

Trabajo integrador.

El objetivo del presente trabajo es implementar de manera práctica, todos los puntos teóricos vistos en el curso y agregar algunas variaciones de dichos temas.

Al acabar el trabajo, el alumno habrá desarrollado, una aplicación web que contendrá las principales funcionalidades que todo sistema web debería tener, principalmente acceso a administradores, gestión de accesos y perfiles utilizando sesiones y resguardo de datos con base de datos.

Para una mejor comprensión y para facilitar el trabajo del alumno, dividiremos el desarrollo en actividades.

Actividad 1. Armado de la plantilla base.

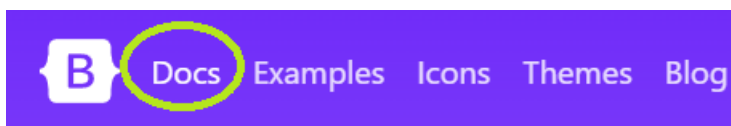
El maquetado del proyecto lo vamos a hacer aplicando los estilos de Bootstrap, por lo que comenzaremos armando nuestra carpeta de trabajo, descargando los archivos de Bootstrap y armando nuestra plantilla base.

Entramos al sitio de Bootstrap y descargamos los archivos de la versión 5. Estos archivos tienen todas las funciones de JavaScript y los archivos de estilo necesarios para implementar este FRAMEWORK por completo en nuestro proyecto.

Si bien solo necesitaremos dos archivos, debido a que son muy livianos y que no nos generan inconvenientes, vamos a armar nuestra carpeta de trabajo base con todo Bootstrap dejando así todo listo para futuros proyectos.

Link: <https://getbootstrap.com/>

Desde la barra de navegación, accedemos a la opción DOCS.



Veremos aparecer un menú lateral de opciones, donde seleccionaremos la opción de descarga:

Getting started

Introduction

Download

Contents

Luego, estando en la sección de descargas, seleccionaremos la primer opción de descarga que se nos ofrece, y pulsaremos sobre el botón DOWNLOAD.

Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.3.3** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

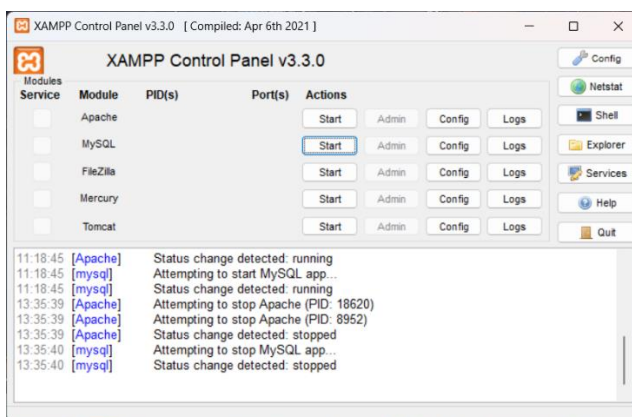
Download

Una vez hecho esto, veremos en la sección de descargas de nuestra PC, un archivo comprimido que es el que tiene todo lo necesario para trabajar con Bootstrap.

Vamos ahora a crear una carpeta de trabajo donde iremos realizando todas las actividades de nuestro proyecto.

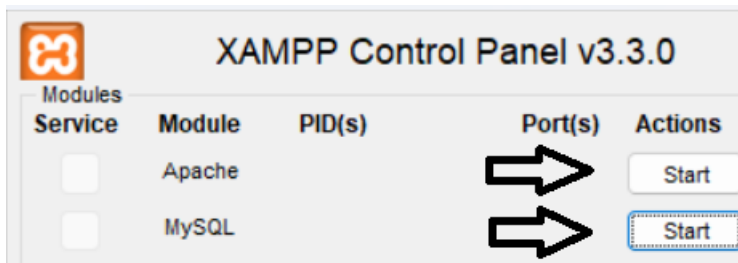
Como todo lo realizaremos con PHP, lo primero es iniciar el servidor de trabajo, para este documento, trabajamos con XAMPP, pero cualquier otra versión comercial de PHP y MySQL sirve para el presente trabajo.

Ejecutamos el panel de control.

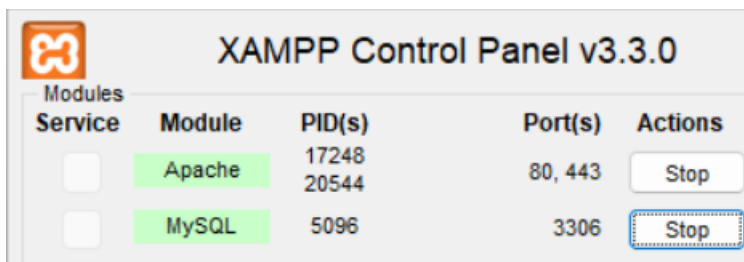


Para esta parte del proyecto, solo vamos a necesitar el servidor Apache, pero no está de mas verificar el funcionamiento del servidor de base de datos, para ir previendo el buen funcionamiento para futuras tareas.

Procedemos a iniciar ambos servidores, pulsando sobre las correspondientes opciones de inicio.



Pulsando en ambos botones START, veremos como el nombre de cada servidor (Apache / MySQL) se remarkan en amarillo y si todo funciona bien, quedan remarcados en color verde.



*** Error en los puertos.

En algunas ocasiones, nos encontramos con el error de puerto ocupado. Existen muchos tutoriales de como hacer el cambio de puerto, les dejo un par de videos, uno para cada servidor, que pueden ser de utilidad.

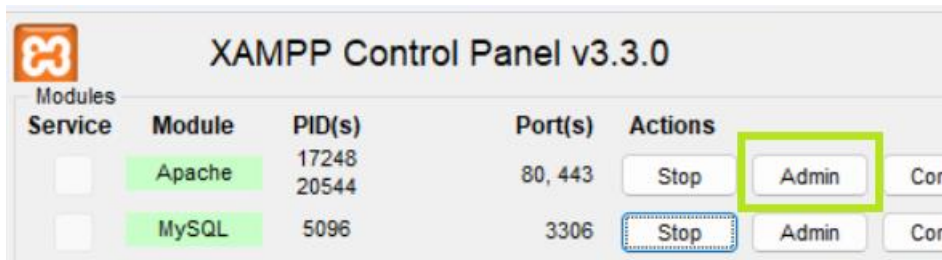
Cambiar puertos Apache.

https://www.youtube.com/watch?v=MPDchC9qtug&ab_channel=JuanMart%C3%ADn

Cambiar puertos MySQL.

https://www.youtube.com/watch?v=9nl0_vEQcGI&ab_channel=Byspel-Iv%C3%A1nL.

Para verificar si el servidor inició correctamente, además de ver el nombre de este remarcado en color verde, vamos a acceder a la página principal de la aplicación web que viene con la instalación, pulsando sobre el botón ADMIN del servidor Apache.



Si el servidor se inició sin errores, veremos la siguiente pantalla abrirse en nuestro navegador:

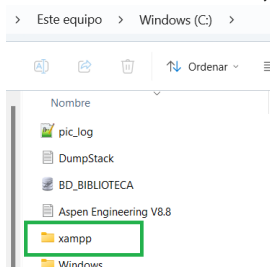


Welcome to XAMPP for Windows 8.1.10

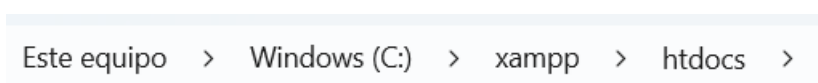
A partir de acá, ya estamos listos para iniciar el armado de nuestra carpeta de trabajo.

Recordemos que siempre que trabajemos con PHP, nuestros proyectos deberán armarse en la carpeta de trabajo del servidor, para el caso de XAMPP, dicha carpeta es HTDOCS.

Si no hemos cambiado la ruta de instalación de XAMPP que viene por defecto en su instalador, lo veremos instalado en la raíz de nuestro disco.



Dentro de la carpeta principal de la instalación, vamos a encontrar nuestra carpeta de trabajo, HTDOCS.



Dentro de esta carpeta vamos a colocar todos nuestros proyectos. Es importante que cada proyecto que vayamos a desarrollar, lo coloquemos dentro de su propia y exclusiva carpeta, para que podamos exportarlo o compartirlo de manera sencilla y sin complicaciones.


En mi caso, voy a crear dentro de HTDOCS, una carpeta llamada PROYECTO donde iré colocando todos los archivos necesarios para el trabajo integrador.


Lo ideal, dentro de las buenas prácticas de programación web, es crear nuestras carpetas y archivos de trabajo con nombres sencillos, en minúscula y sin utilizar caracteres especiales ni espacios en blanco. Consideremos que estos nombres de carpetas y archivos van a terminar formando parte de la URL y debemos mantenerla siempre lo mas simple posible para evitar errores no solo durante el desarrollo del trabajo, sino también para facilitar el uso de nuestro trabajo a los futuros usuarios.

Este equipo > Windows (C:) > xampp > htdocs > proyecto

Con nuestra carpeta debidamente creada, vamos a comenzar nuestro trabajo en ella, colocando el archivo comprimido que descargamos desde la página de Bootstrap y lo descomprimimos.

Nombre

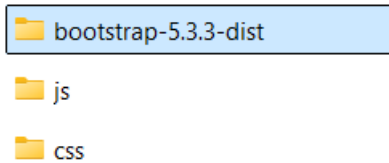
 bootstrap-5.3.3-dist

 bootstrap-5.3.3-dist

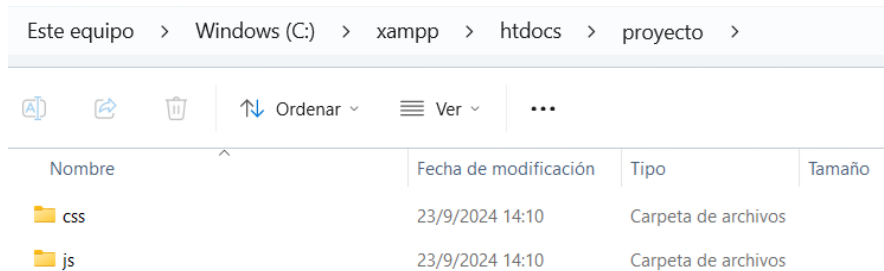
Ya podemos eliminar el archivo comprimido porque no lo vamos a necesitar más.

Vemos que ahora tenemos una carpeta con el mismo nombre del archivo comprimido que descargamos. Dentro de esta carpeta vamos a encontrar otras dos carpetas, CSS y JS que contienen todos los archivos de Bootstrap necesarios para nuestro proyecto.

Vamos a copiar dichas carpetas y las vamos a pegar fuera de la carpeta en la que están, y dentro de nuestra carpeta de trabajo, de esta manera:



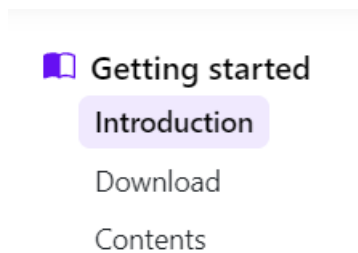
Ahora ya podemos eliminar la carpeta de Bootstrap y quedarnos solo con las carpetas CSS y JS dentro de nuestra carpeta de trabajo.



Ya tenemos nuestros archivos con los estilos y las funciones de JavaScript necesarias para poder implementar Bootstrap.

Comenzaremos con el armado del maquetado base de nuestro proyecto, para lo cual deberemos nuevamente ingresar a la página de Bootstrap.

En el menú de opciones lateral, iremos ahora a la opción INTRODUCTION



Esto nos llevará a la sección de inicio, donde tendremos diferentes opciones para comenzar a trabajar con Bootstrap, dentro de dichas opciones, vamos a utilizar la segunda, que nos ofrece un ejemplo de maquetado web ya armado para el uso de los archivos de nuestro FRAMEWORK.

2. **Include Bootstrap's CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our [CDN links](#).

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" re
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.
  </body>
</html>
```

Este maquetado HTML es el que vamos a utilizar para iniciar nuestra plantilla base, para lo cual, crearemos con nuestro editor favorito (en mi caso VS CODE) un archivo llamado PLANTILLA.PHP.

Dentro de este archivo, copiaremos el maquetado sugerido por Bootstrap en la opción 2 (el de la figura anterior) y lo pegaremos en este archivo.

2. **Include Bootstrap's CSS and JS.** Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our [CDN links](#).

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width. initial-scale=1">
```

Copy to clipboard

Pegamos el código copiado, en nuestro archivo PLANTILLA.PHP

```
plantilla.php •
C: > xampp > htdocs > proyecto > 🐘 plantilla.php > ...
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">
6      <title>Bootstrap demo</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" i
8    </head>
9    <body>
10     <h1>Hello, world!</h1>
11     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="s
12   </body>
13 </html>
14
```

Guardamos los cambios y ya tenemos nuestro maquetado base para empezar a programar.

Vamos a modificar algunas de las líneas de nuestra plantilla base, por ejemplo, el valor del atributo LANG, el TITLE y las rutas de acceso a los archivos CSS y JS ya que vamos a cambiar la búsqueda en internet de dichos archivos para utilizarlos localmente, desde las descargas que hicimos.

Así deberían quedar dichas modificaciones realizadas en las líneas 2,6,7 y 11.

```
plantilla.php X
C: > xampp > htdocs > proyecto > 🐘 plantilla.php > ...
1  <!doctype html>
2  <html lang="es">
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>Plantilla Base</title>
7      <link href="css/bootstrap.min.css" rel="stylesheet">
8    </head>
9    <body>
10     <h1>Hello, world!</h1>
11     <script src="js/bootstrap.bundle.min.js"></script>
12   </body>
13 </html>
```

Vamos a eliminar la línea 10 ya que agregaremos algún título mejor elaborado desde Bootstrap.

La distribución de los elementos no tiene que ser exactamente como se ve en este instructivo, si bien los elementos a agregar deberían ser todos los mismos, cada alumno los puede organizar como mejor le parezca.

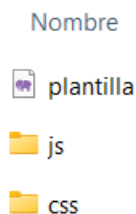
Comencemos con una barra de navegación que contenga en principio las principales actividades que vamos a realizar y con el paso de las clases, iremos agregando las demás.

A modo de sugerencia, vamos a colocar una barra de navegación que se quede fija en la parte superior para facilitar a los usuarios que siempre este disponible cuando necesite usarla.

En cuanto al BODY, voy a optar por un contenedor que aplique un margen importante a izquierda y a derecha de la pantalla ya que no vamos a comenzar agregando gráficos y datos, sino que comenzaremos con las principales funcionalidades.

Es importante pensar nuestro desarrollo como escalable, es decir, a todo lo que vayamos desarrollando, le iremos agregando con el tiempo algunas modificaciones generalmente relacionadas con estilos visuales.

Con estas modificaciones en nuestro maquetado base, nuestra aplicación debería estar así:



Ya tenemos en este punto, armada la estructura base de nuestro proyecto, vamos a concentrarnos ahora en agregar a nuestra plantilla, los primeros elementos que serán los que heredarán los archivos que hagamos a partir de la mencionada plantilla.

AGREGAMOS UNA BARRA DE NAVEGACIÓN.

Vimos que lo interesante de usar Bootstrap es que podemos tomar de manera muy sencilla cualquier elemento HTML y utilizarlo en nuestro proyecto.

Vamos a dar un pasito mas y vamos a crear cada uno de estos con un paso a paso que nos permita entre otras cosas, ir entendiendo e incorporando el por que de cada clase CSS que utilicemos.

La barra de navegación la vamos a colocar en la parte superior de nuestro maquetado, y vamos a configurarla para que quede fija, algo que los alumnos podrán modificar muy fácilmente si lo que quieren es que la barra no esté fija en una determinada posición.

Como la barra de navegación es algo que los usuarios verán, vamos a colocarla dentro del BODY.

Comenzamos agregando la etiqueta NAV que es la que corresponde usar para este elemento.

Para mayor comprensión del código a utilizar, lo voy a desarrollar en un archivo aparte y luego lo voy a colocar en la plantilla que estamos armando.

A la etiqueta NAV le vamos a colocar las siguientes clases CSS:

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">  
  
</nav>
```

¿Qué hace cada una de estas clases?

.navbar: Esta clase se utiliza para crear una barra de navegación. Es el contenedor principal que define el estilo y el comportamiento de la barra de navegación. Incluye soporte para el branding, la navegación y otros elementos como formularios y botones.

.navbar-expand-lg: Esta clase se usa para hacer que la barra de navegación sea responsive. La parte expand-lg indica que la barra de navegación se expandirá (mostrará todos sus elementos) cuando la pantalla sea de tamaño grande (lg) o mayor. En pantallas más pequeñas, los elementos de la barra de navegación se colapsarán en un menú desplegable.

.bg-body-tertiary: Esta clase aplica un color de fondo específico a la barra de navegación. En Bootstrap 5, las clases de fondo como bg-body-tertiary utilizan variables CSS para definir colores que se integran con el esquema de color general del sitio

El primer DIV que agregamos (líneas 2 y 11, es el que va a guardar todo el contenido de la barra, y le vamos a agregar la clase que hace uso de todo el ancho disponible CONTAINER FLUID).

Recordar: al BODY le aplicamos la clase CONTAINER, lo que aplica margen a izquierda y derecha de nuestro maquetado, y a la barra de navegación le aplicamos la clase CONTAINER-FLUID, para que tome todo el ancho de que dispone el BODY.

Vamos a iniciar agregando de a uno los enlaces de nuestro segundo DIV, el que habíamos colocado inmediatamente debajo del botón.
Primero le agregamos las clases de estilo que vamos a utilizar:

```
<!-- Segundo DIV -->
<div class="collapse navbar-collapse" id="navbarSupportedContent">

</div>
```

Todos los enlaces los vamos a armar usando listas, que como ya vimos, son etiquetas LI dentro de etiquetas UL.
Comencemos con algunos enlaces:

```
<!-- Segundo DIV -->
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
|   <li class="nav-item">
|       <a class="nav-link active" aria-current="page" href="#">Inicio</a>
|   </li>
</ul>
</div>
```

Vamos a agregar un par de enlaces más, para usar en las primeras prácticas de nuestro proyecto.

```
<!-- Segundo DIV -->
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item">
      <a class="nav-link active" aria-current="page" href="#">Inicio</a>
    </li>
    <li class="nav-item">
      <a class="nav-link active" href="#">Formularios</a>
    </li>
    <li class="nav-item">
      <a class="nav-link active" href="#">ABM</a>
    </li>
  </ul>
</div>
```

Vamos a agregar un par de clases para darle estilo DARK a nuestro menú. La etiqueta NAV nos debería quedar así:

```
<nav class="navbar navbar-expand-lg bg-body-tertiary navbar-dark bg-dark sticky-top">
```

Vamos a agregar ahora, a nuestra barra de navegación, un enlace que permita acceder a sub opciones, es decir, una opción que contenga mas de una sub opción.

Esto lo hacemos también con elementos UL/LI y las correspondientes clases de Bootstrap.

Agregamos lo siguiente debajo del último enlace agregado, el de la página ABM.

```
<!-- Opción con Sub opciones -->
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle active" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
    Opciones
  </a>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Sub-opción A</a></li>
    <li><a class="dropdown-item" href="#">Sub-opción B</a></li>
    <li><hr class="dropdown-divider"></li>
    <li><a class="dropdown-item" href="#">Sub-opción C</a></li>
  </ul>
</li>
```

Para finalizar, vamos a maquetar un pequeño formulario dentro de la barra de navegación, a la que mas adelante le programaremos la función de búsqueda de contenido dentro de nuestra página.

Debajo de la última etiqueta de cierre /UL que colocamos recién, vamos a agregar el siguiente formulario:

```
<form class="d-flex" role="search">
  <input class="form-control me-2" type="search" placeholder="Buscar contenido" aria-label="Search">
  <button class="btn btn-outline-success" type="submit">Buscar</button>
</form>
```

COLOCAMOS LA BARRA DE NAVEGACIÓN EN EL BODY.

Apenas comienza nuestro BODY, vamos a colocar el código que acabamos de escribir, en mi caso lo arme en un archivo aparte, pero si lo hicieron directamente dentro de la plantilla, está correcto.

```
<body class="container">
  <!-- Barra de navegación -->
  <nav class="navbar navbar-expand-lg bg-body-tertiary navbar">
    <!-- Primer DIV -->
    <div class="container-fluid">
      <a class="navbar-brand" href="http://www.guille">
      <button class="navbar-toggler" type="button" da
        aria-controls="navbarSupportedContent" aria
        <span class="navbar-toggler-icon"></span>
      </button>
      <!-- Segundo DIV -->
      <div class="collapse navbar-collapse" id="navba
        <ul class="navbar-nav me-auto mb-2 mb-lg-0"
          <li class="nav-item">
            <a class="nav-link active" aria-cur
```

Vamos a guardar todos los cambios y ver nuestro proyecto hasta el momento.

Para esto, desde el navegador de internet, vamos a editar la URL.

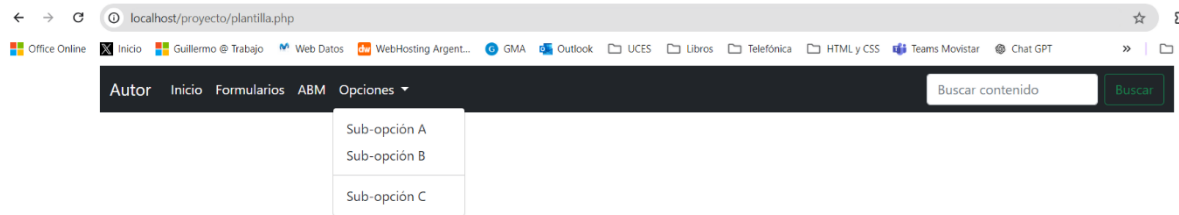
Así se ve ahora:

<http://localhost/dashboard/>

Recordamos que nuestro localhost es la contraparte de nuestra carpeta HTDOCS, por lo que a partir del localhost, vamos a armar la ruta para que el navegador de internet encuentre el archivo que queremos visualizar, para el ejemplo de este apunte, deberíamos escribir lo siguiente:

<http://localhost/proyecto/plantilla.php>

Y esto es lo que deberíamos ver, en mi caso voy a desplegar el menú con sub opciones para verificar que funciona:




Si llegaron hasta acá con el mismo resultado, podemos dar por terminada la actividad número uno.

Actividad 2. Barra de navegación en una función PHP.

Nuestra plantilla base va a servir de punto de partida para la creación de muchos de los archivos que vamos a agregar a nuestro proyecto, por lo que nos sería de utilidad que contenga la mayor cantidad de elementos comunes de nuestro desarrollo, para evitarnos la complicación de tener código reutilizable en más de un archivo.

Habiendo visto como incluir archivos y como declarar funciones, veremos como aprovechar este conocimiento, para crear un archivo externo que contenga a nuestra barra de navegación.

Como primer paso, vamos a crear en nuestro desarrollo, una carpeta llamada INC donde iremos guardando los archivos a incluir, que sean de extensión PHP.

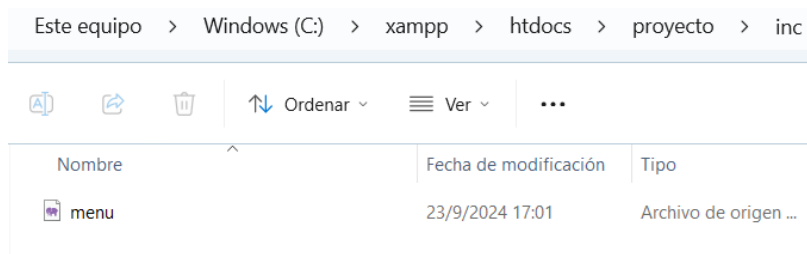
 plantilla

 js

 inc

 css

Dentro de la carpeta INC, vamos a crear un archivo, con extensión PHP, en el que vamos a crear una función llamada MENU.



Dentro del archivo, que va a contener solo código PHP, comenzaremos colocando las etiquetas obligatorias para dicho lenguaje y vamos a declarar la función.

```
<?php
// Declaramos la función
function menu(){

}
?>
```

Dentro de esta función PHP, vamos a colocar el maquetado de la barra de navegación que hicimos en la actividad uno, pero para que el interprete de PHP no se confunda, vamos a partir en dos partes nuestro código, y entre estas dos partes, pegaremos el maquetado, así, de esta manera, cuando el interprete de PHP lea nuestro código, no va a encontrar ningún error de sintaxis.

Vamos con la primer parte, partimos la función en dos partes.

```
<?php
// Declaramos la función
function menu(){
?>

<!--
Este espacio, al no estar entre etiquetas
PHP sin cerrar, no se considera de dicho
lenguaje, lo que nos permite colocar HTML
-->

<?php
}
?>
```


Ahora vamos a ir a nuestra página llamada plantilla y vamos a cortar el maquetado de la barra de navegación, y lo vamos a pegar dentro del archivo que acabamos de crear, el que contiene la función menú, y lo vamos a pegar justo entre las dos partes en que dividimos nuestra función.

Si actualizamos nuestro navegador, veremos que la barra de navegación desapareció.

Vamos a agregar la línea de código que incluye nuestro archivo externo y luego la que invoca la función, para que tengamos nuestra plantilla igual que antes, pero mucho mas prolija y reutilizando el código del menú.

Así debería quedar nuestro código del archivo plantilla.

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Plantilla Base</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <?php
      require("inc/menu.php")
    ?>
  </head>
  <body class="container">
    <!-- Barra de navegación -->
    <?php menu(); ?>
    <script src="js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

Agregué el siguiente código:

```
<?php
  require("inc/menu.php")
?>
```

Y este:

```
<!-- Barra de navegación -->
<?php menu(); ?>
```

Nuevamente, deberíamos poder ver desde nuestro navegador de internet, la siguiente pantalla:

Autor Inicio Formularios ABM Opciones ▾

Buscar contenido

Buscar

Actividad 3: Agregamos un título permanente.

Para que el usuario pueda identificar en que sector de nuestro sistema web se encuentra, además de la etiqueta TITLE que agregaba una breve descripción a la pestaña del navegador, vamos a colocar un pequeño contenedor al que le agregaremos un título genérico y lo modificaremos en cada archivo que vayamos a crear a partir de nuestra plantilla.

El elemento que vamos a agregar es un ALERT de Bootstrap, y lo haremos inmediatamente después del menú de navegación.

Así debería quedar nuestro BODY:

```
<body class="container">
  <!-- Barra de navegación -->
  <?php menu(); ?>
  <!-- Título de la página -->
  <div class="alert alert-primary text-center fst-italic" role="alert">
    <h4>Título de la página.</h4>
  </div>
  <script src="js/bootstrap.bundle.min.js"></script>
</body>
```

Y esto deberíamos ver por pantalla:

Autor Inicio Formularios ABM Opciones ▾

Buscar contenido

Buscar

Título de la página.

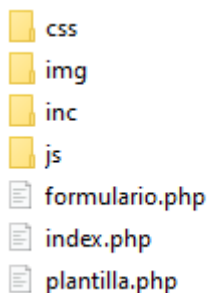
Si llegamos a este resultado, damos por finalizada la actividad tres.

Actividad 4: Página de formulario.

En esta actividad vamos a aprender a enviar datos desde la casa de nuestros usuarios hasta el servidor donde está alojada nuestra página web. Comenzaremos creando el archivo de la práctica a partir de nuestra plantilla.

Vamos a llamarlo "Formulario" y lo guardamos junto a las páginas de nuestra web.

Le modificamos la etiqueta TITLE y el título de la página, podemos colocarle el texto: “Envíos de datos al servidor.”



A continuación, vamos a editar el archivo que contiene el menú de navegación y a agregar el enlace para este archivo que acabamos de crear.

```
<li class="nav-item">
  <a class="nav-link active" aria-current="page" href="index.php">Inicio</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="formulario.php">Formulario</a>
</li>
```

Guardamos todos los cambios y actualizamos para ver si ya tenemos visible nuestra nueva página.

Procedemos luego a maquetar una grilla a partir de una fila que dividiremos en tres secciones.

```
<!-- Fila 1 -->
<div class="row">
  <div class="col-3"></div>
  <div class="col-6"></div>
  <div class="col-3"></div>
</div>
```

Y en la división del centro, colocamos un pequeño formulario en principio para enviar dos datos hasta el servidor, un nombre y un apellido. Colocamos también un botón para el envío del formulario.

```
<form>
  <div class="mb-3">
    <label for="nombre" class="form-label">Nombre</label>
    <input type="text" class="form-control" id="nombre" name="nombre">
  </div>
  <div class="mb-3">
    <label for="apellido" class="form-label">Apellido</label>
    <input type="text" class="form-control" id="apellido" name="apellido">
  </div>
  <button type="submit" class="btn btn-primary">Enviar</button>
</form>
```

Agregamos la página de destino y el método de envío.

Luego creamos dicha página, que podría llamarse “formulario_destino”

Los valores del atributo NAME son muy importantes porque es el índice con el que el servidor va a almacenar los datos que enviamos.

A continuación, abrimos esta nueva página de destino donde comenzaremos a escribir el código que nos permitirá ver el envío de datos desde el cliente, y para hacer las pruebas con diferentes elementos, veremos como se reciben desde el servidor, las asignaremos a nuevas variables y las mostraremos por pantalla.

Colocamos el siguiente código en la nueva página:

```
<?php
// Recibo los datos.
$p_nombre = $_POST['nombre'];
$p_apellido = $_POST['apellido'];

echo 'Nombre recibido: '.$p_nombre.'<br>';
echo 'Apellido recibido: '.$p_apellido;

?>
```

Lo que hicimos fue crear un par de variables y asignarles el contenido del arreglo POST, que es el arreglo donde se guardan los datos que enviamos por dicho método, y recordamos que se guardan en dicho arreglo usando como índice el valor que colocamos dentro del formulario usando el atributo NAME en cada etiqueta.

Luego, los mostramos por pantalla para verificar que los datos fueron enviados.

El nombre ingresado es: Guillermo

El apellido ingresado es: Alfaro

Campo PASSWORD.

El campo PASSWORD es un campo de texto con la particularidad que no se muestran los caracteres ingresados.
Agreguemos uno a nuestro formulario.

```
<div class="form-group">  
  <label for="clave">Clave del usuario.</label>  
  <input type="password" id="clave" name="clave" placeholder="Ingrese su c  
lave" class="form-control">  
</div>
```

Así quedaría nuestro formulario:

Nombre.

Apellido.

Password.

Así como agregamos un elemento en nuestro formulario, tenemos que agregarlo en la página destino si queremos utilizarlo.

```
// Recibo los datos.  
$p_nombre = $_POST['nombre'];  
$p_apellido = $_POST['apellido'];  
$p_clave = $_POST['clave'];  
  
echo 'Nombre recibido: '.$p_nombre.'<br>';  
echo 'Apellido recibido: '.$p_apellido.'<br>';  
echo 'Clave ingresada: '.$p_clave;
```

Nombre recibido: Guillermo
Apellido recibido: Alfaro
Clave ingresada: aplicacionesweb

CHECKBOX.

Este elemento permite al usuario seleccionar varias opciones a la vez.
La siguiente es la sintaxis básica:

```
<hr size="2px" color="black" />  
  
<h6 class="font-italic">Seleccione sus materias favoritas.</h6>  
<div class="form-group">  
  <input type="checkbox" id="materia1" name="materia1" value="php">  
  <label for="materia1">Aplicaciones Web</label>  
</div>  
<div class="form-group">  
  <input type="checkbox" id="materia2" name="materia2" value="java">  
  <label for="materia2">JAVA</label>  
</div>  
<div class="form-group">  
  <input type="checkbox" id="materia3" name="materia2" value="asp_net">  
  <label for="materia3">ASP Net</label>  
</div>
```

De la misma forma que hicimos con los cuadros de texto, para este caso también deberemos recibirlos en la página destino mediante POST, si miramos el código en detalle, veremos que cada una de las opciones del CHECKBOX tiene un valor distinto en el atributo NAME, por lo que

deberemos tratarla a cada una de dichas opciones, como si fuese un elemento independiente de los demás.

```
if(isset($_POST['materia1'])){  
    $p_materia1 = $_POST['materia1'];  
}else $p_materia1 = '';  
  
if(isset($_POST['materia2'])){  
    $p_materia2 = $_POST['materia2'];  
}else $p_materia2 = '';  
  
if(isset($_POST['materia3'])){  
    $p_materia3 = $_POST['materia3'];  
}else $p_materia3 = '';
```

Es importante verificar que el POST exista, caso contrario, si el usuario no selecciona alguna de las opciones, el intérprete PHP nos dará error.

Lo usual es verificar la existencia mediante la función ISSET y en caso de no existir, asignarle un valor por defecto, en nuestro caso, una cadena vacía.

Nombre recibido: Guillermo

Apellido recibido: Alfaro

Clave ingresada:

Tus materias favoritas son: java asp_net

RADIO BUTTON.

Este elemento permite al usuario elegir una, y solo una, opción de varias. Veamos el código básico para agregar dicho elemento a nuestro formulario: Si vemos el siguiente código, notaremos que todos los elementos tienen el mismo valor en el atributo NAME, es por ese motivo que, al seleccionar, solo podremos elegir una sola de todas las opciones.

```
<fieldset>
<legend>Seleccione su nivel de inglés</legend>
  <div class="form-group">
    <label>
      <input type="radio" name="nivel" value="alta">Alto
    </label>
  </div>
  <div class="form-group">
    <label>
      <input type="radio" name="nivel" value="medio">Medio
    </label>
  </div>
  <div class="form-group">
    <label>
      <input type="radio" name="nivel" value="bajo">Bajo
    </label>
  </div>
</fieldset>
```

Veamos como recibir los valores del elemento RADIO en el archivo de destino.

De la misma manera que hicimos con el CHECKBOX, acá también debemos verificar que el dato realmente fue enviado para evitar que se produzca un error.

```
if(isset($_POST['nivel'])){
    $p_nivel = $_POST['nivel'];
}else $p_nivel = 'El nivel de ingles no fue enviado';
```

Desplegable (SELECT).

El elemento SELECT posibilita al usuario elegir una opción de una lista desplegable. Veamos el siguiente ejemplo básico:

```
<div class="form-group">
  <label for="selector1">Seleccione el motivo de su contacto.</label>
  <select name="selector1" id="selector1">
    <option value="consulta">Consulta</option>
    <option value="sugerencia">Sugerencia</option>
    <option value="queja">Queja</option>
  </select>
</div>
```

Y de esta forma recibimos el valor seleccionado en la página de destino:


```
$selector = $_POST['selector1'];
```