

```
<!DOCTYPE html>
<!--
    Declara que el documento utiliza HTML5.

    Esto le indica al navegador que debe interpretar el código
    utilizando el estándar actual de HTML.
-->
<html lang="es">
<!--
    Elemento raíz del documento HTML.
    lang="es" indica que el contenido de la página está escrito
    en español.
-->
<head>
<!--
    <head> contiene información de configuración de la página.
    Esta información normalmente no se muestra directamente
    dentro de la página web.
-->

<meta charset="UTF-8">
<!--
    Define la codificación de caracteres utilizada por la
    página.

    UTF-8 permite mostrar correctamente:
    - Ñ          - Á          - É          - Í          - Ó          - Ú
    - Caracteres especiales
    Sin esta configuración podrían aparecer caracteres
    incorrectos en el navegador.
-->
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<!--
```

Configura la visualización de la página en dispositivos móviles.

`width=device-width`

→ Hace que el ancho de la página se adapte al ancho de la pantalla del dispositivo.

`initial-scale=1.0`

→ Establece el nivel de zoom inicial en 100%.

Esto es importante para crear páginas responsivas.

-->

`<title>FORMULARIO</title>`

`<!--` Define el título de la página.

Este texto aparece normalmente en:

- La pestaña del navegador.

-->

`<link rel="stylesheet" href="style.css">`

`<!--` Vincula el documento HTML con una hoja de estilos CSS.

`href="style.css"`

→ Indica que el archivo de estilos se llama `style.css`.

El CSS será el encargado de controlar aspectos visuales como:

- Colores. - Tamaños. - Márgenes. -

Tipografías.

- Posición de los elementos. - Diseño del formulario.

-->

`</head>`

`<body>`

`<!--` `<body>` contiene todo el contenido visible de la página web.-->

`<form action="" method="post">`

`<!--` `<form>` crea un formulario HTML.

```
action=""
```

→ Indica hacia dónde se enviarán los datos.

Al estar vacío, los datos se envían a la misma página que contiene el formulario.

```
method="post"
```

→ Define el método HTTP utilizado para enviar los datos. POST se utiliza normalmente cuando enviamos información al servidor, por ejemplo:

- Registros.
- Inicio de sesión.
- Formularios.
- Datos que serán almacenados en una base de datos.

Los datos enviados mediante POST se pueden recuperar en PHP utilizando \$_POST. -->

```
<h1>REGÍSTRATE</h1>
```

```
<!--      <h1> representa el encabezado principal de la página.  
En este caso se utiliza como título visual del formulario.
```

```
-->
```

```
<label for="">Nombres:</label>
```

```
<!--
```

```
<label> muestra una etiqueta descriptiva para un campo  
del formulario.
```

```
En este caso indica al usuario que debe introducir  
sus nombres. -->
```

```
<input
```

```
    type="text"        name="nombre"        placeholder="ESCRIBA  
SU NOMBRE" >
```

```
<!--
```

```
<input> crea un campo donde el usuario puede introducir  
información.
```

```
type="text"
```

→ Indica que el campo acepta texto.

```
name="nombre"
```

→ Es MUY IMPORTANTE para PHP.

Este nombre identifica el dato cuando el formulario se envía mediante POST.

En PHP podremos recuperar el valor utilizando:

```
$_POST["nombre"]
```

Por ejemplo, si el usuario escribe:

Juan Pérez

PHP recibirá: `$_POST["nombre"] = "Juan Pérez";`

`placeholder="ESCRIBA SU NOMBRE"`

→ Muestra un texto de ayuda dentro del campo.

Desaparece cuando el usuario comienza a escribir. -->

```
<label for="">Correo:</label>
```

```
<!-- Etiqueta descriptiva para el campo de correo electrónico. -->
```

```
<input type="email" name="email" placeholder="ESCRIBA SU CORREO" >
```

```
<!-- Segundo campo de entrada del formulario.
```

```
type="email"
```

→ Indica que el campo debe contener una dirección de correo electrónico.

Los navegadores pueden realizar una validación básica del formato del correo.

```
name="email"
```

→ Identifica el dato que será enviado al servidor.

En PHP podremos obtenerlo mediante:

```
$_POST["email"]
```

Por ejemplo: `$_POST["email"] = "usuario@gmail.com";`

placeholder

→ Muestra una indicación al usuario sobre qué información debe introducir. -->

```
<input  
    type="submit"    name="register"    id="enviar"    value="REGISTRAR" >
```

<!-- Este input crea el botón que permite enviar el formulario.

```
type="submit"
```

→ Indica que el elemento es un botón de envío.

Al hacer clic, el navegador toma los datos del formulario y los envía utilizando el método definido en <form>.

```
name="register"
```

→ Identifica el botón dentro de los datos enviados.

Esto permite comprobar en PHP si el usuario presionó el botón de registro.

Por ejemplo:

```
if (isset($_POST["register"])) {  
    // Procesar registro  
}
```

```
id="enviar"
```

→ Identificador único del elemento.

Se puede utilizar desde CSS o JavaScript.

Por ejemplo, desde CSS:

```
#enviar {  
    ...  
}  
value="REGISTRAR"  
→ Es el texto que aparecerá visualmente dentro  
del botón. -->
```

</form>

<!-- Finaliza el formulario.

Todo lo que estaba dentro de <form> pertenece al conjunto de datos que puede enviarse al servidor.-->

<?php

// INCLUSIÓN DEL ARCHIVO DE CONEXIÓN

// =====

// include() permite cargar y ejecutar el contenido de otro
// archivo PHP dentro del archivo actual.

//

// conexion.php contiene el código encargado
// de conectarse a la base de datos MySQL.

//

// Por ejemplo:

//

// \$conex = mysqli_connect(...);

//

// Al utilizar include(), podemos reutilizar ese código
// sin tener que escribir nuevamente la conexión.

// Si conexion.php contiene una variable como \$conex,
// dicha variable estará disponible después del include().

include("conexion.php");

```
// INCLUSIÓN DEL ARCHIVO DE REGISTRO
// =====

// Aquí se incluye el archivo registrar.php.
//
// Este archivo probablemente contiene la lógica encargada
// de procesar los datos enviados por el formulario.
//
// Por ejemplo, podría:
//
// 1. Comprobar si se presionó el botón REGISTRAR.
// 2. Obtener el nombre mediante $_POST["nombre"].
// 3. Obtener el correo mediante $_POST["email"].
// 4. Ejecutar una consulta SQL.
// 5. Guardar los datos en la base de datos.
//
// RESUMEN:
// conexion.php
// → Se encarga de conectarse a la base de datos.
//
// registrar.php
// → Se encarga de procesar y guardar los datos.
//
// Este método permite organizar mejor el proyecto.
include("registrar.php");

?>
</body>
<!-- Finaliza el contenido visible de la página.-->

</html>
<!-- Finaliza el documento HTML.-->
```