



Desarrollo Full Stack

Fundación Telefónica Movistar
CURSO DE DESARROLLO FULL STACK.

Módulo 1. HTML y CSS.

Objetivo del curso.

Al final del curso, el alumno deberá ser capaz de planear, diseñar y desarrollar una aplicación web completa, utilizando para programar, el lenguaje PHP y manejando bases de datos mediante el uso de MySQL.

Cuando hablamos de aplicaciones desarrolladas para el mundo web, no alcanza solo con conocer la sintaxis del lenguaje que elijamos para trabajar o aprender a realizar ciertas tareas de forma automática sin analizar el cómo y el porqué de cada cosa que hacemos.

Es por eso por lo que, a lo largo de la cursada, nos vamos a dedicar a probar de diferentes formas cada tema que abordemos, y no hay mejor forma de probar las ideas que llevándolas a cabo.

Vamos a incorporar el análisis como la herramienta más usada en nuestra materia, algo que nos va a servir a lo largo de nuestra vida profesional.

Como analistas de sistemas es primordial entender un problema antes de crearle una solución.

Metodología de trabajo.

Este es un curso en el que vamos a incorporar bastante conocimiento teórico, y necesitamos reforzar dicho conocimiento con una buena cantidad de práctica.

Eso no significa que nosotros no vamos a practicar lo aprendido, de hecho, vamos a ver en acción cada tema que aprendamos, haciendo entre todos y compartiendo paso a paso cada ejercitación.

Para trabajar vamos a necesitar tener instaladas algunas herramientas, y vamos a ver en detalle como instalar y configurar cada una de ellas.

Nos vamos a basar en cinco puntos fundamentales para nuestra labor, y cada uno de esos puntos los vamos a encarar desde cero para que todos podamos nivelar nuestros conocimientos y experiencias y avancemos en equipo.



Una palabra que van a escuchar mucho de mí y la van a ver a menudo en mis apuntes, es EQUIPO, es la clave para tener éxito en lo que encaremos.

Trabajo práctico integrador.

A lo largo del curso vamos a realizar un trabajo práctico que nos permita integrar los conocimientos que vamos viendo durante la cursada.

La entrega de los TP es de carácter obligatoria y necesaria para aprobar la cursada.

Deberán realizarlos de manera individual y deberán hacer presentación del avance en forma regular, con cada tarea que les defina, les pondré también la fecha de presentación.

A su debido momento les daré la consigna de cada TP, pero les adelanto que en el trabajo integrador van a tener que desarrollar una aplicación web funcional que nos permita hacer alta, baja y modificación en una base de datos.

Me interesa que todos trabajen, que todos tengan la oportunidad de llevarse la experiencia de desarrollar colaborando en equipo y que no pasen por el curso como meros espectadores.

¡A programar se ha dicho!

Introducción a las aplicaciones web.

Internet.

El desarrollo de aplicaciones informáticas y de gestión de datos o información fue migrando con el paso de los años, de ser mayormente local (aplicaciones o sistemas que se instalaban en un servidor o en una computadora y se utilizaban solo en dicho equipo) a ser hoy en día casi exclusivamente para trabajos en red.

Si de redes hablamos, lo primero que nos viene a la mente es internet, la red de redes. Saber desarrollar sistemas orientados al uso de internet o redes locales como una intranet es casi imprescindible para los programadores de hoy en día.

El uso de redes trae muchos beneficios a las empresas que se animan a adoptar esta metodología de trabajo, por ejemplo, el trabajo a distancia, los



proyectos colaborativos en tiempo real, el mantenimiento actualizado de sus bases de datos, transacciones financieras sin tiempo de espera y la expansión a nuevos mercados a los cuales resultaba casi imposible acceder sin el uso de estas tecnologías.

En general, llamamos a este tipo de desarrollos que hacen uso de red, aplicaciones web.

Lo más interesante de este tipo de aplicaciones es que nos permiten utilizar y combinar distintos tipos de lenguajes y tecnologías.

Esta particular característica sumada a la continua evolución de las mismas nos permite y al mismo tiempo nos exige, desarrollar sistemas escalables, es decir, que admitan la mayor cantidad de mejoras y actualizaciones con un mínimo requerimiento de modificaciones.

En pocas palabras, nos exige pensar a futuro, cumplir siempre con las necesidades del cliente y anticiparse a las cosas que aún no sabe, pero va a necesitar.

Una red es un conjunto de ordenadores conectados entre sí, mediante un protocolo (conjunto de reglas) con la finalidad de compartir datos y recursos. Según las características de la red, podemos clasificarlas en:

- LAN: Local Area Network.
- WAN: Wide Area Network
- Internet: Red compuesta de redes

Cuando decimos que internet es la red de redes, no es solo una expresión, sino que efectivamente, internet es una red, compuesta de muchas otras redes.

Internet se basa en la arquitectura cliente servidor.

Cliente es el ordenador que realiza una petición, como enviar un correo electrónico, consultar una página web o consultar una base de datos.

Servidor es aquel ordenador que presta un servicio, procesando la solicitud del cliente, gestionando todas las solicitudes (no todas pueden atenderse al mismo tiempo y con la misma prioridad) y generando un contenido para ser entregado a cada cliente, en base a las características de este.



Dominios.

A medida que las redes crecían en tamaño y servicios, surgió la necesidad de interconectarlas para sacarles el mayor provecho, más que nada de parte de grandes empresas y organismos estatales que comenzaban a ver el potencial de tener la mayor cantidad de recursos al alcance de la mano.

Nace entonces el protocolo TCP/IP que exige, entre otras cosas, que exista un número que identifique unívocamente a cada equipo conectado a la red. Dicho número es la dirección IP.

El dominio web es lo que las personas escriben en el navegador, para acceder a un sitio o página de internet.

Dicho dominio está relacionado con una dirección IP, pero es mucho más sencillo de recordar y facilita la indexación (el ordenamiento) de las páginas y su posterior búsqueda.

Un dominio web es esencialmente el equivalente a la dirección física. Se forma a partir de dos elementos, el nombre del dominio y su extensión.

Los registros de nombres de dominios son supervisados por una organización llamada ICANN (Corporación de internet para nombres y números asignados).

Mantienen un registro actualizado de las direcciones IP, los nombres de dominios y los servidores a los cuales apuntan.

DNS.

El DNS (Domain Name System) es un sistema de nombres de dominio encargado de relacionar los nombres de dominio (utilizados por los humanos) con las direcciones IP (utilizadas por las computadoras).

Es un sistema de bases de datos distribuidos en la red con la única intención de traducir el nombre de un dominio a una dirección física.

Hosting.

El hosting es el alojamiento de una página o aplicación web en un servidor de red. Como vamos a ver a lo largo del curso, nuestros desarrollos web se



componen de carpetas y archivos organizados con una determinada estructura y son esos archivos los que deberemos alojar físicamente en un servidor de red para que sea accesible al resto de las personas.

Hay empresas de hosting gratuitas y empresas que ofrecen diferentes servicios a cambio de un abono que por lo general mensual o anual dependiendo del tipo de proyecto que necesitemos desarrollar.

La diferencia entre uno y otro servicio suelen ser las publicidades (en los servicios pagos no se muestra un anuncio a quienes visitan la página a menos que nosotros lo permitamos), las capacidades de almacenamiento, la disponibilidad de diferentes tecnologías y acceso a datos y, sobre todo, el soporte técnico a clientes que suele ser muy limitado en los servicios gratuitos.

El desarrollo por lo general suele ser un trabajo en equipo, y casi siempre, los miembros del equipo suelen estar en diferentes lugares.

Lo usual es trabajar cada parte del proyecto en forma local y organizados por el líder del proyecto, y solo hacer las pruebas en el servidor web una vez que tengamos funcionando lo que nos toca desarrollar.

En general hay dos formas de subir nuestros desarrollos a un servidor web de hosting.

La primera es mediante algún módulo web que permita el intercambio de archivos y la más utilizada es mediante una aplicación FTP para transferencia de archivos (son más rápidas).

Desarrollo de un sitio web.

Habíamos visto que una de las principales características de las aplicaciones web, era la facilidad con que podíamos incorporar a nuestros desarrollos, diferentes tecnologías y lenguajes de programación.

Podemos programar una página web simple, estática, cuyo contenido no cambie, por ejemplo, para promocionar un producto o un servicio, o podemos hacer aplicaciones web dinámicas que además de proveer información estática, permiten la conexión a bases de datos y muchas

veces habilita el acceso y modificación de dicha base, a usuarios cuya identidad fue previamente verificada.



Como vemos, la programación de sitios web dinámicos exige, además de un conocimiento del lenguaje HTML, el desarrollo de hojas de estilo mediante CSS, el uso de diferentes herramientas de estilo y funcionalidades, como puede ser Bootstrap, y la implementación de un lenguaje que facilite el diseño de procesos (la lógica de nuestra aplicación) y el acceso a datos, sin dudas el más difundido e idóneo para este tipo de tareas es PHP.

Y ya que nos vamos a conectar a bases de datos, vamos a necesitar conocer el lenguaje por excelencia para consultarlos, SQL.

Vimos que en internet se utiliza la arquitectura cliente servidor, y PHP (salvo por algún framework reciente) es un lenguaje que se ejecuta del lado del servidor, lo que ayuda mucho a la hora de aplicar políticas de seguridad en nuestro sitio.

Pero en ocasiones necesitamos ejecutar un código breve, alguna función, del lado del cliente, lo que hace a nuestra página aún más dinámica y ahorra algo de tiempo al usuario, y para estos casos, contamos con lenguajes como JavaScript que se ejecutan del lado del cliente.

Este lenguaje también puede ser incluido en nuestras aplicaciones web.

Herramientas de trabajo.

Para el desarrollo de nuestras aplicaciones, vamos a necesitar herramientas que nos faciliten la tarea.

La principal es un editor de texto, si bien nuestras aplicaciones, tanto la parte en HTML como la de PHP las podemos escribir en un editor de texto simple como el que viene incluido en Windows, es una buena práctica hacerlo con un editor que reconozca el lenguaje que estamos utilizando y nos ayude al desarrollo y mantenimiento de nuestro código mediante el indentado (espacio entre la línea de código y el margen del documento) o el uso de diferentes colores para diferentes bloques de código.

Quienes programamos sabemos que nuestro trabajo puede pasar a tener muchas líneas y el poder identificar cada parte del mismo a simple vista, suele resultar muy útil.

Existen varias opciones muy buenas y gratuitas para escribir código, por ejemplo:



- Notepad++ <https://notepad-plus-plus.org/>
- Sublimetext: <https://www.sublimetext.com/>
- Visual Studio Code: <https://code.visualstudio.com/>
- Atom: <https://atom.io/>

Para desarrollar la parte en HTML y CSS de nuestras aplicaciones, no necesitamos más que un editor de texto y un navegador web para ir probando nuestro trabajo.

Para trabajar con PHP vamos a necesitar un servidor (veremos más adelante que función cumple un servidor PHP y por qué lo necesitamos).

Las dos opciones más utilizadas son:

- Wampserver: <http://www.wampserver.com/en/>
- Xampp: <https://www.apachefriends.org/es/index.html>

La parte de bases de datos la tenemos cubierta con la instalación de un servidor PHP ya que ambas distribuciones traen incorporados un gestor de bases de datos MySQL.

Servidores.

Ya vimos que un servidor es un ordenador que proporciona un servicio a un cliente que lo solicita, en el caso de los servidores web, el cliente suele ser el navegador.

Comercialmente contamos con dos tipos de servidores (hay otros no tan difundidos para usos más específicos), los servidores Apache y los IIS.

Servidor Apache.

Es un servidor web HTTP de código abierto para plataformas Unix, Windows, Macintosh y otras a las que suele ser fácilmente adaptable.

Es un servidor multiplataforma, se conecta a diferentes tipos de bases de datos, tiene ayudas y tutoriales muy difundidos, es muy estable en desarrollos exigentes y lo más importante, funciona muy bien con PHP y MySQL también de uso gratuito.



Como desventajas podemos citar la obvia falta de soporte técnico al ser de distribución gratuita salvo que (algo similar a lo que sucede con Linux) compremos algún tipo especial de licencia por distribución que incluya soporte.

Al carecer de soporte es difícil de administrar y configurar (aunque no imposible) y las actualizaciones no son automáticas, deberemos estar atentos a cualquier modificación para implementarlas por nuestra cuenta.

Servidor IIS.

El IIS (Internet Information Services) es un servidor web más un conjunto de servicios orientados al sistema operativo Windows.

Es un servidor seguro y confiable, fácil de usar, viene incluido en Windows, soporta gran cantidad de lenguajes y acceso a diferentes bases de datos.

No es multiplataforma, no es gratuito y el código fuente es propietario, por lo que no podemos modificarlo a gusto para crear nuevas distribuciones.

Motor de base de datos MySQL.

MySQL es un sistema de gestión de bases de datos.

Este tipo de sistemas está definido como un conjunto de programas o herramientas trabajando como un solo programa, dedicados a facilitar el almacenamiento, extracción y modificación de información en una base de datos.

El acceso puede ser mediante la simple escritura de una consulta en lenguaje SQL, mediante el uso de herramientas del tipo visuales o mediante la explotación de datos por medio de un sistema web.

Estos sistemas proveen todas las herramientas necesarias para el desarrollo de nuestras bases de datos siguiendo los protocolos y reglas estándar exigidas para garantizar la integridad y confiabilidad de nuestros datos.

Permite el acceso simultáneo de usuarios para realizar consultas y también permite la gestión de transacciones (movimiento de datos).



Vamos a tener oportunidad de repasar lo que ya saben de SQL cuando diseñemos la base de datos de nuestra aplicación web para el trabajo práctico integrador.

HTML.

Es un lenguaje para el maquetado de páginas web, básicamente nos permite indicarle al navegador la manera de distribuir los elementos dentro de la pantalla, permitiendo en las últimas versiones, adaptar el contenido a los diferentes dispositivos (diseño adaptativo o responsive).

La versión actual del lenguaje es la 5, en la que se incorporaron elementos que van más allá de la simple distribución de nuestro contenido en pantalla, proporcionando diversas utilidades que facilitan el trabajo de los desarrolladores.

Podemos ver las novedades y características de dicha versión en: <https://developer.mozilla.org/es/docs/HTML/HTML5>

Etiquetas.

HTML es un lenguaje de marcado (no es un lenguaje de programación) y sirve para definir la estructura de nuestra aplicación web.

El lenguaje fue creado y es mantenido por una organización sin ánimo de lucro llamada W3C, y es un consorcio de más de 400 empresas, entre ellas, Google, Microsoft y Mozilla.

Este consorcio (conjunto de empresas con un fin común) trabaja continuamente en las definiciones de este lenguaje y de los estándares que conforman la web.

Las empresas que desarrollan los navegadores deberían seguir al pie de la letra este tipo de definiciones, pero como bien sabemos quiénes nos dedicamos a la programación web, esto no se cumple, lo que nos obliga a verificar constantemente nuestros desarrollos para conocer su porcentaje de compatibilidad con los diferentes navegadores del mercado.

En HTML5 tenemos más de 100 elementos que nos permiten crear etiquetas, y ni siquiera los programadores más experimentados, se las conocen todas de memoria, lo importante no es memorizarlas sino saber dónde encontrarlas.



Las etiquetas tienen una apertura y un cierre:

```
<elemento atributo = "valor"> Contenido </elemento>
```

Por ejemplo:

```
<div class="card">
  ... Esto es un contenedor
</div>
```

Existen también los elementos autocontenidos, es decir, no precisan de una etiqueta de cierre, por ejemplo, el elemento IMG.

El atributo y el valor son opcionales y solo se agregan para darle un formato o atributo específico al elemento, por ejemplo, para agregarles estilo.

Las etiquetas pueden anidarse, es decir, podemos hacer que una etiqueta contenga una o más etiquetas dentro.

Cuando anidamos etiquetas una dentro de otra, es importante el orden de cierre, la primera etiqueta en cerrarse deberá ser la última que se abrió.

Ejemplo:

```
<p>UNLZ <strong> Ingenieria </strong></p> <!-- Correcto-->

<p> UNLZ <strong>Ingenieria </p></strong> <!-- Incorrecto-->
```

Estructura de una página web.

Dijimos que las etiquetas pueden anidarse, y eso es válido para casi todas, salvo algunas que deben respetar una estructura básica.

La siguiente es la estructura que nuestra página web (archivo individual dentro de un sitio web) debe respetar:



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>Título de la página</title>
</head>
<body></body>
</html>
```

- La primera línea le indica al navegador que el código que sigue a la misma es HTML5.

Las versiones anteriores requerían que esta primera línea sea bastante más compleja, lo que a menudo, si algún dato no era correcto, confundía al navegador y el resultado visualizado por pantalla, no era el esperado.

Esto se simplificó mucho en la última versión del lenguaje.

- Dentro de las etiquetas HTML van todas las demás etiquetas.

Son las etiquetas que contienen a toda nuestra página web, pudiendo indicarse entre otras cosas, el idioma que vamos a utilizar para nuestro contenido.

- Las etiquetas HEAD se utilizan para contener a las etiquetas con información.

Principalmente a las que brindan información al navegador, a los buscadores y a otras páginas. La información que se coloca dentro del HEAD no se muestra dentro de la página que ve el usuario.



➤ **Atributo CHARSET.**

De esta manera le indicamos al navegador que tipo de caracteres contiene la página, es decir, cuál de todos los juegos de caracteres disponibles vamos a utilizar.

Con el valor UTF-8 vamos a poder crear contenido en la mayoría de los sistemas de escritura como el español, el inglés y el francés.

➤ **La etiqueta TITLE sirve para darle nombre a la página.**

Es el valor que vemos en las pestañas de cada página que muestran los navegadores.

- **La etiqueta BODY agrupa todo el contenido que será visualizado por el usuario dentro de nuestra página.**
- Si leemos con atención la documentación oficial de HTML5, veremos que ya no es obligatorio el uso de las etiquetas HTML, HEAD y BODY, pero por convención es una muy buena práctica hacerlo para asegurarnos un mayor porcentaje de compatibilidad de nuestro desarrollo y los diferentes navegadores y sus versiones que aún están activos entre los usuarios.

Etiquetas básicas.

Las siguientes son las etiquetas básicas y las de uso más común, y podemos ver un listado actualizado de todas las etiquetas en:

<https://www.w3schools.com/tags/default.asp>

- **<p>Contenido</p>**

Sirve para mostrar un párrafo por pantalla. Ejemplo:



`<p>Esto es un párrafo de texto</p>`

- `
`

Representa un salto de línea.

- `<h1>Contenido</h1>`

Se utiliza para hacer encabezados de página. Va del número 1 al número 6 variando el tamaño de la letra para diferenciar diferentes tipos de títulos, subtítulos y encabezados.

```
<h1>Encabezado h1</h1>
<h2>Encabezado h2</h2>
<h3>Encabezado h3</h3>
<h4>Encabezado h4</h4>
<h5>Encabezado h5</h5>
<h6>Encabezado h6</h6>
```

Así se ve por pantalla:

Encabezado h1

Encabezado h2

Encabezado h3

Encabezado h4

Encabezado h5

Encabezado h6



- **Contenido**

Representa una lista de elementos desordenados (no es importante el orden en que se muestren). Se lo suele utilizar combinado con otras etiquetas para hacer un menú de opciones o de navegación.

```
<ul>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
  <li>Elemento 4</li>
</ul>
```

Así se ve por pantalla:

- Elemento 1
- Elemento 2
- Elemento 3
- Elemento 4

- **Contenido**

Representa una lista de elementos ordenados, asigna un número de orden a cada uno. El número asignado se puede configurar.

```
<ol>
  <li>Opción Uno</li>
  <li>Opción Dos</li>
  <li>Opción Tres</li>
</ol>

<ol start="50">
  <li>Opción Uno</li>
  <li>Opción Dos</li>
  <li>Opción Tres</li>
</ol>
```



Con el código anterior lo que hicimos fue armar dos listas ordenadas, la primera con un orden por defecto, es decir, asigna el número 1 al primer elemento, el número 2 al segundo y así con todos los elementos que contenga la lista.

A la segunda lista le indicamos el número que debe tener el primer elemento y a partir de dicho número comenzara a contar.

Así se ven ambas listas:

1. Opción Uno
2. Opción Dos
3. Opción Tres

50. Opción Uno
51. Opción Dos
52. Opción Tres

- `Contenido`

Se utiliza para definir un elemento de una lista.

Menú de navegación.

La etiqueta que nos permite crear un menú simple de navegación es `<nav>`.

Dicho menú deberá contener los enlaces a las páginas o archivos que a los que deberá acceder el usuario.

Estos enlaces se hacen con la etiqueta `Contenido`

```
<nav>  
  <a href="#">Link 1</a>  
  <a href="#">Link 2</a>  
  <a href="#">Link 3</a>  
</nav>
```



[Link 1](#) [Link 2](#) [Link 3](#)

Si necesitamos ir armando el menú de navegación, pero todavía no tenemos creados los archivos, en lugar de escribir la ruta pondremos el carácter #.

Donde dice Link va el texto que será visto por el usuario.

Tablas.

Vamos a ver la estructura básica de una tabla, cuáles son sus elementos y cuando veamos conexión a datos con PHP veremos en detalle como cargarle los datos que extraemos de una tabla en MySQL.

Una tabla sencilla debe tener al menos los siguientes elementos:

- `<table>` `</table>`

Son las etiquetas que encierran todo el contenido de una tabla.

- `<tr>` `</tr>`

Dentro de estas etiquetas se coloca el contenido de una fila. La primera fila es casi siempre el nombre que tiene cada columna.

- `<td>` `</td>`

Estas son las etiquetas que contienen el valor de cada celda.

Una tabla de ejemplo:



```
<table>
  <tr>
    <th>Columna 1</th>
    <th>Columna 2</th>
    <th>Columna 3</th>
  </tr>
  <tr>
    <td>Celda 1-1</td>
    <td>Celda 1-2</td>
    <td>Celda 1-3</td>
  </tr>
  <tr>
    <td>Celda 2-1</td>
    <td>Celda 2-2</td>
    <td>Celda 2-3</td>
  </tr>
</table>
```

Los títulos de las columnas los podemos encerrar entre etiquetas TD o podemos, como en nuestro ejemplo, ponerle etiquetas TH para poner el contenido en negrita.

Así se ve por pantalla:

Columna 1 Columna 2 Columna 3

Celda 1-1 Celda 1-2 Celda 1-3

Celda 2-1 Celda 2-2 Celda 2-3

Las tablas admiten diferentes tipos de atributos, por ejemplo, podemos definir el tipo y tamaño de la letra, el color de fondo y el tipo de borde.

```
<table border="1">
```



Columna 1	Columna 2	Columna 3
Celda 1-1	Celda 1-2	Celda 1-3
Celda 2-1	Celda 2-2	Celda 2-3

Agregamos color de fondo y texto:

```
<table border="1">
  <tr style="background:#003255;color:white;">
```

Columna 1	Columna 2	Columna 3
Celda 1-1	Celda 1-2	Celda 1-3
Celda 2-1	Celda 2-2	Celda 2-3

Formularios.

Los formularios sirven para recolectar información del usuario y enviarla al servidor para ser procesada, pudiendo, en caso de trabajar con bases de datos, guardarla en la misma.

Nos permite una interacción básica con el usuario, por ejemplo, para crear formularios de contacto, listas de compras, ingreso a un sistema, etc.

Estos datos, al ser enviados al servidor, son procesados por el lenguaje que utilicemos para la lógica de nuestra aplicación, por ejemplo, PHP.

La etiqueta para crear un formulario es <FORM>, veamos un ejemplo simple de formulario y analicemos el código.

Como primer paso, y para ir acostumbrándonos a la sintaxis, armamos una plantilla básica de HTML:



```
<!DOCTYPE html>

<html>

  <head>

    <title>Formularios</title>

  </head>

  <body>

    <h1>Elementos básicos de un formulario.</h1>

  </body>

</html>
```

Luego, empezamos a armar un formulario dentro del body:

```
<form action="pagina_destino.php" method="get">

</form>
```

La etiqueta FORM tiene dos atributos muy importantes:

- **ACTION:** indica la página donde serán enviados los datos.
- **METHOD:** indica el método mediante el cual enviamos esos datos.

El funcionamiento de cada uno de estos atributos los veremos en detalle cuando trabajemos con PHP, por ahora nos vamos a centrar solamente en el maquetado del formulario.

La diferencia entre los métodos consiste en la manera de enviar los datos.

El método GET los envía usando la URL (quedan visibles para el usuario y para los que puedan acceder al historial de búsqueda del navegador) y por lo general lo utilizamos solo para el envío de datos que no sean secretos, que no nos importe que sean públicos, por ejemplo, cuando consultamos



mediante un formulario, el menú de una casa de comidas o los precios de diferentes artículos en un supermercado.

En cambio, POST no muestra los datos en la URL, y su uso se reserva para el envío de datos sensibles como por ejemplo datos personales o contraseñas.

También existe diferencia en la forma en que podemos recuperar esos datos, veremos cuando trabajemos con PHP que el método GET ni siquiera precisa del armado de un formulario, se pueden enviar datos por ejemplo mediante una etiqueta <a>.

Elementos básicos de un formulario.

INPUT.

La etiqueta INPUT, en su formato más básico, permite al usuario ingresar cualquier tipo de dato.

```
<form action="pagina_destino.php" method="get">  
    <input>  
</form>
```

Estos datos que el usuario ingresa necesitan ser validados, ya sea por seguridad, en cuyo caso recurriremos más adelante a PHP, pero también para brindarle una mejor experiencia al usuario, ya que hay validaciones básicas que HTML nos permite hacer si le utilizamos algunos atributos específicos de la etiqueta INPUT.

El atributo más usado para esta etiqueta es TYPE, que nos permite especificar qué tipo de dato queremos que sea ingresado.

El valor por defecto es TEXT, pero disponemos de varios tipos, veamos algunos ejemplos:



```
<form action="pagina_destino.php" method="get">

  Tipo Texto: <input type="text"><br><br>

  Tipo Botón: <input type="button" value="Aceptar"><br><br>

  Tipo CheckBox: <input type="checkbox"><br><br>

  Tipo Date: <input type="date"><br><br>

  Tipo File: <input type="file"><br><br>

  Tipo PassWord: <input type="password"><br><br>

  Tipo Radio: <input type="radio"><br><br>

  Tipo Range: <input type="range"><br><br>

  Tipo Reset: <input type="reset"><br><br>


</form>
```

Elementos básicos de un formulario.

Tipo Texto:

Tipo Botón:

Tipo CheckBox: ☐

Tipo Date: 

Tipo File: Ningún archivo seleccionado

Tipo PassWord:

Tipo Radio: ☐

Tipo Range:

Tipo Reset:



El más común es el tipo TEXT, que es el valor por defecto si no especificamos otro, y se usa para que el usuario escriba dentro.

El tipo BUTTON genera un botón sin un uso específico.

El tipo CHECKBOX (se suele usar en grupo) genera una casilla para que el usuario tilde una opción seleccionada.

El tipo DATE (solo para HTML5) despliega un calendario para poder seleccionar una fecha y olvidarnos del formato en que el usuario lo ingrese, siempre va a tener el mismo formato.

El tipo FILE permite seleccionar un archivo de nuestro equipo para subir a la web o enviarlo por medio de un formulario.

El tipo PASSWORD oculta lo que el usuario está escribiendo.

El tipo RADIO es similar al CHECKBOX y genera un selector similar.

LABEL

Esta etiqueta permite asignarle un texto a un determinado INPUT.

Por ejemplo, si queremos hacer un formulario simple para que el usuario cargue su nombre y apellido, hacemos lo siguiente:

```
<form action="pagina_destino.php" method="get">
  <label for="nombre">Ingrese su nombre</label>
  <input type="text" id="nombre"><br><br>
  <label for="apellido">Ingrese su apellido</label>
  <input type="text" id="apellido">
</form>
```



Elementos básicos de un formulario.

Ingrese su nombre

Ingrese su apellido

En la etiqueta LABEL colocamos el atributo FOR, donde indicamos a que etiqueta INPUT hace referencia.

Vemos incluso que, si pulsamos sobre alguno de los LABEL, la página hará foco sobre la etiqueta INPUT que le corresponde.

Podemos hacer la sintaxis un poco más simple si colocamos el INPUT dentro de la etiqueta LABEL, ahorrándonos el atributo FOR:

```
<label>Ingrese su apellido  
  <input type="text">  
</label>
```

Como vimos al inicio de la clase, los datos que se cargan en un formulario son para ser enviados a un servidor que los procese.

Este servidor deberá saber que dato enviado corresponde a que elemento del formulario, y para ello, utilizamos el atributo NAME del input, para que el servidor pueda identificar cada elemento que enviamos en el formulario.

```
<form action="pagina_destino.php" method="get">  
  <label>Ingrese el producto  
    <input name="producto">  
  </label>  
</form>
```



Botón Enviar.

Si bien tenemos la etiqueta `BUTTON` para crear elementos clickeables, lo usual en un formulario, es usar una etiqueta `INPUT` para enviar los datos, de manera similar al ejemplo del `RESET` que vimos.

```
<form>

  <label>Ingrese su nombre

    <input name="nombre">

  </label>

  <br><br>

  <label>Ingrese su apellido

    <input name="apellido">

  </label>

  <br><br>

  <input type="submit">

</form>
```

Elementos básicos de un formulario.

Ingrese su nombre

Ingrese su apellido

Si vemos este formulario por pantalla, veremos que el botón aparece con la leyenda “Enviar” por defecto, pero podemos asignarle la que necesitamos que tenga, mediante el atributo `VALUE`.



```
<input type="submit" value="Aceptar">
```

Elementos básicos de un formulario.

Ingrese su nombre

Ingrese su apellido

Aceptar

Este atributo VALUE también podemos utilizarlo en otras etiquetas, por ejemplo, si el usuario quisiera modificar sus datos, previamente cargados en una base de datos, podemos mostrarle lo que tenemos para que el solo modifique lo que necesita y no tenga que cargarlos todos nuevamente.

```
<form>

  <label>Ingrese su nombre
    <input name="nombre" value="Guillermo">
  </label>

  <br><br>

  <label>Ingrese su apellido
    <input name="apellido" value="Alfaro">
  </label>

  <br><br>

  <input type="submit" value="Aceptar">

</form>
```



Elementos básicos de un formulario.

Ingrese su nombre

Ingrese su apellido

Cuando comencemos con PHP, vamos a volver sobre el tema de formularios, donde haremos desde cero un formulario completo en el que enviaremos datos al servidor y vamos a ver como recibimos esos datos y cómo podemos hacer uso de ellos.

Botones.

La etiqueta `BUTTON` sirve para crear elementos clickeables (botones) que se pueden utilizar en formularios o en cualquier parte de nuestra página.

De forma predeterminada, los botones se muestran siempre con un estilo similar en todas las plataformas, pero este estilo puede modificarse muy fácilmente.

Este elemento en particular además de los clásicos atributos tiene también la posibilidad de manejar diferentes tipos de eventos.

Botón simple: `<button>Aceptar</button>`

`

`

Botón deshabilitado: `<button disabled>Aceptar</button>`

`

`

Botón con evento: `<button onclick="nombre_de_funcion">Aceptar</butto`

`n>`



Botón simple:

Botón deshabilitado:

Botón con evento:

El botón con el atributo ONCLICK nos permite especificar que, al presionar el botón, se invocará inmediatamente una función, por lo general en JavaScript.

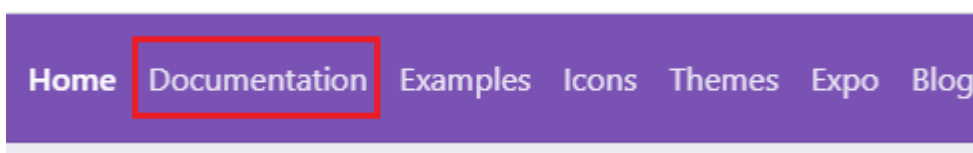
Introducción a Bootstrap.

Es un FRAMEWORK que nos posibilita la creación de sitios o aplicaciones web de manera sencilla, sin complicarnos con el diseño de estilos.

Es un conjunto de herramientas CSS y JS que podemos incorporar a nuestras aplicaciones, ya sea tomando los archivos necesarios desde la web o descargándolos a nuestro servidor para tomarlos en forma local.

Desde la página oficial <https://getbootstrap.com/> podemos descargar todas las versiones o ver un ejemplo de plantilla.

Como primer paso, ingresamos a la web de Bootstrap y vamos al apartado de documentación.



Dentro de la documentación, buscamos el título STARTER TEMPLATE.

Starter template

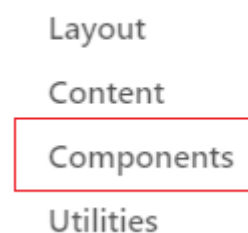
Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors. Put it all together and your pages should look like this:



Nos copiamos el código que aparece y utilizamos esa plantilla como base para practicar el uso de los elementos de Bootstrap.

Lo siguiente es ir, mediante el menú de navegación de la izquierda, al apartado de componentes.

Una vez que ingresemos, el menú nos muestra una lista de los componentes que podemos incorporar a nuestros desarrollos.



Veamos algunos ejemplos de uso, comparando los elementos que vimos hasta ahora con el estilo que nos aporta el framework.

```
<button>Aceptar</button>

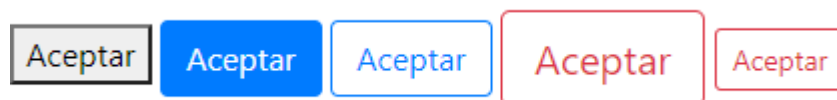
<button class="btn btn-primary">Aceptar</button>

<button class="btn btn-outline-primary">Aceptar</button>

<button class="btn btn-outline-danger btn-lg">Aceptar</button>

<button class="btn btn-outline-danger btn-sm">Aceptar</button>
```

El primer ejemplo, es un botón simple, sin aplicar ningún estilo de Bootstrap, los demás tienen algún tipo de estilo aplicado.



***** Hacer uso de la plantilla para probar diferentes elementos.**

