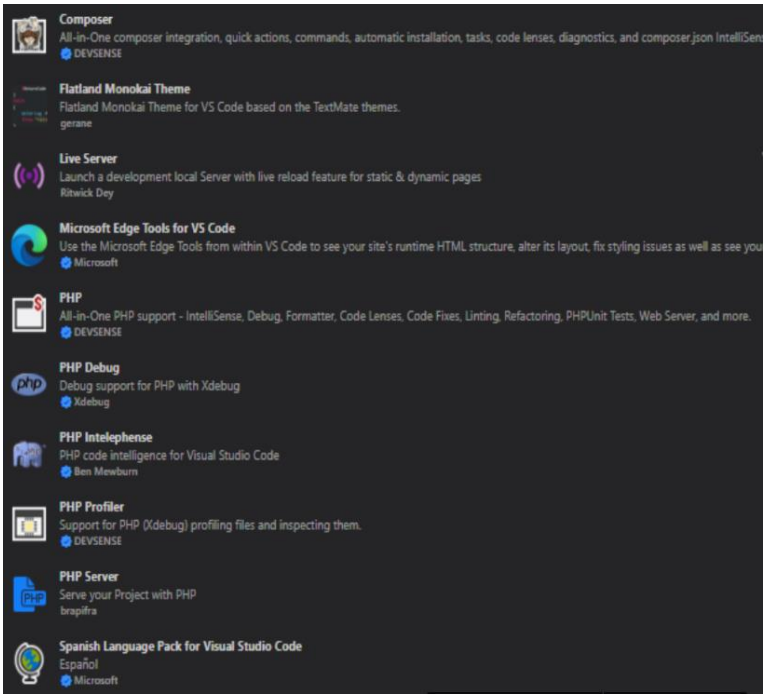
```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>index.html</title>
    <link rel="stylesheet" type="text/css" href="normalize.css">
    <link rel="stylesheet" type="text/css" href="estilo.css">
    <script src=https://kit.fontawesome.com/62ea397d3a.js crossorigin="Anonymous"></script>
</head>

<body>

    <header>
        <nav>

        </nav>
    </header>

    <main>

    </main>

    <footer>

    </footer>

</body>
</html>
```

```css
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0;
    font-weight: 100;
}
```

Requerimientos

Visual Studio Code

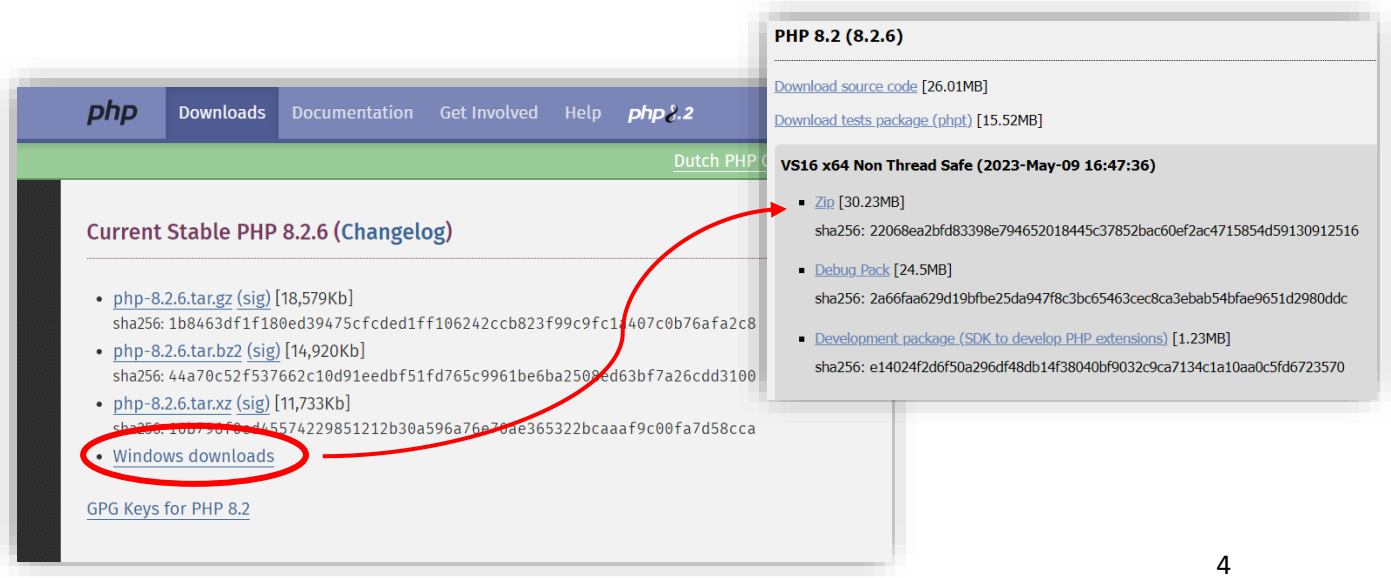
- Se requieren las siguientes extensiones



PHP

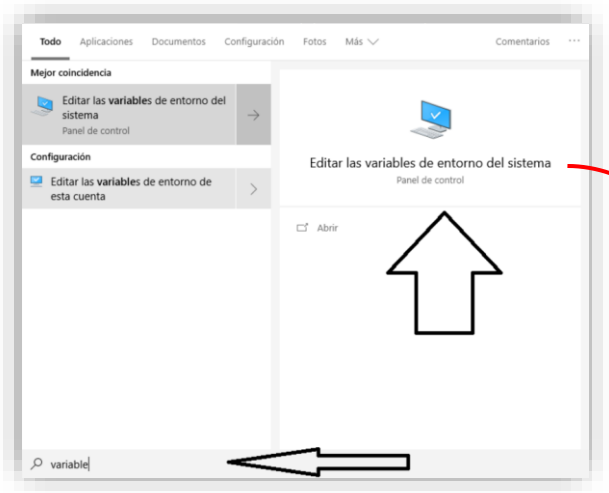
La instalación se hace directamente desde la página web, en la cual se descargará una carpeta la cual debe ubicarse dentro de un directorio de instalación que se desee (recomendable dentro del disco C).

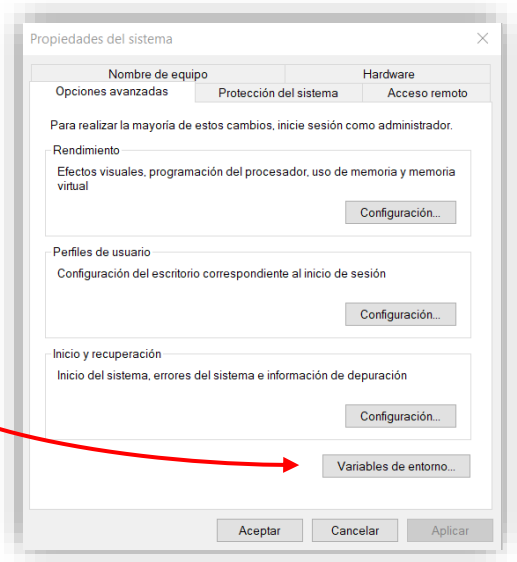
https://www.php.net/downloads.php

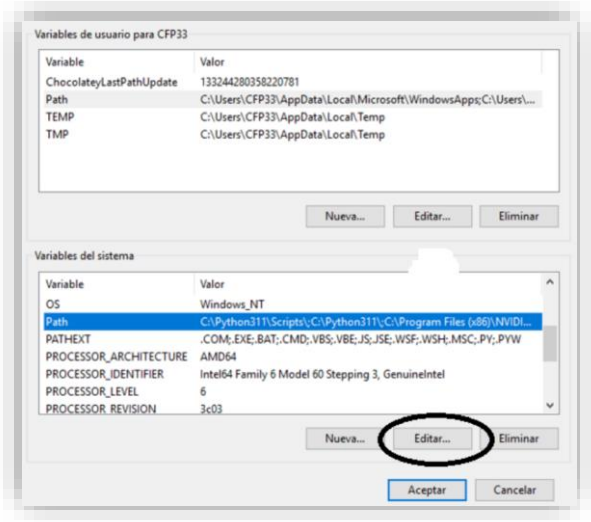


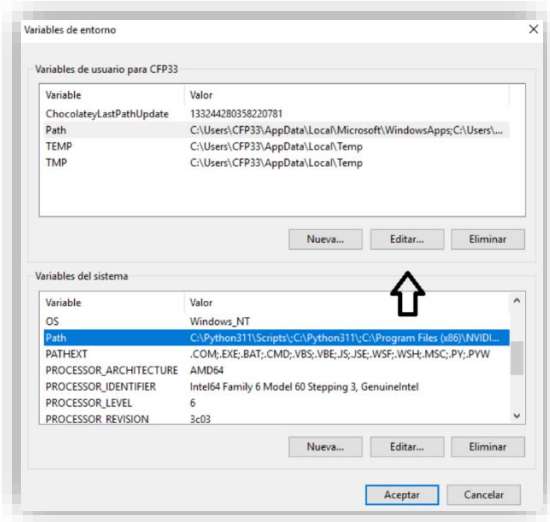
Variables de entorno

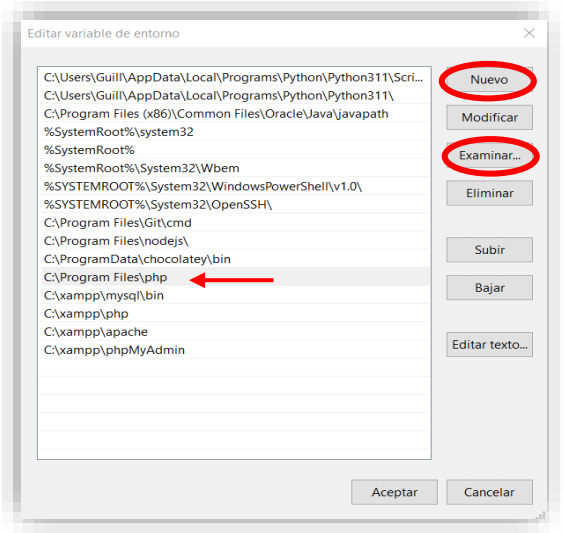
Es necesario para poder ejecutar un servidor en PHP.









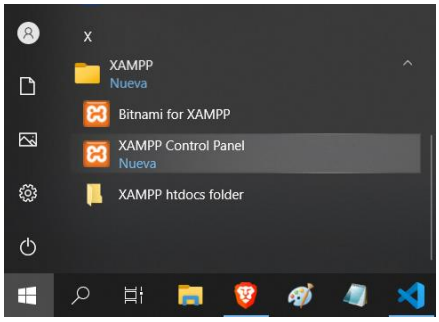


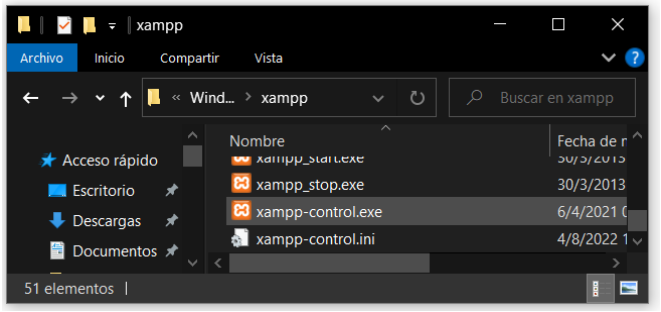
XAMPP

El servidor XAMPP permite alojar un servidor tanto en Windows como en Linux.

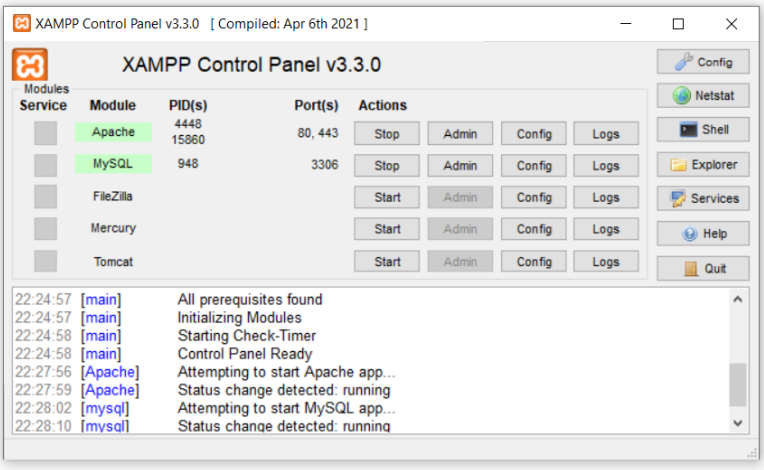
Una página segura de donde descargar es: https://www.apachefriends.org/es/download.html

Una vez instalado, se debe ejecutar el Panel de Control de XAMPP. Se puede encontrar en el menú inicio o en el directorio donde se instaló el programa.



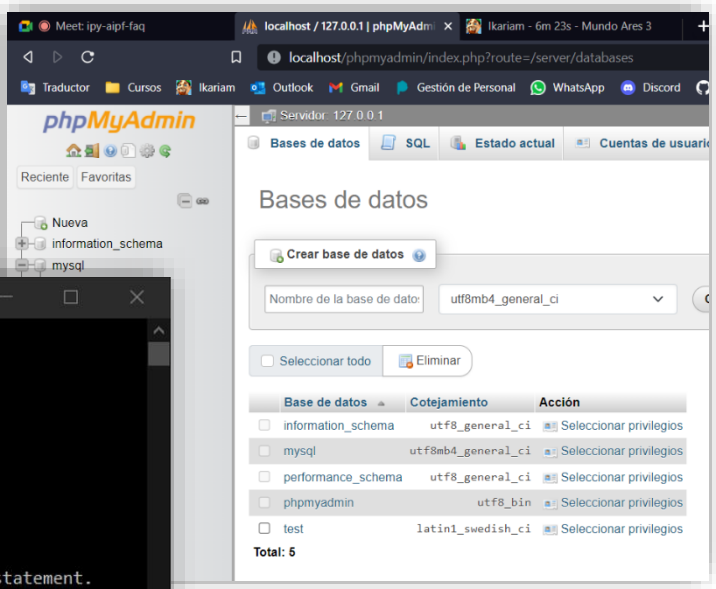


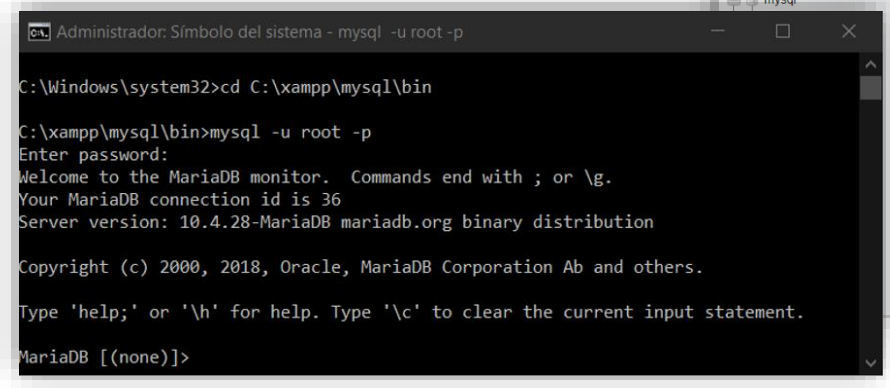
Luego, inicializar (start) tanto **Apache** como **MySQL** (los dos primeros).



De esta forma, quedan activa la Base de Datos con el servidor que puede ser usado por nuestra página web, la cual se puede visualizar desde phpMyAdmin.

O bien, trabajar desde la consola del sistema con privilegios de administrador.





Comandos de la Base de Datos

- Ver Bases de Datos existentes `SHOW DATABASES;`
- Crear una Base de Datos `CREATE DATABASE nombre_database;`
- Borrado de datos
  - Borrar una Base de Datos `DROP DATABASE nombre_database;`
  - Borrar una tabla `DROP TABLE nombre_database;`

- Acceder a una Base de Datos `USE nombre_database;`
  - Ver Tablas `SHOW TABLES;`
  - Ver detalles de una Tabla `DESCRIBE nombre_tabla;`
  - Crear una Tabla `CREATE TABLE nombre_tabla (parámetros);`

    donde los parámetros se envía el nombre de la columna, el tipo de dato, se especifica si puede ser NULL y cuál es la columna primaria

```
CREATE TABLE table_name (id INT AUTO_INCREMENT,  name varchar(90) NOT NULL,  number INT,  PRIMARY KEY(id) );
```

    - Seleccionar una Columna `SELECT * FROM nombre_tabla;`
    - Ingresar una Fila `INSERT INTO nombre_tabla (id, name, number) VALUES (null, 'Guille', 28);`
    - Borrar una Fila `DELETE FROM nombre_tabla WHERE id = 2;`
    - Editar una Fila `UPDATE nombre_tabla SET name = 'Guille' WHERE name = 'Nico';`

Existen varias formas de usar WHERE:

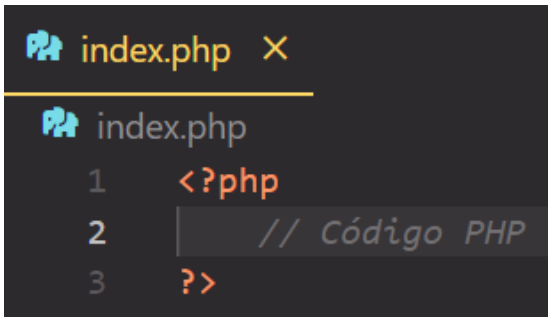
```
select * from usuarios WHERE nombre like 'y%';

select * from usuarios WHERE nombre like 'm%a';

select * from usuarios WHERE email like '%com';

select * from usuarios ORDER BY nombre DESC;

select * from usuarios ORDER BY nombre ASC;

select * from usuarios LIMIT 1;

select * from usuarios LIMIT 2;

select * from usuarios LIMIT 2,3;
```

Además, se puede usar `fecha DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP` para crear una columna que guarde la fecha en la que se realice un registro.

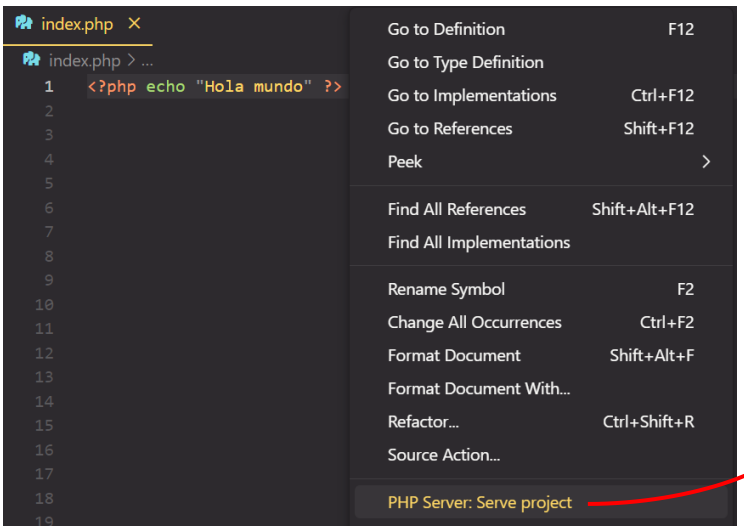
PHP

Permite la creación de páginas web alojadas en un servidor. Es la base de HTML y es la parte lógica de una página web, tanto para controlar una Base de Datos como para definir variables y programación backend.

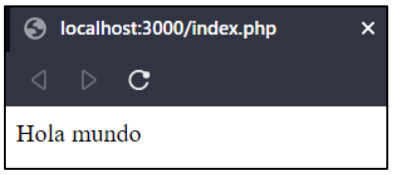
Un código PHP se escribe con el siguiente formato:



Un primer ejemplo es



El método `echo` permite imprimir por pantalla un mensaje



Ejecuta la página de PHP

Variables

Una variable es un espacio (o “caja”) que se guarda en memoria.

Tipos de datos

- **String** Una cadena de caracteres
- **Boolean** Un dato lógico (**true** o **false**)
- **Integer** Un número entero
- **Double** Un número con decimales
- **Array** Lista de datos (arreglo)

Declaración e inicialización de variables

Son los nombres que se le asignan a las variables para almacenarlas en memoria y trabajar con ellas.

Todas las variables inician con $

```php
$userName = "Guille";
```

Se puede saber el tipo de dato que se está utilizando en una variable con el método **gettype**

```php
gettype($userName); // -> string
```

Operadores

Símbolo que indica que debe ser llevada a cabo una operación especificada sobre cierto número de operador (tipo de dato).

- **Asignación** **=** Asigna un valor a un operando.

- **Aritméticos**
  - **Suma** **+** Suma dos números o concatena textos.
  - **Sustracción** **-** Resta dos números.
  - **Multiplicación** ***** Multiplica números.
  - **División** **/** Divide números.
  - **Resto** **%** Divide números y se obtiene el resto de dicha división.
  - **Exponenciación** ****** Aplica potencias.

- **Relacionales** Compara dos valores del mismo tipo y devuelve un valor lógico.
  - **Mayor estricto** **>** **true** *o* **false**
  - **Menor estricto** **<**
  - **Mayor o igual** **>=**
  - **Menor o igual** **<=**
  - **Igualdad** **==**
  - **Desigualdad** **!=**
  - **Idéntico** **===** los operandos de distinto tipo son considerados diferentes.
  - **No idéntico** **!==**

- **Lógicos**
  - **And** **&&** Si todo es verdadero, retorna verdadero.
  - **Or** **||** Si al menos uno es verdadero, retorna verdadero.
  - **Not** **!** Convierte lo verdadero en falso y viceversa.
  - **Xor** **^**

- **Binarios**
  - **Despl. Izq.** **<<**
  - **Despl. Der.** **>>**
  - **Despl. Sin signo** **>>>**

Concatenación

Para unir textos se utiliza un punto .

```php
$nombre = "Guillermo";
$apellido = "Hernandez";
```

```php
$nombreCompleto = $nombre . $apellido;
```

```
"GuillermoHernandez"
```

Uso de variables en textos

Se puede acceder a una variable directamente desde un string de la siguiente manera:

```php
$world = "mundo";

$mensaje = "Hola $world";
```

→ Hola mundo

A su vez, si se utiliza entre llaves, se puede acceder al valor de otra variable.

```php
$mundo = "¿Variables dentro de variables?";

$world = "mundo";

$mensaje = "Hola {$$world}";
```

Hola ¿Variables dentro de variables?

Estructuras condicionales

Herramientas que permiten el control del flujo de información.

Condicionales

Ejecutan un bloque de código si se cumple una condición.

```php
$manzanas = 3;

if ($manzanas == 2) {
    echo "Hay 2 manzanas";
}
elseif ($manzanas == 1) {
    echo "Hay " . $manzanas . " manzanas";
}
else {
    echo "Hay $manzanas manzanas";
}
```

```php
switch ($manzanas) {
    case 2:
        echo "Hay 2 manzanas";
        break;

    case 1:
        echo "Hay " . $manzanas . " manzanas";
        break;

    default:
        echo "Hay $manzanas manzanas";
        break;
}
```

Condicional ternario

```php
echo ($manzanas > 0) ? "Hay manzanas" : "No hay manzanas";
```

Strings

- empty — devuelve FALSE si la cadena no está vacía (o si la variable no existe).
- strlen — longitud de la cadena.
- strpos — busca la primera aparición de una subcadena.
- substr — devuelve una parte de la cadena.
- str_replace — reemplaza todas las apariciones de una subcadena.
- strtoupper — convierte todo en mayúsculas.
- strtolower — convierte todo en minúsculas.
- trim — quita los espacios en blanco.
- ltrim — quita los espacios en blanco a la izquierda.
- rtrim — quita los espacios en blanco a la derecha.
- explode — divide una cadena en un array
- implode — une un array en una cadena
- strrev — invierte una cadena

- htmlspecialchars — convierte los caracteres especiales de HTML para que el navegador no los entienda como etiquetas y así evitar inyección de HTML por parte del usuario por medio de strings.

Arrays

Conjunto de elementos (no necesariamente del mismo tipo de dato)

Estos arrays pueden ser de tipo simple

```php
$array = ['Guille', 28, true];
```

o bien de tipo asociativo (clave – valor)

```php
$array = array("key" => "value");
```

Existen diversos métodos que se pueden usar

- empty — devuelve FALSE si el array tiene elementos (o si la variable no existe).
- count — cantidad de elementos.
- array_push — agrega un elemento al final del array.
- array_pop — elimina y devuelve el elemento al final del array.
- array_shift — elimina y devuelve el elemento al principio del array.
- array_unshift — agrega un elemento al principio del array.
- array_merge — combina dos arrays.
- array_slice — extrae una parte de un array y lo devuelve como un nuevo array.
- array_splice — cambia el contenido de un array, eliminando, reemplazando o insertando.
- in_array — comprueba si un valor existe en un array
- array_search — busca un valor en un array y devuelve su clave si la encuentra.
- array_keys — devuelve todas las claves de un array como un nuevo array.
- array_values — devuelve todos los valores de un array como un nuevo array.
- sort — ordena un array en orden ascendente.
- rsort — ordena un array en orden descendente.
- asort / arsort — ordena un array asociativo en orden (as/des)cendente por valores.
- ksort / krsort — ordena un array asociativo en orden (as/des)cendente por claves.

```php
$array = array();
$array[] = "Hola"; #push
```

- array_filter — filtra elementos de un array basándose en una función.
- array_map — aplica una función a cada elemento de un array y devuelve un nuevo array con los resultados.

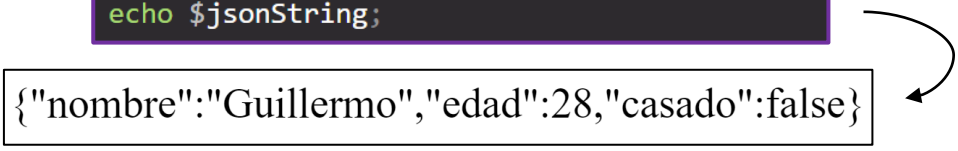
Array asociativo

Un array asociativo es aquel que tiene personalizado los nombres de las posiciones de sus elementos. Es decir, no se identifican los elementos por su index en el array sino por un nombre específico.

```php
$asociativeArray = array(
    "nombre" => "Guillermo",
    "edad" => 28,
    "casado" => false
);
```

Este tipo de arrays se utiliza para trabajar con objetos de tipo JSON mediante los métodos:

```php
$jsonString = json_encode($asociativeArray);

// json serializado (convertido en string)
echo $jsonString;
```



A su vez, los JSON serializados (string) se pueden convertir en Arrays Asociativos mediante el método:

```php
$object = json_decode($jsonString);

// json deserializado (objeto / array asociativo)
echo gettype($object) . "<br>";
```

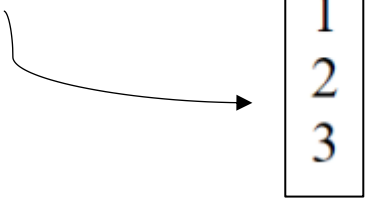
Iteradores

Ejecuta un bloque de código varias veces.

While

Ejecuta un bloque de código hasta que la condición deje de ser verdadera.

```php
<?php
    $numero = 0;
    while ($numero < 4) {
        echo $numero . "<br>";
        $numero++;
    }
?>
```



Do-While

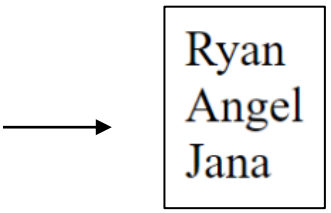
Ejecuta primero el bloque de código y luego, si la condición es verdadera, repite el bloque de código.

For

```
for( declaracion de variable ; condicion ; iteracion )
```

Ejecuta un bloque de código una determinada cantidad de veces.

```php
<?php
    $array = ['Ryan', 'Angel', 'Jana'];
    for ($i = 0; $i < count($array); $i++) {
        echo $array[$i] . "<br>";
    }
?>
```



- **break** Termina la iteración / bucle de forma forzada, ignorando las condiciones establecidas.
- **continue** Saltea la iteración actual pero no termina el bucle.

ForEach

Permite recorrer un conjunto de elementos sin necesidad de tener en cuenta la cantidad de elementos que existan.

```php
<?php
    $array = ['Ryan', 'Angel', 'Jana'];
    foreach ($array as $element) {
        echo $element . "<br>";
    }
?>
```

→

Ryan
Angel
Jana

Si se quiere obtener el index de cada elemento, se utiliza el siguiente formato:

```php
<?php
    $array = ['Ryan', 'Angel', 'Jana'];

    foreach ($array as $index => $element) {
        echo "[$index]: $element <br>";
    }
?>
```

→

[0]: Ryan
[1]: Angel
[2]: Jana

Además, permite el recorrido de un **Array Asociativo** / **Objeto JSON**.

```php
<?php

    $json = array(
        "nombre" => "Guillermo",
        "edad" => 28,
        "casado" => false # bool no se imprime
    );

    foreach ($json as $key => $value) {
        echo "$key: $value <br>";
    }
?>
```

nombre: Guillermo
edad: 28
casado:

Funciones

Definición de una tarea dentro de un bloque de código para luego poder realizar dicha tarea en nuestro programa con solo hacer referencia a dicho bloque de código creado.

```
function nombre_de_mi_función(datos de entrada) {
    . . .
}
```

Los datos de entrada (**parámetros**) son los datos que recibe la función y con los que puede trabajar.

Una función puede devolver un valor mediante la palabra reservada **return**, la cual terminará la ejecución de la función. Dicho valor puede ser guardado dentro de una variable.

```php
<?php
    1 reference
    function sumar($a, $b) {
        return $a + $b;
    }

    $resultado = sumar(1, 2);

    echo $resultado;
?>
```

3

Parámetro rest

En general, las funciones pueden recibir un número específico de parámetros (los definidos en su firma), sin embargo se pueden definir parámetros rest que pueden obtener parámetros no previstos pasados a una función, los cuales son procesados en un array.

```php
<?php
    1 reference
    function sumar(...$params) {
        // $params es un array

        $resultado = 0;

        foreach ($params as $param)
            $resultado += $param;

        return $resultado;
    }

    $sumatoria = sumar(1, 2, 4, 5, 6);

    echo $sumatoria;
?>
```

18

Control y Manejo de errores

El flujo del programa es el orden en que se ejecuta el código. En PHP, esto se realiza de arriba hacia abajo, y se divide en bloques de código marcados con llaves `{ … }` dentro de bloques globales `<?php … ?p>`

El flujo se puede controlar mediante las sentencias condicionales de las estructuras de control (if, if-else, switch).

Sin embargo, el flujo del programa se verá completamente interrumpido si se presenta una **excepción** (cualquier tipo de error que se presente durante la ejecución del programa).

Para evitar que un error detenga la ejecución completa del programa, se utilizan las sentencias de manejo de excepciones.

```php
<?php

    try {
        # Bloque que puede contener un error
    }
    catch (Error $e) {
        echo $e; // Muestra el error
    }
    finally {
        /* Bloque que se ejecutará siempre
        incluso si hay un return en el bloque try */
    }

?>
```

Se pueden ejecutar errores forzados mediante la palabra reservada **throw**.

```php
throw new Error("Personalizado");
```

Die

Este método detiene completamente la ejecución de código y de la página en general, incluso si existiera código HTML posterior.

```php
<?php
    echo "Este mensaje se mostrará";
    die();
    echo "Este mensaje no";
?>
```

Clases

En PHP existe la programación orientada a objetos, y se pueden crear clases.

```php
<?php
    class Persona {

        // Atributos
        private $name;
        private $age;


        // Constructor
        public function __construct(string $nombre, int $edad) {
            $this->name = $nombre;
            $this->age = $edad;
        }


        // Métodos
        public function esMayorDeEdad(): bool {
            return $this->age >= 18;
        }
        public function presentarse(): void {
            echo "Hola, soy " . $this->name . " y tengo " . $this->age . " años.";
        }

        // Métodos estáticos (no requieren instancia)
        public static function descripcion() {
            echo "Una persona es un ser humano.";
        }
    }
?>
```

Creada la clase, se pueden crear instancias / objetos de la clase

```php
<?php
    // Instancia de clase
    $Guille = new Persona("Guillermo", 28);
    $Guille -> presentarse();
?>
```

```php
<?php
    // Uso de método estatico
    Persona::descripcion();
?>
```

Herencia

Creación de clases padres – hijas en PHP con la palabra reservada **extends**.

Para poder acceder a atributos de la clase padre, es necesario que estén declarados con el modificador de acceso **protected**.

```php
protected $name;
protected $age;
```

```php
<?php

    class Trabajador extends Persona {

        private $work;

        public function __construct(string $nombre, string $edad, string $nombreTrabajo) {
            parent::__construct($nombre, $edad); // constructor de Persona
            $this->work = $nombreTrabajo;
        }

        public function presentarse(): void {
            echo "Hola, soy $this->name, tengo $this->age años y trabajo como $this->work.";
        }
    }
?>
```

Así, se puede hacer una instancia de la clase.

```php
<?php
    // Instancia de clase
    $Guille = new Trabajador("Guillermo", 28, "Enfermero");
    $Guille -> presentarse();
?>
```

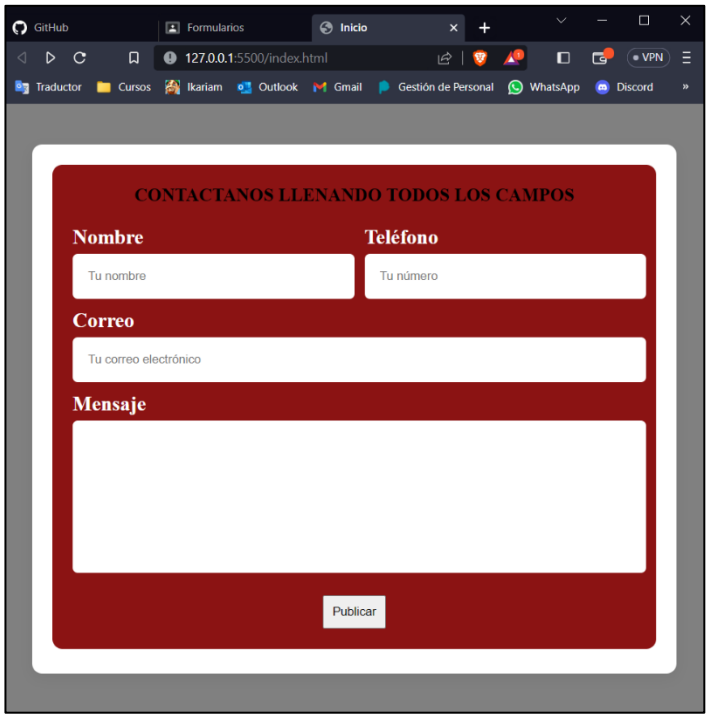
Notar que el método **presentarse** se encuentra definida en ambas clases, pero se utilizará la de la clase más hija ya que esta sobreescribe la de la clase padre.

Se pueden obtener o importar clases, y cualquier otro tipo de código alojado en otro archivo php, mediante la palabra reservada **include**.

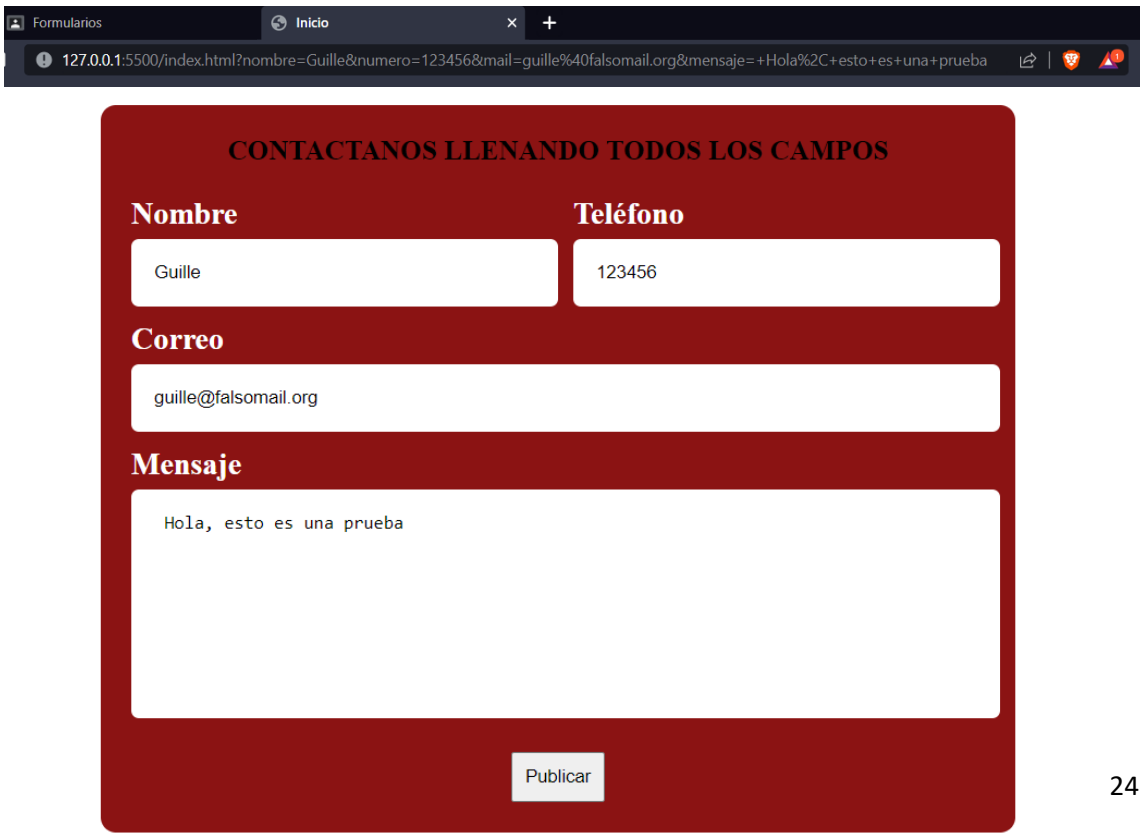
```php
<?php

    // Obtener la clase alojada en otro archivo php
    include ("Persona.php");

    // Instancia de clase
    $Guille = new Persona("Guillermo", 28);
    $Guille -> presentarse();

?>
```

Formularios

```html
<form class="contacto">
    <fieldset class="form-campos">
        <legend>Contactanos llenando todos los campos</legend>

        <div class="contenedor-campos">

            <div class="campo">
                <label>Nombre</label>
                <input type="text" placeholder="Tu nombre" name="nombre">
            </div>

            <div class="campo">
                <label>Teléfono</label>
                <input type="tel" placeholder="Tu número" name="numero">
            </div>

            <div class="campo">
                <label>Correo</label>
                <input type="email" placeholder="Tu correo electrónico" name="mail">
            </div>

            <div class="campo">
                <label>Mensaje</label>
                <textarea type="text" name="mensaje"> </textarea>
            </div>
        </div>

        <div class="boton-formulario">
            <input type="submit" value="Publicar">
        </div>

    </fieldset>
</form>
```



Luego de tocar el botón *submit* (publicar), se recargará la página con los parámetros escritos



Método GET

Permite obtener los parámetros enviados.

```php
<?php
    if ($_GET) { // Existen parámetros
        print_r($_GET); // Imprime el contenido del GET


        # Verificar que el parámetro existe y no está vacío
        if (isset($_GET['nombre']) && !empty($_GET['nombre'])) {
            $nombre = htmlspecialchars($_GET['nombre']); // Elimina caracteres html
            echo "¡Hola " . $nombre . "!";
        } else echo "No ingresaste un nombre";


        # Verificar que el parámetro es un número
        $numero = filter_var($_GET['numero'], FILTER_SANITIZE_NUMBER_INT);
        /* Caracteres no numéricos eliminados (CUIDADO, TAMBIÉN SE ELIMINA EL PUNTO) */
        if (filter_var($numero, FILTER_VALIDATE_INT) !== false) { /* Es un entero */
            echo "Tu número es: " . htmlspecialchars($numero);
        } else echo "Por favor, ingrese un número entero válido";


        # Verificar que el parámetro es un email válido
        if (filter_var($_GET['mail'], FILTER_VALIDATE_EMAIL)) {
            echo "Tu email es: " . htmlspecialchars($_GET['mail']);
        } else echo "Por favor, ingrese un correo válido";
    }
?>
```

También se pueden obtener los parámetros con JavaScript mediante:

```javascript
const params = new Proxy(new URLSearchParams(window.location.search), {
    get: (searchParams, prop) => searchParams.get(prop),
});

let nombre = params.nombre;
let numero = params.numero;
let email = params.mail;
let mensaje = params.mensaje;
```

Método POST

El método POST permite ocultar los parámetros de la dirección de la página, la cual es visible.

Para usarse, es necesario aclarar que se utilizará dicho método dentro del formulario de HTML.

index.php

```
index.php > html > body > main.content.main > section.formulario.som
35    <form class="contacto" method="post" action="procesar.php">
```

Aclarado el método, se procede a crear *procesar.php*, que es donde se procesarán los datos del formulario sin exponerlos en la URL de la página.

Puede ser de utilidad usar un condicional ternario para obtener los casos en que lo que se esté pasando por método post sea un objeto json.

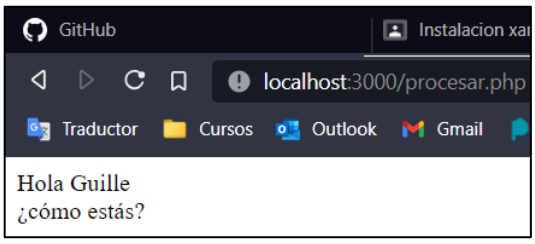
```
$POST = ($_POST) ? $_POST : json_decode(file_get_contents("php://input"), true);
```

index.php × procesar.php ×

procesar.php > ...

```
1  <?php
2      $nombre = $_POST['nombre'];
3      $correo = $_POST['mail'];
4      $numero = $_POST['numero'];
5      $mensaje = $_POST['mensaje'];
6
7      echo "Hola " . $nombre . "<br>";
8      echo "¿cómo estás?";
9  ?>
```

De esta manera, los datos se enviarán y obtendrán sin mostrar los datos.



Esta forma de ser redireccionado a otra página también se puede usar también para el método GET (reemplazando por GET)

```
<form class="contacto" method="get" action="procesar.php">
```

Bases de Datos

Para trabajar con Bases de Datos en PHP será necesario conocer los comandos de la base de datos a utilizar.

Es recomendable utilizar el objeto **PDO** para trabajar fácilmente, ya que esta API permite ejecutar queries de bases de datos directamente.

Cada vez que el PDO falla en cumplir la query, devuelve un error (el cual puede ser capturado con try – catch).

Creación de la base de datos

1. **Establecer la conexión al servidor**

   Lo primero a realizar es conectarse al servidor.

```php
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";

    try {
        $pdo = new PDO("mysql:host=$servername", $username, $password);
    }

    // La conexión con la base de datos falló (cuando la db no existe)
    catch (PDOException $noLogroConectarse) { echo $noLogroConectarse; }
?>
```

2. **Crear la base de datos**

   Una vez conectado al servidor, se puede crear la base de datos.

```php
try { $pdo -> query("CREATE DATABASE $dbname"); }
catch (PDOException $e) { echo "Ya existe"; }
```

3. **Crear las tablas a utilizar**

   Para crear tablas en una base de datos, al igual que en consola, es necesario definir que se trabajará con una determinada base de datos y luego ejecutar el comando requerido para crear tablas.

```php
try {
    $tablename = "users";
    $columnas = "id INT AUTO_INCREMENT, name varchar(90) NOT NULL, number INT, PRIMARY KEY(id)";

    $pdo -> query("USE $dbname");
    $pdo -> query("CREATE TABLE $tablename ($columnas)");
} catch (PDOException $e) { echo $e; }
```


Recordando el formato de la creación de tablas

```
CREATE TABLE table_name (id INT AUTO_INCREMENT,  name varchar(90) NOT NULL,  number INT,  PRIMARY KEY(id) );
```

Acceso a la base de datos desde PHP

Con todo creado y configurado, se pueden realizar las operaciones de bases de datos desde PHP.

- **Leer una tabla**

  Usaremos una tabla de ejemplo para mostrar:

```
MariaDB [dbtest]> SELECT * FROM users;
+----+---------+--------+
| id | name    | number |
+----+---------+--------+
|  1 | Guille  |     28 |
|  2 | Jose    |     44 |
|  3 | Marcela |     46 |
+----+---------+--------+
3 rows in set (0.000 sec)
```

  Al leer una tabla, se obtendrá un conjunto de arrays que representarán cada uno a una fila.

  Estos arrays serán de tipo asociativo pero también de tipo común. Esto quiere decir que se pueden acceder a sus valores tanto por el número de columna (siendo id el 0) como por el nombre de la columna.

```
Array ( [id] => 1 [0] => 1 [name] => Guille [1] => Guille [number] => 28 [2] => 28 )
Array ( [id] => 2 [0] => 2 [name] => Jose [1] => Jose [number] => 44 [2] => 44 )
Array ( [id] => 3 [0] => 3 [name] => Marcela [1] => Marcela [number] => 46 [2] => 46 )
```

  Usando PDO se puede recorrer la tabla de la siguiente manera:

```php
try {
    $rows = $pdo -> query("SELECT * FROM $tablename");

    foreach ($rows as $index => $row) { # $row is an array
        count($row) / 2; // Cantidad de columnas (incluída el ID)

        echo "[Fila $index] "; // index comienza desde cero
        echo "Nombre: " . $row['name'];
        echo " | Edad: " . $row['number'];
        echo "<br>";
    }
} catch (PDOException $e) { echo $e -> getMessage(); }
```

```
[Fila 0] Nombre: Guille | Edad: 28
[Fila 1] Nombre: Jose | Edad: 44
[Fila 2] Nombre: Marcela | Edad: 46
```

- **Añadir una fila en una tabla**

```php
try {
    $pdo -> query("INSERT INTO $tablename (id, name, number) VALUES (null, 'Guille', 28)");
                                                  claves                    valores
                                                (columnas)
} catch (PDOException $e) { echo $e; }
```

- **Eliminar una fila de una tabla**

```php
try {
    $pdo -> query("DELETE FROM $tablename WHERE id=2");
}
catch (PDOException $e) { echo $e; }
```

- **Editar una fila de una tabla**

```php
try {
    $pdo -> query("UPDATE $tablename SET name = 'Nicolas' WHERE name = 'Guillermo'");
    # Busca la fila que tenga como nombre 'Guillermo' y la cambia por 'Nicolas'
}
catch (PDOException $e) { echo $e; }
```

Inyección de código SQL

Hay que tener en cuenta que se puede inyectar código SQL en formularios y otros tipos de inputs donde el usuario puede escribir texto.

Para evitar esto, es necesario trabajar todo texto ingresado por el usuario para que no pueda ser interpretado por PHP como código de SQL ni cualquier otro.

Algunos de los métodos útiles para esto son:

```php
function secureString($text) {
    return stripslashes(htmlspecialchars($text));
}
```

Peticiones desde JavaScript

Se pueden hacer peticiones a Bases de Datos de PHP desde JavaScript.

Esto se puede lograr con el método fetch desde un archivo JavaScript para realizar una petición a un archivo PHP. Este último tendrá el código necesario para acceder a la Base de Datos e imprimirá por pantalla los datos obtenidos para ser recibidos por el fetch de JavaScript.

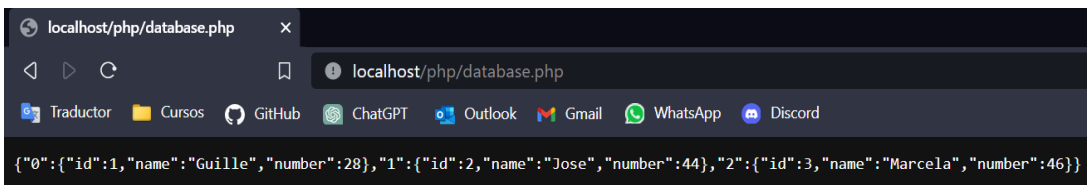
Un método útil para imprimir datos en PHP y puedan ser recogidos por JavaScript es el siguiente:

```php
<?php
   function printResponse ($response) {
        ob_clean();
        if (is_bool($response)) $json = ($response) ? array("response" => true) : array("response" => false);
        elseif (is_numeric($response)) $json = array("response" => $response);
        elseif (empty($response)) $json = array();
        elseif (is_string($response) && !strpos($response, "{")) $json = array("response" => $response);
        elseif (is_string($response)) $json = json_decode($response, true);
        elseif (is_array($response) && array_keys($response) !== range(0, count($response) - 1)) $json = $response;
        elseif (is_array($response)) $json = (object)$response;
        elseif (is_object($response)) $json = $response;
        else $json = array("catch" => "Invalid data format (" . gettype($response) . ")");
        header('Content-Type: application/json');
        echo json_encode((object)$json);
    }
?>
```

Con esta función y la capacidad de leer datos de una tabla, se pueden imprimir dichos datos en una página PHP para poder ser leídos desde un archivo JavaScript.

```php
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "dbtest";
    $tablename = "users";

    try {
        $pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

        $rows = $pdo -> query("SELECT * FROM $tablename");

        $data = array();
        foreach ($rows as $row) {
            for ($i = 0; $i <= count($row) / 2; $i++) unset($row[$i]); # delete array index
            $data[] = $row; # push to array
        }

        printResponse($data);

    } catch (PDOException $e) { printResponse($e -> getMessage()); }
?>
```

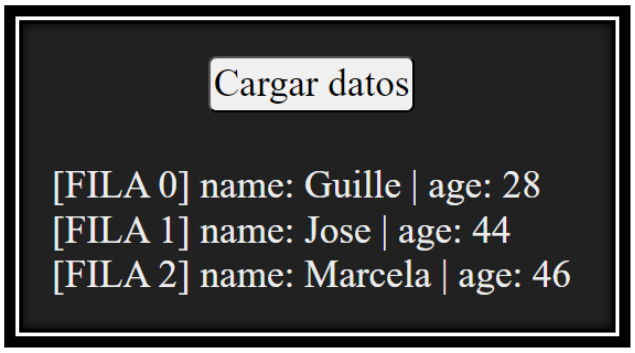
```
MariaDB [dbtest]> SELECT * FROM users;
+----+---------+--------+
| id | name    | number |
+----+---------+--------+
|  1 | Guille  |     28 |
|  2 | Jose    |     44 |
|  3 | Marcela |     46 |
+----+---------+--------+
3 rows in set (0.000 sec)
```



Teniendo el formato JSON correcto en el archivo PHP que leerá la base de datos, se puede leerlo desde JavaScript.

```html
<button id="button">Cargar datos</button>
<div id="data-container"></div>
```

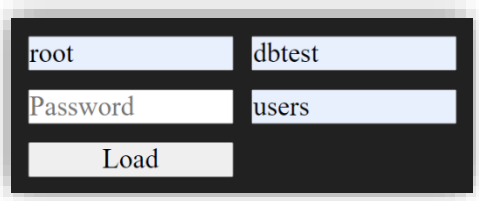
```javascript
document.getElementById("button").addEventListener("click", async () => {

    fetch("database.php").then(response => response.json())
    .then(table => { // table is a json object
        for (const row in table) {
            print(`[FILA ${row}] name: ${table[row]["name"]} | age: ${table[row].number}`);
        }
    })
    .catch(err => console.error(err));

});
```



Peticiones directas en **HTML**

Se puede ejecutar código de PHP directamente en el HTML, sin embargo, para que esto funcione, el archivo debe tener extensión *php*.

```html
<section>
    <form id="form" method="post" action="index.php">
        <input type="text" placeholder="User" name="user">
        <input type="text" placeholder="DataBase name" name="dbname">
        <input type="password" placeholder="Password" name="password">
        <input type="text" placeholder="Table name" name="tablename">
        <input type="submit" value="Load">
    </form>

</section>
```




```php
<section id="data-container">
    <?php
        if ($_POST) try { // Hay parámetros post

            $POST = ($_POST) ? $_POST : json_decode(file_get_contents("php://input"), true);

            $servername = "localhost";
            $username = $POST["user"];
            $password = $POST["password"]; # this is insecure
            $dbname = $POST["dbname"];
            $tablename = $POST["tablename"];

            $pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
            $rows = $pdo -> query("SELECT * FROM $tablename");

            # Columnas de la tabla
            ?> <table> <tr> <td>ID</td> <td>Nombre</td> <td>Edad</td> <td></td> </tr> <?php

            $data = array();
            foreach ($rows as $row) {
                for ($i = 0; $i <= count($row) / 2; $i++) unset($row[$i]); # delete array index
                ?>
                    <tr>
                        <td><?php echo $row['id']?></td>
                        <td><?php echo $row['name']?></td>
                        <td><?php echo $row['number']?></td>
                        <td><a href=<?php print 'deleterow.php?id=' . $row['id']?>>Eliminar</a></td>
                    </tr>
                <?php
            }

        ?> </table> <?php

        } catch(PDOException $e) { ?> </table> <p> <?php echo $e -> getMessage(); ?> </p> <?php }
    ?>

</section>
```

ID	Nombre	Edad		
1	Guille	28	Eliminar	→ localhost/php/deleterow.php?id=1
2	Jose	44	Eliminar	→ localhost/php/deleterow.php?id=2
3	Marcela	46	Eliminar	→ localhost/php/deleterow.php?id=3

Session

Permite mantener información en todas las páginas mientras el navegador esté abierto.

1. **Inicializar la sesión**

```php
<?php
    # Crear sesión
    session_start();

    # Establecer datos
    $_SESSION["usuario"] = "GuilleHern";
    $_SESSION["estatus"] = "logeado";

?>
```

2. **Cargar sesión en otra página**

```php
<?php
    # Cargar sesión
    session_start();

    echo "Usario: " . $_SESSION["usuario"] . "<br>";
    echo "Estatus: " . $_SESSION["estatus"] . "<br>";

?>
```

3. **Cerrar sesión**

   La sesión se puede cerrar automáticamente (al cerrar el navegador) o bien cerrarlo manualmente destruyendo la variable de sesión:

```php
<?php
    session_destroy();
?>
```

API

4:48 develoteca

Include y Require

Archivos

Laravel