

Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 8- Archivos externos. Sesiones.

Una forma práctica y prolija de trabajar en PHP, es la inclusión de archivos externos. Generalmente nuestros programas van a ser más largos de los que estamos haciendo ahora, y seguramente nuestro código se extienda mucho más. Para que esto no suceda, podemos utilizar otros archivos, que pueden contener funciones y procedimientos, y llamarlos solamente cuando necesitemos utilizarlos.

Supongamos por ejemplo que necesitamos crear una base de datos con varias tablas, seguramente ese esquema de datos lo vamos a utilizarlo varias veces, no solamente una vez, si necesitamos hacer un insert, mostrar datos de una o varias formas, etc...

Entonces cada vez que hacemos el llamado a la base de datos, tendríamos que escribir el código de conexión a la misma. Si la llamamos 800 veces tendríamos que escribir la conexión 800 veces. Para evitar esto se utilizan **archivos externos**, para que en vez de escribir la conexión 800 veces, la escribamos **solamente una vez** y cada vez que la necesitemos llamemos al archivo de conexión. Esto nos ahorra tiempo, código y queda mucho más prolijo y entendible.

Otro ejemplo, puede ser algo muy común y hasta diría obligatorio en las páginas web, como lo son los pies de página (**footer** en inglés), cada vez que hacemos una página nueva de nuestro sitio web debemos agregarle el pie de página, y esto sería repetir código. Entonces ¿por qué no dejar el footer en un archivo y cada vez que lo necesite sólo llamarlo? Bueno, también para esto se utilizan archivos externos.

En PHP hay varias formas de llamar a archivos externos:

- **Include.**
- **Include_once.**
- **Require.**
- **Require_once.**

La sentencia **include** incluye y evalúa el archivo especificado, para que esto suceda nuestro archivo debe tener la extensión **.php**. Y para incluir el archivo debemos hacerlo de la siguiente manera:

```
include 'inc/conexion.php' ;
```

En este ejemplo anterior, estamos llamando a un archivo que se encuentra en el directorio **inc** y el mismo se llama **conexion.php**.

Este archivo debería tener la conexión a la base de datos, de esta manera siempre que necesitemos conectar a la base, solamente hacemos el llamado a este archivo (conexion.php).

Ejemplo práctico:

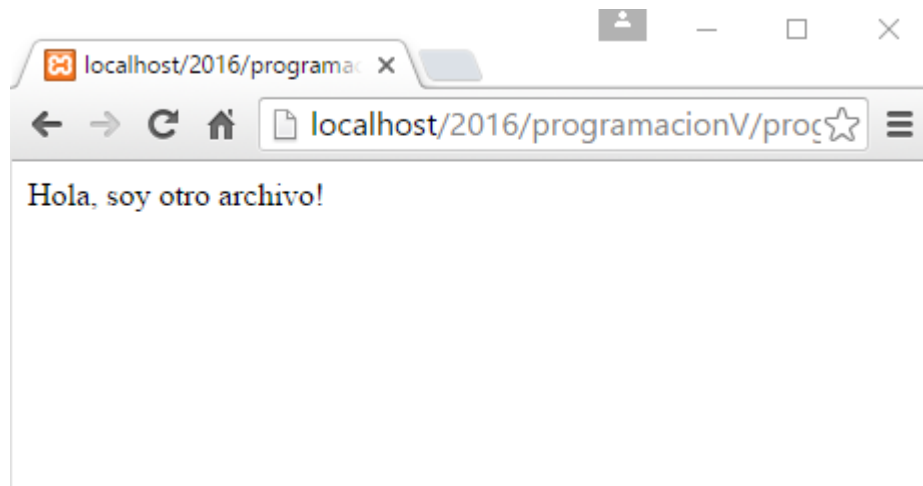
index.php

```
<?php
    include 'mensaje.php';
?>
```

mensaje.php

```
<?php
    echo 'Hola, soy otro archivo!';
?>
```

Resultado en el navegador:



De esta manera, estamos llamando a un archivo dentro de nuestro servidor que contiene un mensaje.

La sentencia **include_once**, es muy parecida a la función **include**, nada más que esta, incluye el archivo sólo una vez, es decir si el archivo ya fue incluido y es vuelto a ser llamado, no lo volverá a incluir. Como su nombre lo indica, lo incluye una vez.

include_once se puede utilizar en casos donde el mismo fichero podría ser incluido y evaluado más de una vez durante una ejecución particular de un script, así que en este caso, puede ser de ayuda para evitar problemas como la redefinición de funciones, reasignación de valores de variables, etc.

Y por último la sentencia **require**, es también similar a la función **include**, nada más que en caso de producirse una excepción o fallo producirá un error fatal, es decir se detiene el script mientras que el **include** sólo emitirá una advertencia. Por ejemplo si cargamos un archivo de conexión a base de datos y por alguna razón este archivo no

es encontrado, el script no se ejecutará, se detendrá, mostrará un mensaje de **error fatal**.

Con **require_once** pasa exactamente lo mismo que con `include_once`, incluye el archivo una sola vez, pero si este no está disponible nos daría un error del tipo fatal; es decir no se ejecutaría el script.

¿Cuántos **include** podemos utilizar por programa?

La respuesta es simple: Todos los que queramos y donde queramos. Dentro de una página PHP podemos hacer múltiples llamadas a distintos archivos a través de **include**, incluso, podemos llamar a un mismo archivo varias veces en distintos momentos de la ejecución.

El lugar de la ejecución es simplemente donde necesitemos hacer la inclusión. Si necesitamos incluir al principio, lo hacemos al principio, si es al medio o dentro de un ciclo o dentro de una iteración condicional, lo hacemos ahí. No hay límites con respecto a eso. Además, los **include** pueden ser anidados, es decir, podemos hacer una llamada a una página que a su vez realice una llamada a otra página.

Los **include** son muy útiles, no solamente para guardar funciones en los mismos, sino también para maquetar una página web, con la ayuda de estilos **CSS** (recordemos que el archivo tiene que tener la extensión .php).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Incluir archivos</title>
</head>
<body>
    <?php
        include 'inc/head.php';
        include 'inc/detail.php';
        include 'inc/footer.php'
    ?>
</body>
</html>
```

Generalmente estos tres documentos .php, son los que comúnmente se repiten en una página web. La cabecera, la parte del medio y el pie de página. Y como podemos ver en el código de ejemplo anterior están guardados en un directorio llamado **inc**. De esta manera no tenemos que repetir cada página cada vez que lo necesitamos, sino que simplemente hacemos el llamado a ese archivo.

Sesiones

Muchas veces nos sucede que necesitamos pasar una variable entre varias páginas, para hacer esto debemos crear sesiones, o mejor dicho variables de sesión. Hasta ahora hemos visto que creando un formulario en HTML, el usuario ingresaba un valor y ese valor podía ser procesado en el servidor, pero ¿qué pasaría si yo quiero guardar ese valor y utilizarlo más adelante en otra página sin utilizar base de datos? La respuesta es utilizando **sesiones**.

Una sesión es una variable que se crea en el servidor y esta variable puede ejecutarse sin que el usuario de la Web tenga conocimiento alguno de ello y la misma puede utilizarse en cualquier momento y lugar de nuestro sitio web. El ejemplo clásico de sesiones, es el carrito de compras de una página web.

En una tienda de libros, un usuario inicia sesión (puede ser con usuario y contraseña), busca en la web varios libros y a medida que va buscando elige algunos para posteriormente adquirirlos, para esto el usuario, pasa por varias páginas, “novelas”, “drama”, “tecnología” entre otras y va eligiendo en cada una de ellas lo que quiere comprar. Al finalizar su búsqueda, este usuario eligió comprar 7 libros, se dirige a la sección de compras de la página web, la misma le realiza la suma del total a pagar, ingresa sus datos de envío del producto y finaliza la operación.

Para nosotros los programadores, este usuario del ejemplo anterior, cada libro que elige es guardado en una variable de sesión, para nosotros no son libros, sino variables que va generando el usuario a medida que navega por nuestra página web, y que vamos pasando de página en página hasta que el usuario decide finalizar la sesión y comprar o no el producto. Es lógico que cada sesión tenga una duración si la página web esta inactiva, esta sesión se destruye sola y el usuario si quiere comprar va a tener que logearse nuevamente para comenzar el proceso de compra.

Pero ¿en qué se diferencian las sesiones de los métodos **POST** y **GET**? Los métodos **POST** y **GET** permiten que los usuarios asignen valores a variables, y también permiten que la propia página tome valores internos, por ejemplo de una base de datos, y opere con ellos. Los valores emitidos por **POST** y **GET** pueden tener un origen conocido para el usuario y también su destino puede ser. De cualquier forma, el usuario podrá tener conocimiento de las variables que se envían y de sus valores. Pues bien, con las sesiones podemos hacer lo mismo, pero con una diferencia, la variable de sesión será recuperable en cualquier parte del sitio Web sin tener que crear enlaces de pasos de variable o formularios con métodos **GET** o **POST**, por lo que el usuario no sabrá ni cuando se crea (aunque lo pueda intuir) ni donde o cuando se recupera y ejecuta la variable. Simplemente cuando pasas de una página a otra mediante un enlace normal y corriente, tendrás la variable disponible para usarla.

Veamos un ejemplo práctico para entender un poco más el tema:

Un usuario ingresa su nombre, luego pasa a otra página donde nos muestra su nombre y elige una bebida entre varias, para finalizar pasa a otra página donde se muestra el nombre, y la bebida que eligió.

Para este ejemplo voy a utilizar 3 archivos **.php**, el primer archivo podría ser **.html**, eso como más les guste, **pero si o si los siguientes 2 deben tener extensión .php**.

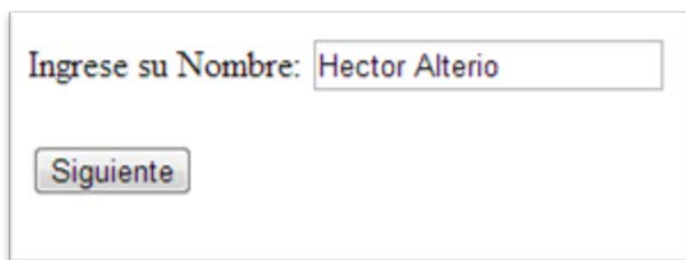
index.php o index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Encuesta</title>
</head>
    <body>
        <form action="2.php" method="POST">
            <label>Ingrese su Nombre: </label>
            <input type="text"
name="nombre"><br><br>
            <input type="submit"
value="Siguiente">
        </form>
    </body>
</html>

```

Hasta ahora es todo conocido, un formulario con un **label** y un input del tipo **text**, acompañados de un **botón** "Siguiente":



1.php

```

<?php
    SESSION_START( );

    $nombre = $_POST['nombre'];
    echo "Mi nombre por POST es:
".$nombre."<br><br>";
    $_SESSION['nombre'] = $nombre;
    echo "Mi nombre por SESSION es:
".$_SESSION['nombre']."<br><br>";

    echo '
<form action="3.php" method="POST" >
    <select name="bebidas">
        <label>Seleccione la bebida: </label>
        <option>Gaseosa</option>

```

```
        <option>Vino</option>
        <option>Cerveza</option>
    </select>
    <input type="submit" value="Siguiente">
</form><br>
<a href="1.php">volver</a>
';
?>
```

En este archivo tenemos que tener en cuenta varias cosas... para comenzar a utilizar variables de sesión, siempre delante de todo en la cabecera del archivo debemos escribir:

```
SESSION_START( );
```

Esto hace que el intérprete, sepa que vamos a utilizar variables de sesión. Sin esa función por más que agreguemos variables de sesión el sistema no las va a reconocer como tales. Esta función deberá estar al comienzo de todas las páginas que manejemos variables de sesión.

Luego para guardar una variable de sesión hacemos lo siguiente:

```
$_SESSION['nombre'] = $nombre;
```

Es decir al contenido de la variable \$nombre, lo guardo como variable de sesión con **\$_SESSION[' ']**. De esta forma guardamos datos en una variable de sesión y la misma va a estar guardada todo lo que dure la sesión.

Luego de mostrarle el nombre ingresado al usuario por POST y por SESSION le doy opciones para elegir una bebida, agrego un botón siguiente y uno para volver atrás:



The screenshot shows a web form with the following content:

- Mi nombre por POST es: Hector Alterio
- Mi nombre por SESSION es: Hector Alterio
- A dropdown menu with "Cerveza" selected.
- A "Siguiente" button.
- A purple underlined link labeled "volver".

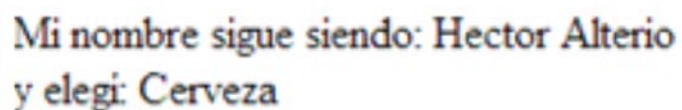
3.php

```
<?php
    SESSION_START( );

    $_SESSION['bebida'] = $_POST['bebidas'];
    echo "Mi nombre sigue siendo:
".$_SESSION['nombre']. "<br> y elegí:
".$_SESSION['bebida'] ;

    SESSION_DESTROY( );
?>
```

Nuestro último archivo, comienza con **SESSION_START()**, ya que como dije anteriormente seguimos utilizando variables de sesión, guardo en una nueva variable de sesión la bebida que el usuario eligió en la página anterior, y por ultimo muestro su nombre y la bebida elegida. Y destruyo la sesión con **SESSION_DESTROY()**.



Mi nombre sigue siendo: Hector Alterio
y elegi: Cerveza

Resumiendo, si utilizo variables de sesión siempre en el encabezado del archivo PHP en donde va a pasar la variable debo comenzar la sesión con **SESSION_START()**, y cuando termino la sesión (en el último archivo que la utilice) debo finalizarla con **SESSION_DESTROY()**.

Para crear una variable de sesión utilizo **\$_SESSION[' ']**, entre comillas el nombre de la variable.

Actividades

1. Realizar un sitio web en donde el usuario ingrese un número y me muestre en otra página la tabla de multiplicar del mismo de 0 a 10. Se deberá utilizar una función y deberá ser incluida en otro archivo.
2. Realizar un sitio web que me permita calcular el iva (21%) de un monto ingresado por el usuario. El iva deberá ser calculado con una función y deberá estar en otro archivo que deberá ser incluido.
3. Realizar una encuesta que tenga 4 pantallas. La primera que le pida al usuario sus datos personales (Nombre, Apellido, Dirección, Mail, Teléfono, Sexo, Edad), una vez que lleno sus datos personales la segunda pantalla deberá pedirle los estudios cursados (Primario, Secundario, Terciario, Universitario), la tercer pantalla deberá preguntarle si practica algún deporte (fútbol, tenis, vóley basquet), que libros lee (Acción, Novelas, Dramas, Técnicos) y que tipo de comida le gusta (China, Vegetariana, Pastas). Por último la cuarta pantalla deberá mostrar todos los datos ingresados por el usuario en todas las pantallas anteriores. "Ud es: [Datos personales], tiene estudios de: [Estudios cursados], le gusta practicar [Deporte], le gusta leer [Libros] y le gusta la comida: [Comida]."
4. Agregarle al punto anterior la posibilidad de que el usuario ingrese al sistema a través de un login (sin base de datos). Si usuario es igual a [USUARIO] y contraseña es igual a [CONTRASEÑA], que pueda ingresar.

Autoevaluación

1. ¿En qué nos beneficia el incluir un archivo?
2. ¿Qué diferencia hay entre include y require?
3. ¿Qué diferencia hay entre include_once e include?
4. ¿Para qué se utilizan las variables de sesión?
5. Explique la función **SESSION_START()**.

Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 9- Estilos CSS.

Las hojas de estilo **en cascada** o **CSS** (en inglés, **Cascading Style Sheets**) se utilizan para definir la parte visual de un documento HTML.

Entre los elementos que podemos definir se encuentran, por ejemplo, el texto, el color, las imágenes, las posiciones de las secciones y más.

Las **instrucciones CSS** (reglas de formato), pueden escribirse directamente en el archivo HTML de la página, o bien en un archivo separado que luego deberemos vincular al archivo HTML.

Si agregamos el código CSS en el mismo documento HTML, debemos hacerlo entre las etiquetas **<head>** y **</head>** de nuestra página web.

Las siguientes líneas de código muestran un ejemplo de estilo CSS para definir el formato de un párrafo:

```
<style>
  p{ font-family: Arial; font-size: 32px} ;
</style>
```

En este ejemplo, vemos cómo, entre las etiquetas de apertura y cierre **<style></style>**, estamos modificando un párrafo **<p></p>**, cambiándole el tipo de fuente (**Arial**) y su tamaño (**32px**).

Una buena costumbre no escrita en programación, pero que recomendamos, es siempre utilizar un archivo separado para escribir las instrucciones de hojas de estilo (CSS). De esta forma queda bien separado, por un lado, el diseño y, por otro, la programación. Es más prolijo, sobre todo cuando el código de nuestra página web o proyecto es extenso.

Para esto, lo que tenemos que hacer es escribir el conjunto de reglas de formato en un archivo con la extensión **.css** y, posteriormente, en el archivo de nuestra página web, vincular el archivo **.css** al documento HTML.

Por ejemplo, en un archivo llamado **estilo.css** creamos un formato para los encabezados **h1**: queremos que sean de color rojo, tengan fuente Verdana y un tamaño en pixeles de 15, como se muestra en el código a continuación:

```
h1{
  font-family: Verdana;
  font-size: 15px;
  color: red;
}
```

Para vincular este archivo, escribimos entre las etiquetas **<head></head>** del documento HTML el siguiente código:

```
<head>
  <link href="css/estilo.css" type= "text/css"
  rel="StyleSheet" />
```

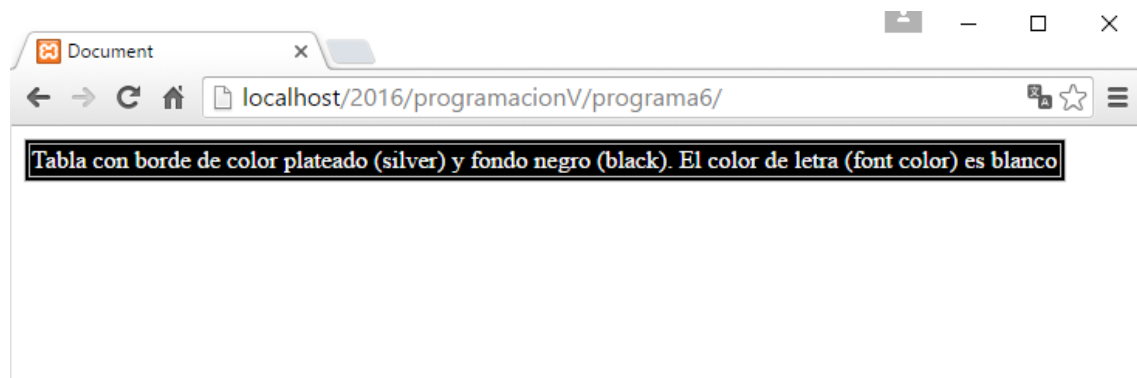
```
</head>
```

Las etiquetas **<link></link>** nos sirven para escribir un **enlace a un documento externo** y, mediante la propiedad **href**, indicamos dónde se encuentra ubicado el archivo (en nuestro ejemplo, se encuentra dentro de la carpeta **css**).

Con el atributo **type= "text/css"**, le estamos indicando el contenido del archivo. Por último, con el atributo **rel** especificamos la relación que existe entre nuestra página y la de destino, por eso, en este caso debemos poner el valor **StyleSheet**, porque nuestro destino es una hoja de estilo que va a contener los estilos del documento HTML.

Otra forma de agregar estilo **y para nada recomendada** es hacerlo directamente sobre el componente (Por favor no lo hagan, aunque a veces puede servir para probar de manera rápida un estilo):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <table border=1 bordercolor="silver"
bgcolor="black">
    <td><font color="white">Tabla con borde de
color plateado (silver) y fondo negro (black). El
color de letra (font color) es blanco</font></td>
  </table>
</body>
</html>
```



Si por ejemplo queremos que todos párrafos tengan un determinado estilo, como vimos anteriormente, debemos en nuestra hoja de estilo escribir la “**p**” y a partir de ahí el estilo correspondiente, entre llaves. Si queremos que todos los títulos h2 tengan un determinado estilo, debemos escribir en la hoja de estilo “**h2**” y entre llaves el estilo. O si queremos que todos los formularios tengan un determinado estilo debemos escribir “**form**” y el estilo que queremos que tenga entre llaves. Pero ¿qué pasaría si queremos que solamente un párrafo o cualquier componente individualmente tengan un estilo?

Para esto tenemos dos formas de hacerlo, siempre tenemos que pensar bien que es lo que queremos lograr para utilizar cualquiera de las dos formas.

Si solamente vamos a cambiar un estilo de un componente en particular, podemos hacerlo a través de un identificador (**id**); ahora si queremos que un grupo de componentes del mismo tipo, tengan un estilo determinado, podemos hacerlo a través de una clase (**class**).

Para utilizar un id o identificador, debemos agregárselo al componente:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link href="estilo.css" type="text/css"
rel="StyleSheet" />
  <title>Estilos</title>
</head>
<body>
  <h1 id="titulo">Título H1</h1>

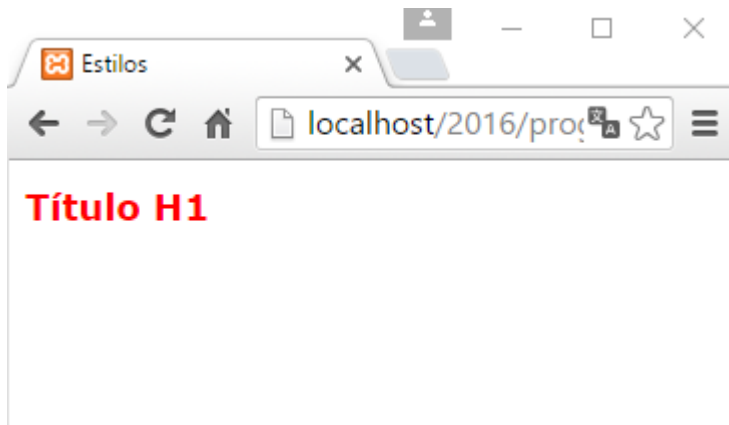
</body>
</html>
```

Y en la hoja de estilo (estilo.css):

```
/*Esto es un comentario en CSS*/

#titulo{
  font-family: verdana;
  font-size: 18px;
  color: red;
}
```

Y en el navegador veríamos lo siguiente:



En el ejemplo anterior a través del id, estamos cambiando el estilo. El identificador es único a diferencia de la clase que se puede utilizar en varios componentes del mismo tipo:

index.html

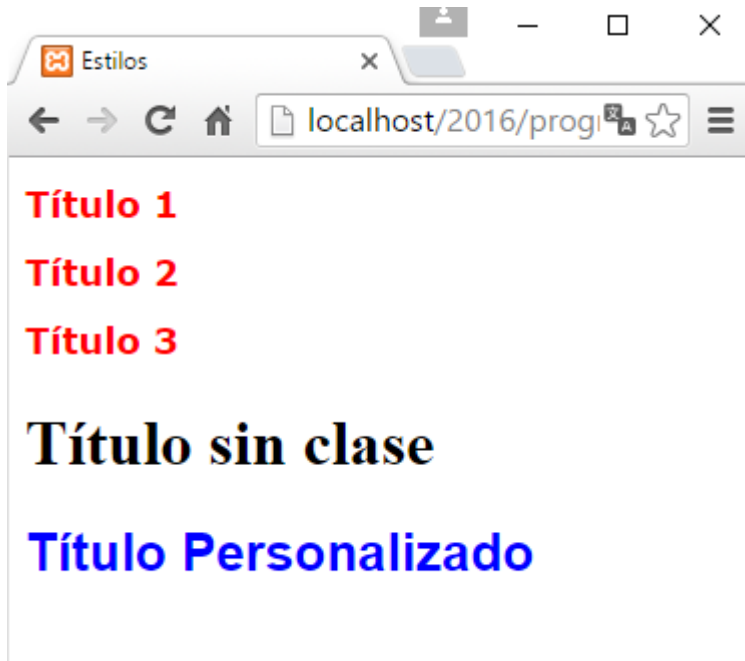
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <link href="estilo.css" type="text/css"
rel="StyleSheet" />
  <title>Estilos</title>
</head>
<body>
  <h1 class="titulo">Título 1</h1>
  <h1 class="titulo">Título 2</h1>
  <h1 class="titulo">Título 3</h1>
  <h1>Título sin clase</h1>
  <h1 id="tituloPersonal">Título
Personalizado</h1>
</body>
</html>
```

Estilo.css

```
.titulo{
  font-family: verdana;
  font-size: 18px;
  color: red;
}
#tituloPersonal{
  font-family: arial;
```

```
font-size: 26px;  
color: blue;  
}
```

Navegador:



Como vemos en el ejemplo anterior, los 3 primeros títulos corresponden a una clase llamada “**título**”, el título 4 **no tiene ningún id o clase, es del tipo h1**; y el último título al ser personalizado tiene un **identificador** llamado tituloPersonal.

Entonces cuando llamamos a un identificador desde la hoja de estilo, debemos hacerlo con un “#”, y si es una clase con un “.”.

Propiedades para los componentes hay miles, les recomiendo visitar: http://www.mclibre.org/consultar/htmlcss/css/css_propiedades.html aunque a medida que utilicen estilos, y por necesidad, las van a ir aprendiendo.

Como maquetar una página web utilizando etiquetas div y estilos

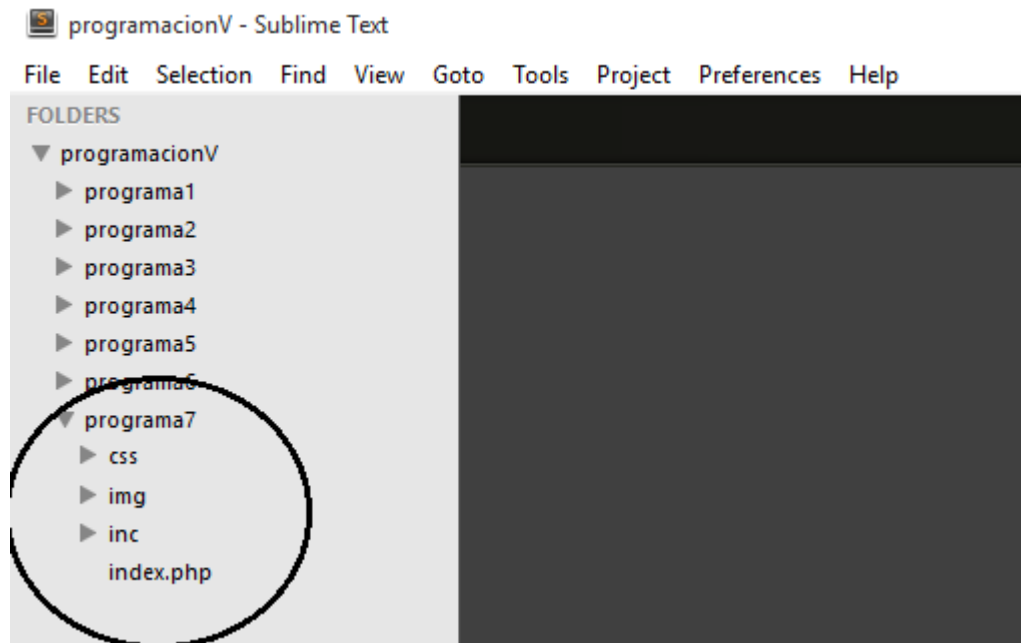
Como vimos anteriormente, cuando utilizamos un **include**, podemos incluir cualquier parte de código que necesitemos; y esto lo podemos utilizar para maquetar una web. Podemos darle una posición fija al pie de página, al encabezado y al contenido por ejemplo.

Ejemplo práctico

En primer lugar, lo que tenemos que hacer es organizarnos. Para esto creamos una carpeta contenedora del proyecto, en mi caso le puse **programa7**, pueden ponerle el nombre que quieran, dentro de la misma vamos a crear otras 3 carpetas llamadas **css**,

img, e **inc**. En la primera carpeta **css** vamos a guardar los archivos de estilo, en la carpeta **img** las imágenes del proyecto y en la carpeta **inc**, los archivos php que vamos a incluir en nuestro proyecto.

Creamos la primera página y la llamaremos **index.php** como hacemos siempre, este archivo tiene que estar en el directorio principal del proyecto. La ruta general nos tendría que quedar como la siguiente imagen:



En el archivo **index.php** escribimos el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/estilo.css">
    <title>Mi sitio</title>
</head>
<body>
    <?php
        include "inc/head.php";
        include "inc/detail.php" ;
        include "inc/footer.php";
    ?>
</body>
</html>
```

En el código anterior, lo primero que hacemos es una llamada a la carpeta donde va a estar nuestro archivo **estilo.css** para poder trabajar con los mismos. Y luego en el cuerpo de la página hago un include a los archivos **head.php**, **detail.php** y **footer.php**, estos tres archivos que crearemos más adelante, deben estar dentro de la carpeta **inc**.

Comencemos primero a trabajar con la cabecera de nuestra página. Creamos un archivo php y lo llamaremos **head.php**, el mismo debe estar dentro de la carpeta **inc**. Y dentro de la carpeta **img** agregamos una imagen con extensión **.png** que nos va a servir de logo, la misma debe tener un ancho y alto de 100px. Y el código del archivo es el siguiente:

```
<header id="head">
    <div>
        
        <h1>Mi web</h1>
    </div>
</header>
```

A la etiqueta **header**, le agrego un identificador para luego utilizar en la hoja de estilo y darle una posición.

Luego en un div agrego la imagen y un título. Para acomodar estos elementos en la hoja de estilo (estilo.css) que se encuentra en el directorio **css**, agrego el siguiente código:

```
#head{
    width: 100%;
    border: 1px solid gray;
    box-shadow: 0 1px 5px #999;
}
#head h1{
    margin-top: 5px;
    margin-left: 20px;
    font-size: 20px;
    font-family: verdana;
    color: gray;
}
#head img{
    border-radius: 8px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0
6px 20px 0 rgba(0, 0, 0, 0.19);
}
```

AL comienzo lo que hago es acomodar el elemento head, le doy un ancho del 100%, es decir que tome el total del ancho de la página, le doy un borde gris de 1px para que se vea, y le agrego una sombra. Los parámetros para la propiedad **box-shadow** son:

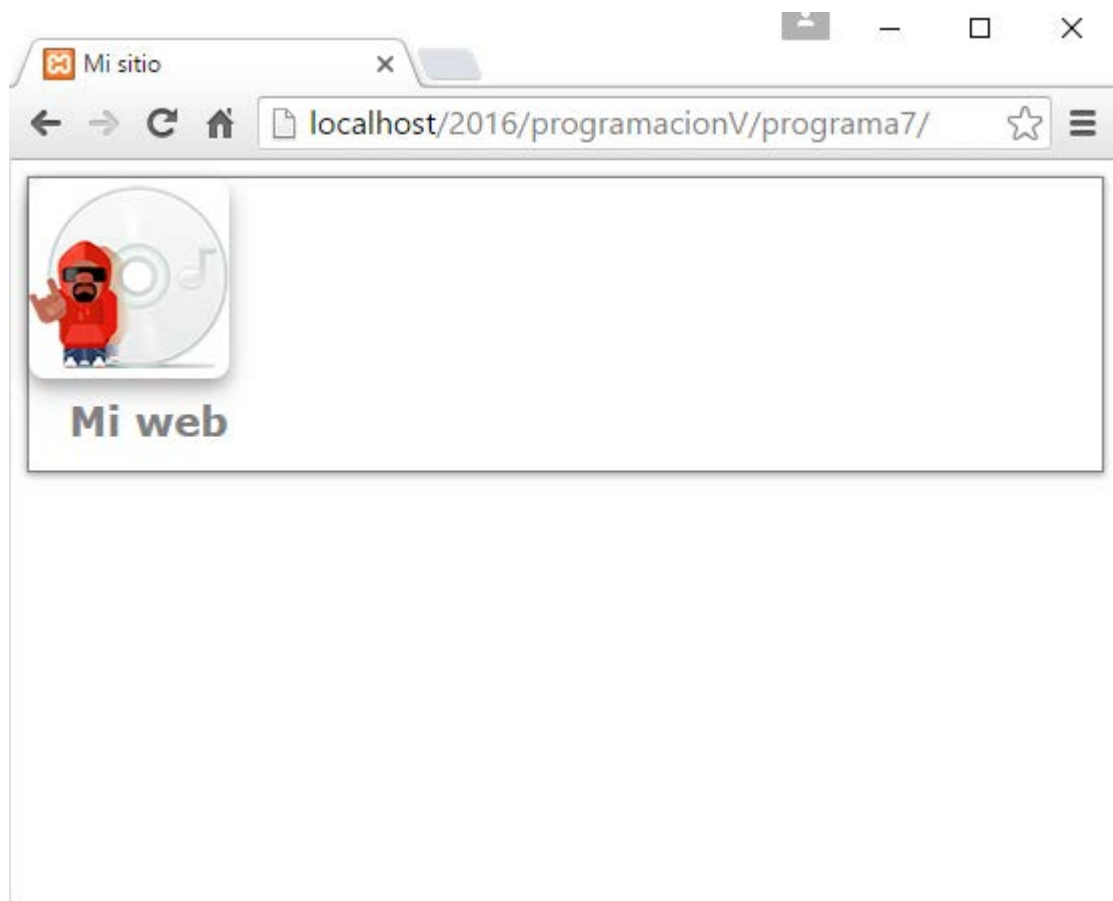
arriba, al costado, abajo y el color de la misma. Por eso recibe por valor **0 1px 5px #999**.

Para el título “**Mi web**”, le estoy diciendo que a todos los títulos **h1** que estén dentro del componente head, tengan un margen, tamaño de fuente, fuente y un color determinado.

Cabe aclarar que, siempre que quiera afectar con algún estilo en particular a un componente que sea hijo de un componente mayor, como en este caso, en el que el componente padre es **head** y los hijos son **h1** e **img**, puedo primero escribir el id del padre seguido del hijo, de esta forma afectara a todos los hijos que tengan ese nombre:

```
#head h1{  
}
```

Por ultimo para la imagen, le damos un efecto en el radio y una sombra.



Para lo que sería la parte central, creamos un archivo llamado **detail.php**, dentro de la carpeta inc.

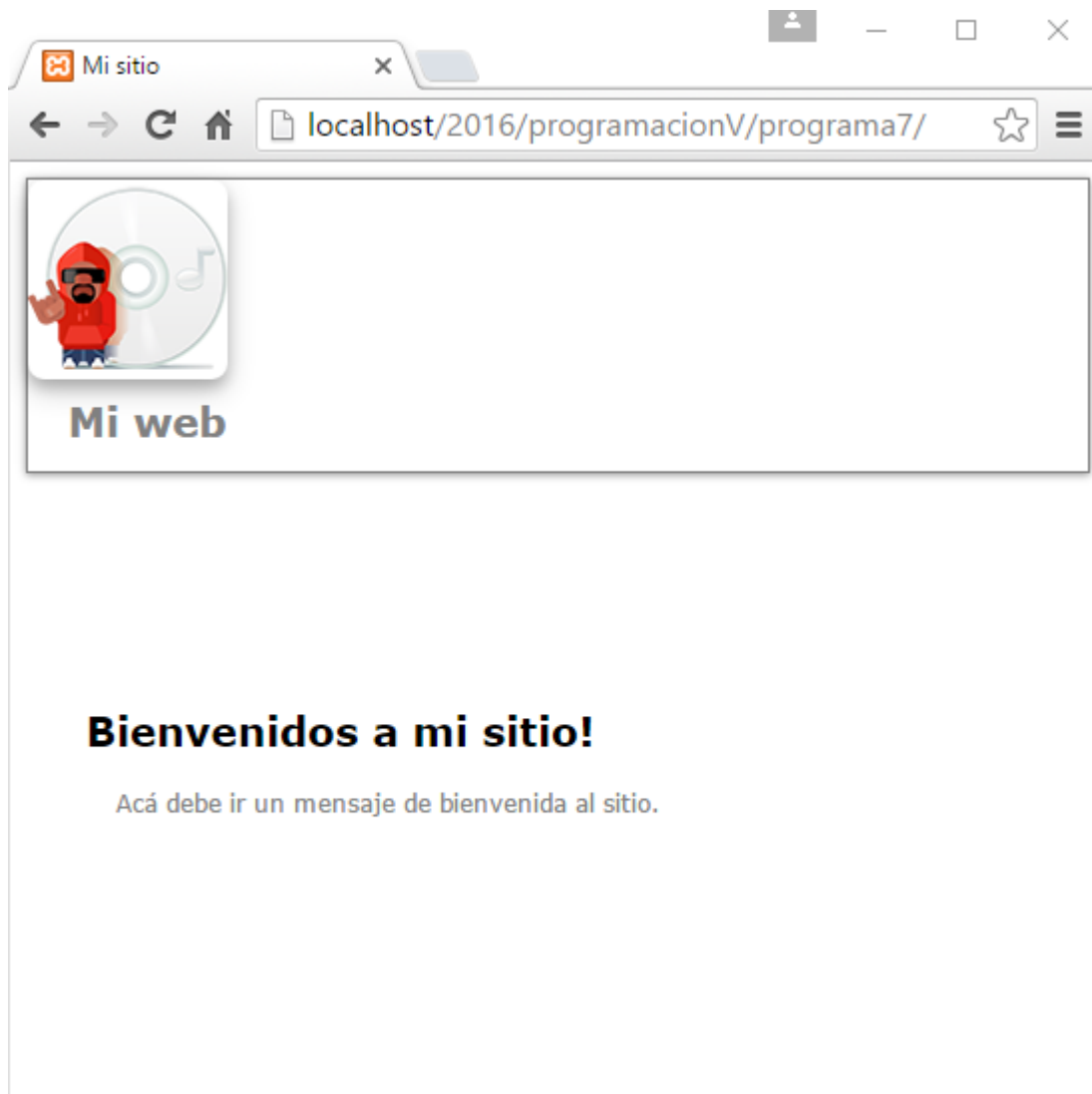

```
<article id="contenido">
  <div id="contenidoPcupal">
    <h1>Bienvenidos a mi sitio!</h1>
    <p>Acá debe ir un mensaje de bienvenida al
sitio.</p>
  </div>
</article>
```

Utilizo la etiqueta **article**, y le doy como un identificador llamado **contenido**, y también creo un div con el id **contenidoPcupal**. Esto es para poder luego, acomodarlo con el estilo. También agrego un título del tipo **h1** y un párrafo (**p**).

En el estilo para acomodarlo hago lo siguiente:

```
#contenido{
  padding-top: 100px;
}
#contenidoPcupal{
  margin-left: 30px;
}
#contenidoPcupal h1{
  font-size: 20px;
  font-family: verdana;
}
#contenidoPcupal p{
  padding-left: 15px;
  font-size: 12px;
  font-family: verdana;
  color: gray;
}
```

No hay mucho para explicar de este código, en contenido le doy un **padding-top** de 100px, es decir que tenga un espacio de 100px entre el head y el contenido. Y al div **contenidoPcupal** le doy un margen izquierdo de 30px y al título y al párrafo les doy color, tamaño y tipo de fuente.



Por último y para finalizar, nos queda el pie de página. Para esto agregamos un archivo que llamaremos **footer.php** dentro de la carpeta **inc**.

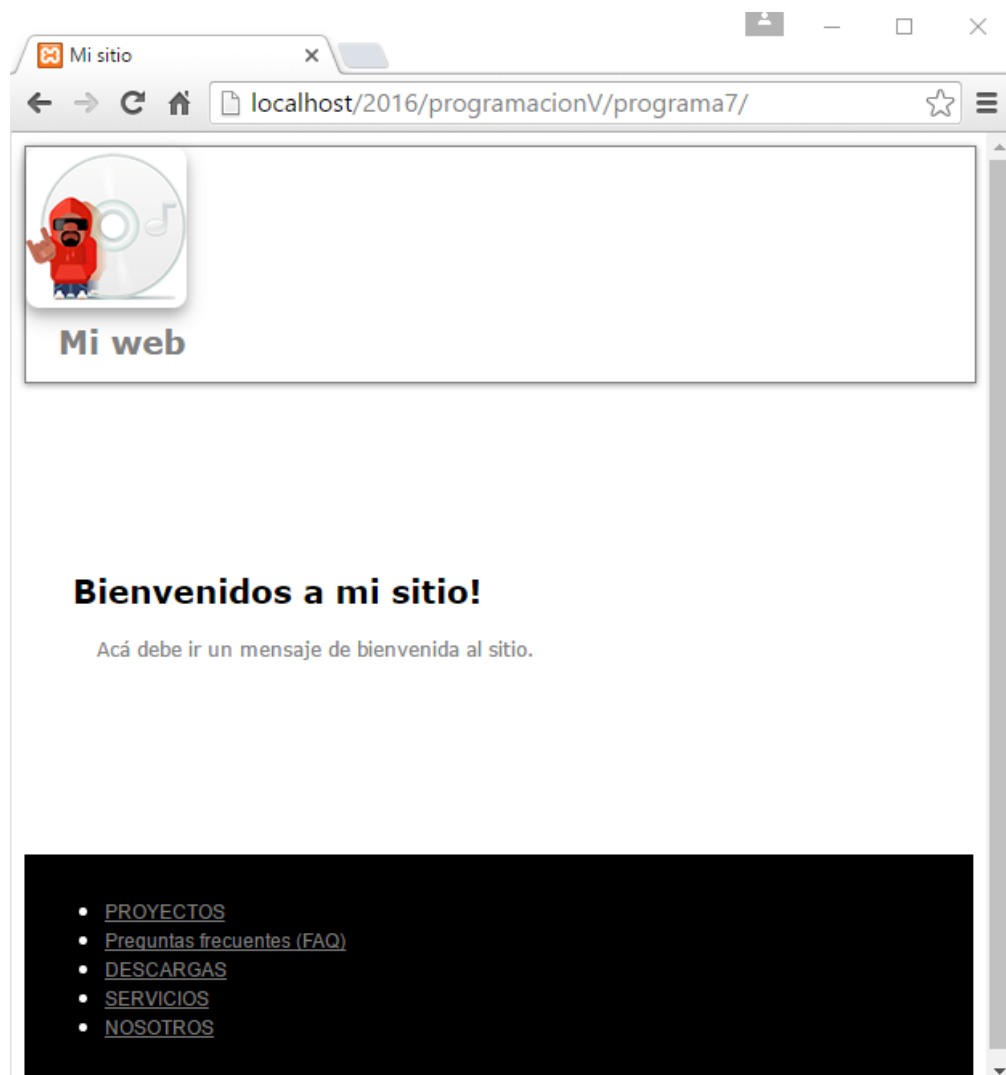
```
<footer id="footer">
  <ul>
    <li><a href="#">PROYECTOS</a></li>
    <li><a href="#">Preguntas frecuentes
(FAQ)</a></li>
    <li><a href="#">DESCARGAS</a></li>
    <li><a href="#">SERVICIOS</a></li>
    <li><a href="#">NOSOTROS</a></li>
  </ul>
</footer>
```

Agregamos un identificador llamado **footer**, y unos links que hacen referencia a parte de nuestro sitio (menú). Esto último lo hacemos con una lista HTML.

Para darle una ubicación agregamos el siguiente código a nuestro archivo **estilo.css**:

```
#footer{
    margin-top: 120px;
    padding: 10px;
    background-color: #000000;
}
#footer li{
    color: #fff;
}
#footer a, footer a:hover{
    color:gray;
    font-family: arial;
    font-size: 12px;
}
```

En el estilo, le damos una posición al pie de página y un color negro. A la lista le damos un color blanco y a los links de la misma le damos un tipo de letra Arial, un tamaño de 12px y un color gris. **Fíjense que este estilo se lo agrego al enlace (a) y cuando pasa el cursor por el mismo (hover).** Y el proyecto finalizado se vería de la siguiente manera:



Para más información sobre estilos: http://www.w3schools.com/css/css_boxmodel.asp

Videos CSS: <https://www.youtube.com/watch?v=5YiU9dgB6ZQ>

Sobre estilos parte de la información fue extraída de: *Bootstrap* (2015) de editorial Users, de mi autoría.

Actividades

1. Realizar un sitio web con todo lo aprendido en este capítulo.

Autoevaluación

1. ¿Para qué sirven los estilos CSS?
2. ¿Cuál es la mejor forma de utilizar un estilo, dentro del mismo documento o con un archivo aparte?
3. ¿Qué diferencia hay entre una clase y un identificador al hacer referencia al mismo a través de la hoja de estilos?
4. ¿De qué manera nos son útiles la función include?
5. ¿Se puede realizar una animación con estilos CSS3? De un ejemplo.

Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 11- Bootstrap.

Bootstrap es un framework creado por el equipo de desarrollo de la red social **Twitter** para realizar interfaces web adaptables (*responsive web design*) a cualquier dispositivo, ya sea una tablet, un teléfono o una PC de escritorio. Esto quiere decir que la interfaz se adapta automáticamente a cualquier tamaño y resolución de pantalla sin que el usuario tenga que hacer nada.

El framework utiliza hojas de estilo **CSS**, combinadas con el lenguaje de programación **JavaScript** (que veremos más en detalle al final de este capítulo) y, además, es compatible con la mayoría de los navegadores web, como Chrome, Mozilla Firefox, Safari.

Bootstrap es un software libre, por lo que los usuarios tienen la libertad de usarlo, mejorarlo y distribuirlo libremente.

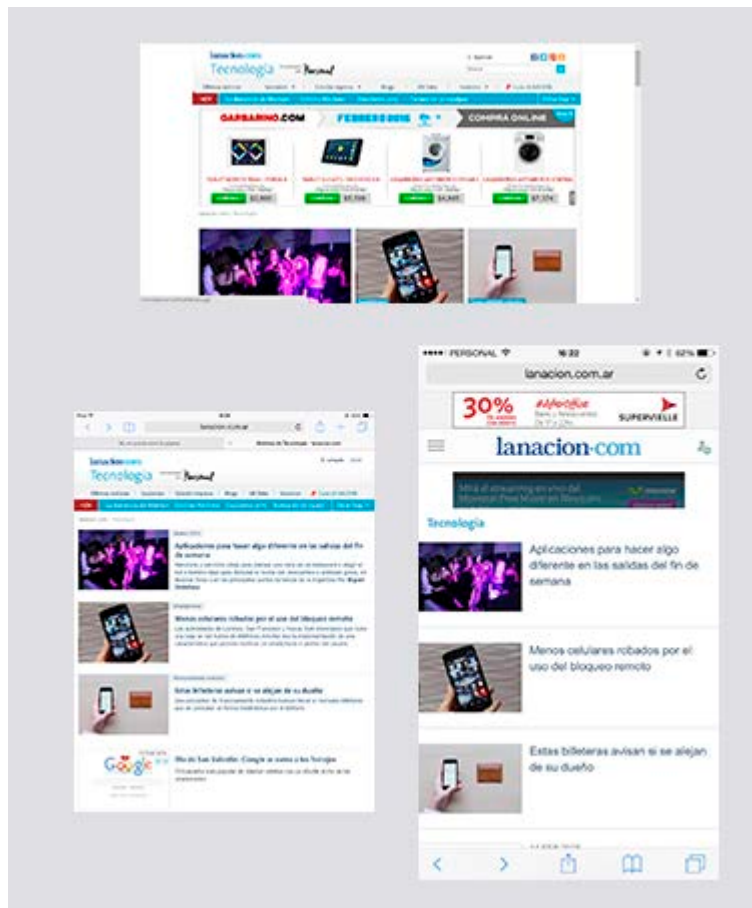
¿Qué es el diseño web adaptable?

Antes de que profundicemos en el concepto que maneja el framework de Bootstrap, tenemos que entender qué es **el diseño web adaptable** o, en inglés, **responsive web design (RWD)**.

Con el correr de los años, la tecnología creció y actualmente, a diferencia de años anteriores, hay más tablets y teléfonos inteligentes que computadoras de escritorio. Es decir, que disponemos de tamaños y resoluciones de pantallas diferentes, y, para que los sitios se visualicen bien en cada uno de estos dispositivos, los diseñadores o programadores deben adaptar el formato de sus sitios web a cada una de las pantallas disponibles en el mercado. Por esto surge la necesidad de desarrollar un formato de sitio web estándar que se adapte a todos los tamaños y resoluciones de pantalla: es ahí que surge el concepto de “web adaptable”.

Podríamos definir al diseño web adaptable como la técnica que nos permite diseñar páginas web que pueden ser visualizadas perfectamente en todos los dispositivos disponibles. Gracias a este tipo de diseño, ya no necesitamos programar o diseñar una página web para cada tipo de dispositivo existente en el mercado, ya que el sitio web se adaptará automáticamente al tamaño, resolución y orientación del dispositivo que estemos utilizando para acceder. La página web detecta de qué dispositivo se está conectando el usuario (mediante código de programación) y elige la versión que más se adapte a la resolución de pantalla de este.

En la siguiente imagen podemos observar varios diseños visto desde distintos dispositivos:

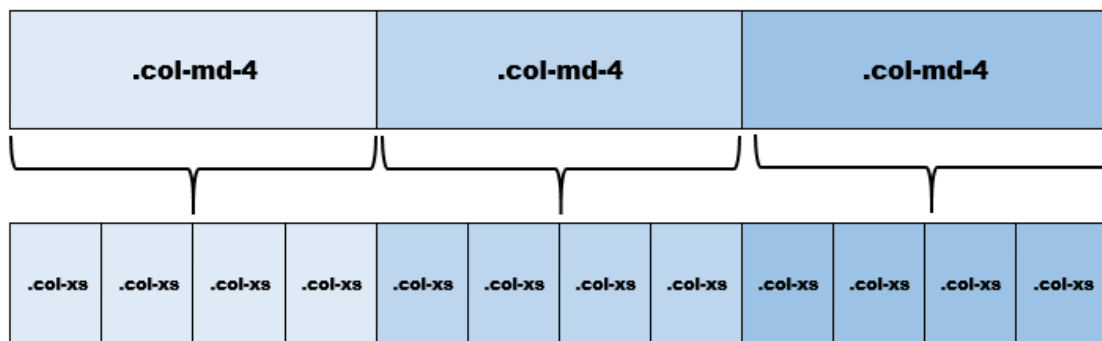


El sistema de rejillas (12 columnas)

Para hacer que un sitio web se adapte a todo tipo de pantalla, ya sea una tablet, teléfono inteligente o computadora de escritorio, **Bootstrap** utiliza un sistema de 12 **columnas** —llamadas también **rejillas o grids**— para adaptar el contenido de la página web a las distintas resoluciones y tamaños de los dispositivos que se pueden utilizar.

Por ejemplo, si diseñamos una página web basada en cuatro columnas, cuando accedemos a ella desde una tablet, el contenido puede reorganizarse a tres columnas, o a una sola columna si lo hacemos desde un teléfono inteligente.

En Bootstrap, ese sistema de columnas lleva un prefijo de clase compuesto por 4 diferentes tamaños, de acuerdo al dispositivo para el cual se esté programando. Por ejemplo, si tenemos una resolución **menor a 768 px**, estaríamos hablando de un teléfono inteligente, entonces le corresponde el prefijo de clase **.col-xs-**. Si tuviera una resolución **mayor o igual a 768 px**, en ese caso estaríamos hablando de una tablet, cuyo prefijo de clase sería **.col-sm-**. Para resoluciones **mayores o iguales a 992 px**, el prefijo sería **.col-md-**, correspondiente a resoluciones medias de computadoras de escritorio. Y, por último, para resoluciones **mayores o iguales a 1200 px**, el prefijo de la clase sería **.col-lg-** que corresponde a resoluciones mayores para computadoras de escritorio.



Descargar Bootstrap

Para descargar e instalar **Bootstrap**, tenemos que acceder a su página web oficial: <http://getbootstrap.com>. Una vez allí, podemos descargar el framework desde el botón central **Download Bootstrap** (en español, “descargar Bootstrap”) o bien desde el menú principal haciendo clic en **Getting started** (en español, “comenzando”).

De esta forma, se nos descargará un archivo comprimido que contiene las librerías necesarias para trabajar de manera independiente en nuestro servidor. El nombre del archivo es **bootstrap-X.X.X-dist.zip**, donde la letra **X** nos indica la versión.

Para trabajar localmente, simplemente tendremos que descomprimir este archivo en el directorio raíz de nuestro proyecto. Al descomprimir el archivo Zip que descargamos del sitio de Bootstrap, vamos a encontrar tres carpetas: **CSS**, **js** y **fonts**, las cuales contienen archivos con hojas de estilo, librerías de JavaScript e íconos gráficos con los cuales vamos a trabajar en nuestros proyectos de Bootstrap.



Otra manera de trabajar sería enlazar las librerías de Bootstrap por **CDN** (recomendado). Para eso, desde la página oficial tenemos un apartado denominado **Bootstrap CDN**, con los diferentes enlaces de las librerías. <http://getbootstrap.com/getting-started/>

También podemos descargar el código fuente del framework en el apartado **Source code** (en español, “código fuente”).

Content delivery network (CDN) —en español, “red de entrega de contenidos”— son servidores con distintas ubicaciones en todo el mundo que distribuyen un contenido estático en particular (archivos, imágenes, aplicaciones web, entre otros), permitiendo una actualización permanente y una reducción del tiempo de respuestas entre cliente-servidor.

Instalación

Una vez que descargamos el framework, sólo nos resta instalarlo. Tenemos varias formas de enlazar las librerías para comenzar a trabajar con **Bootstrap**.

Si lo hacemos por medio de CDN, tenemos que escribir el siguiente código entre las etiquetas `<head></head>` de nuestro documento HTML:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap.min.css">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap-theme.min.css">
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3
.4/js/bootstrap.min.js"></script>
```

Los enlaces anteriores son a servidores externos (CDN), los dos primeros son a los estilos **CSS** del framework y, el último, a la librería de Bootstrap, con la extensión **.js** (JavaScript).

En cambio, si descargamos el framework para alojarlo en nuestro servidor —ya sea un hosting contratado o nuestra computadora de manera local—, tendríamos que copiar primero los archivos que conforman la distribución de Bootstrap dentro de la carpeta de nuestro proyecto. Luego podemos enlazar, desde las etiquetas `<head></head>` de nuestro documento HTML, los dos directorios descargados (**js** y **css**) con sus correspondientes archivos y ubicar en el mismo directorio la carpeta **fonts**, ya que es utilizada por los estilos CSS. De esta forma, tendríamos acceso a las librerías y estilos de Bootstrap, como podemos apreciar en el siguiente ejemplo de código:

```
<link rel="stylesheet" href="css/
bootstrap.min.css">
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
<script src="js/bootstrap.min.js"></script>
```

Primera plantilla

Para probar el perfecto funcionamiento de Bootstrap, el editor y nuestro servidor local, vamos a realizar una **plantilla de muestra**.

En nuestra primera plantilla vamos a hacer un ejemplo sencillo que contendrá el texto “Este es un ejemplo básico de plantilla”, en un título `<h1></h1>`, y una etiqueta resaltada en azul con la misma leyenda, “Este es un ejemplo básico de plantilla”, como se muestra en el siguiente código:

```
<!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>Bootstrap Plantilla ejemplo</title>

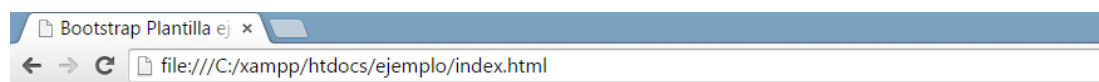
<!-- Bootstrap -->

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap.min.css">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap-theme.min.css">
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3
.4/js/bootstrap.min.js"></script>

</head>

<body>
  <h1> Este es un ejemplo básico de plantilla
</h1>
  <span class="label label-primary"> Este es un
ejemplo básico de plantilla </span>

</body>
</html>
```



Este es un ejemplo básico de plantilla

Este es un ejemplo básico de plantilla

Analizando el código anterior podemos observar que con la etiqueta `<!DOCTYPE html>` estamos definiendo que el tipo de archivo que creando es un documento HTML, y con el atributo `lang` que el lenguaje predeterminado del documento será español.

Posteriormente dentro del `<head></head>` incluimos una serie de etiquetas HTML con las cuales estamos indicando la codificación de caracteres utilizada y la compatibilidad del documento con Internet Explorer.

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
```

Luego incluimos la etiqueta `viewport`, para que el navegador sepa que tiene que adaptar la página a los diferentes dispositivos.

```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

A continuación agregamos las librerías necesarias para trabajar con Bootstrap, en este ejemplo estamos enlazando por medio de CDN los archivos [bootstrap.min.css](https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css) y [bootstrap-theme.min.css](https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js), [bootstrap.min.js](https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js).

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap.min.css">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap-theme.min.css">
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3
.4/js/bootstrap.min.js"></script>
```

Finalmente dentro del `<body></body>`, se incluye el contenido de la página como veremos a partir del capítulo 3. A modo de ejemplo solamente agregamos un título y una etiqueta resaltada.

```
<h1> Este es un ejemplo básico de plantilla </h1>
  <span class="label label-primary"> Este es un
ejemplo básico de plantilla </span>
```

¿Cómo adaptar nuestro diseño?

Hasta hace unos pocos años atrás, la mayoría de los usuarios accedían a internet mediante una computadora de escritorio o notebook. Hoy en día, con el advenimiento de las tablets y teléfonos inteligentes nos encontramos con que más usuarios emplean estos dispositivos para navegar por los diferentes sitios. Es por esto que a la hora de diseñar un sitio web debemos pensar que nuestro sitio web puede ser visualizado en diversos dispositivos, los cuales pueden tener distintos tamaños y resoluciones de pantalla.

Por este motivo, debemos diseñar un solo sitio y que el contenido se organice adaptándose automáticamente a todos los dispositivos, ya sea un smartphone, una tablet o una computadora de escritorio.

Para lograr esto, seguiremos la filosofía de trabajo **Mobile First**, que plantea que partamos por el diseño de la interfaz del sitio para una pantalla chica y posteriormente vayamos adaptando el diseño para un dispositivo de pantalla mediana y luego para un dispositivo de pantalla grande.

Para adaptar nuestro diseño, ya sea para una tablet, teléfono inteligente o computadora de escritorio, en Bootstrap contamos con medidas que contemplan estas diferentes resoluciones de dispositivos. Por lo tanto, no sólo tenemos que tener presente el ancho de nuestras 12 columnas para ubicar los distintos elementos en nuestro sitio web, sino que también tenemos que tener en cuenta el tipo de dispositivo en el cual se va a visualizar el sitio.

Por ejemplo: un menú principal, en una resolución de pantalla de un ancho de 1200 px, no se va a ver igual en una resolución de un ancho de 768 px. Y si queremos hacer la página web para ambas resoluciones, tenemos que saber bien cuando cambiará la forma en que se visualice el menú, es decir, programar que cuando la resolución de pantalla cambie de 1200 px a 768 px, el menú principal se visualice de otra forma acorde a esa resolución.

Para todo esto, el framework nos proporciona **clases** que nos facilitan la tarea cuando estamos haciendo una página web adaptable. Así, de acuerdo a la resolución del dispositivo con que se visualice la página web, esta tendrá un comportamiento distinto adaptándose a cada una de las resoluciones disponibles.

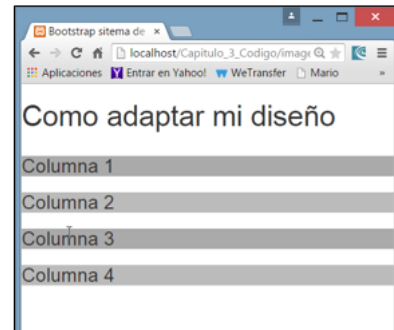
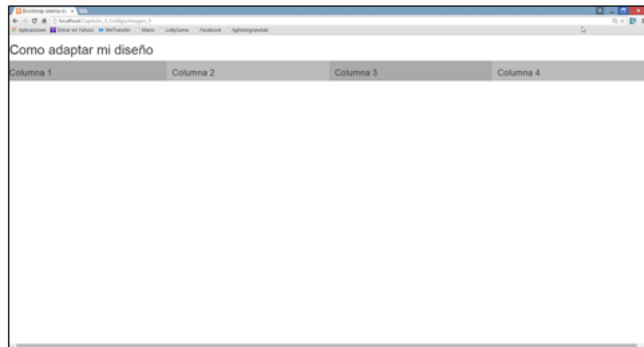
Siempre tenemos que tener presente con qué resolución vamos a trabajar y cuando necesitamos que el contenido cambie si la resolución es menor o mayor a la que estemos utilizando.

Si trabajamos para una resolución mayor o igual a 1200 px, trabajamos con el prefijo de la clase **.col-lg**. Y si, por ejemplo, tenemos 4 columnas, cuando baje de esa resolución —es decir, sea menor a 1200 px— se verán de forma vertical (una debajo de la otra) y no de forma horizontal (una al lado de la otra), como deberían verse normalmente.

```
<div class="row">

    <div class="col-lg-3" style="background-
color:#aaa" ><h3>Columna 1</h3></div>
    <div class="col-lg-3" style="background-
color:#bbb" ><h3>Columna 2</h3></div>
    <div class="col-lg-3" style="background-
color:#aaa"><h3>Columna 3</h3></div>
    <div class="col-lg-3" style="background-
color:#bbb" ><h3>Columna 4</h3></div>
```

</div>



La primera imagen se ve en una resolución de 1200 px y la otra en una resolución menor a 1200 px.

Esto mismo ocurriría si siguiéramos la lógica de Bootstrap, donde, como dijimos anteriormente, si la resolución es menor a 768 px (como en un teléfono inteligente), el prefijo de la clase debería ser **.col-xs**. Para mayor o igual a 768 px, el prefijo de la clase sería **.col-sm** y para mayores o iguales a 992 px, sería **.col-md**. Y como vimos en este último ejemplo si son mayores o iguales a 1200 px correspondería el prefijo de la clase **.col-lg**.

En la siguiente tabla se resumen los prefijos de clase que utiliza Bootstrap para establecer los puntos de cortes según el tipo de pantalla.

Prefijos de clase		
Tipo de dispositivo	Resolución	Prefijo de clase
Grandes (Large devices)	A partir de 1200 px	.col-lg
Medio (Medium devices)	A partir de 992 px	.col-md
Tipo Tablet (Small devices)	A partir de 768 px	.col-sm
Dispositivos móviles (Extra small devices)	Menos de 768 px	.col-xs

Viewport

Para asegurarnos que nuestro sistema esté optimizado y se adapte a dispositivos móviles, es necesario agregar en las etiquetas `<head></head>` de nuestra página HTML la etiqueta meta viewport.

Una **etiqueta meta** (en inglés, **meta-tag**) es la que se utiliza para agregar información de nuestro sitio web a los motores de búsqueda o navegadores, y solo es visible para los programadores (se mantiene oculta para los usuarios comunes).

```
<head>
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
</head>
```

En este ejemplo de código le estamos indicando al navegador, que adapte el contenido al ancho total (100%) de la pantalla del dispositivo (**width=device-width**). Y con **initial-scale** estamos controlando el nivel de zoom. Para deshabilitar el zoom, simplemente hay que agregarle **user-scalable=no**; con esto no se podrá hacer zoom en nuestra página web.

De esta forma, nuestro documento HTML listo para comenzar a trabajar en Bootstrap quedaría así:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1">
    <title>Bootstrap </title>

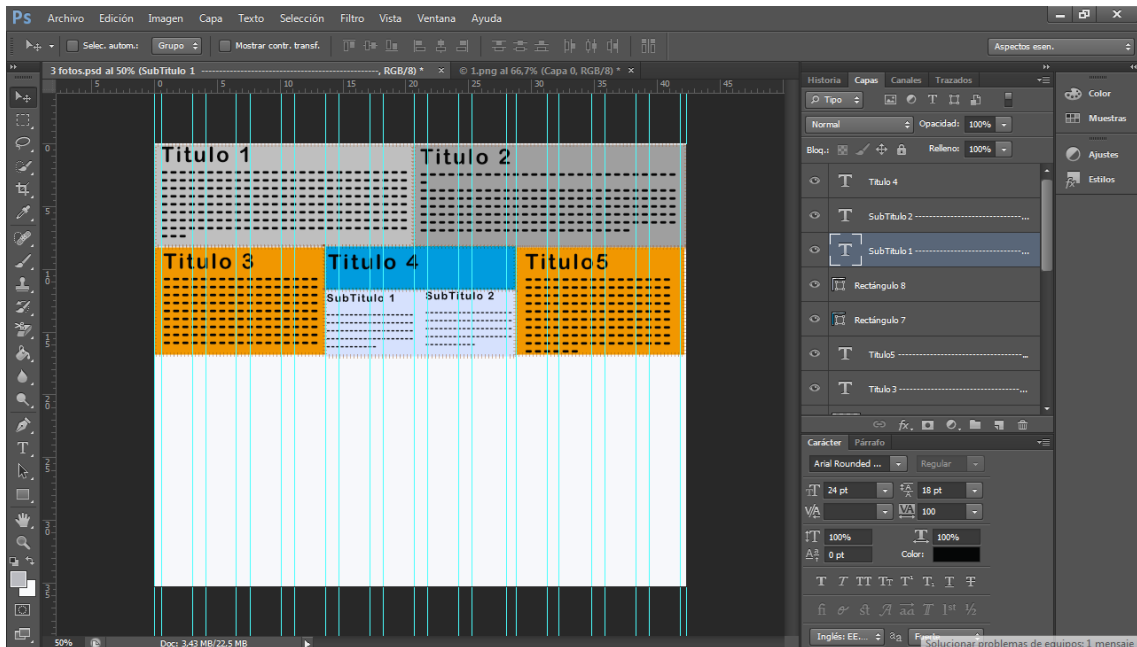
    <!-- Bootstrap -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap.min.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.4/css/bootstrap-theme.min.css">
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3
.4/js/bootstrap.min.js"></script>
  </head>
  <body>

  </body>
</html>
```

Ejemplo práctico

Vamos a ver ahora un ejemplo de cómo implementar el sistema de columnas en Bootstrap a partir de un diseño.

Supongamos que tenemos el siguiente diseño que nos fue entregado por un diseñador gráfico realizado en Adobe Photoshop:



Como podemos observar en el diseño, hay definidas claramente dos secciones: una parte superior, que podemos llamar **A**, y una parte inferior, a la que podríamos llamar **B**.

Estas dos secciones, a su vez, tienen varias columnas. Por un lado, la sección A está dividida en 2 columnas. Por otro lado, la sección B se divide en 3 columnas, y a su vez la columna del medio (Título 4) tiene dos columnas donde irían los subtítulos.

Para realizar la maquetación en Bootstrap, comenzaremos con la sección A, que tiene solamente una fila y dos columnas. Entonces, si las filas son 12 y necesitamos nada más que 2, el resultado de dividir 12 sobre 2 nos da **6**, que es la longitud que debe tener la clase **class="col-md-6"**. Así, quedaría de la siguiente forma:

```
<div class="row">
  <div class="col-md-6" style="background-color:#aaa" >
    <h3>Título 1</h3>
    <p>Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especimen.</p>
  </div>
  <div class="col-md-6" style="background-color:#aaa" >
    <h3>Titulo 2</h3>
    <p>Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especimen.</p>
  </div>
</div>
```

```

</div>
<div class="col-md-6" style="background-
color:#bbb" >
    <h3>Título 2</h3>
    <p>Lorem Ipsum es simplemente el texto de
relleno de las imprentas y archivos de texto.
Lorem Ipsum ha sido el texto de relleno estándar
de las industrias desde el año 1500, cuando un
impresor (N. del T. persona que se dedica a la
imprenta) desconocido usó una galería de textos y
los mezcló de tal manera que logró hacer un libro
de textos especimen.</p>
</div>
</div>

```

En este ejemplo estamos utilizando el prefijo de la clase **md** para resoluciones mayores o iguales a 992 px (computadoras de escritorio). También le estamos agregando un **estilo** para que se pueda apreciar mejor el ejemplo acorde al diseño. Tengan presente que por una cuestión de practicidad no utilice un archivo separado para el estilo CSS, en la siguiente imagen podemos observar la primera parte del ejemplo (**Seccion A**):



Al ser la resolución mayor o igual a 992 px (**col-md-**), si achicamos el navegador y lo llevamos a una resolución menor a 992 px, veremos como el sistema de columnas en este ejemplo 2 colapsa; es decir, se ve de forma diferente en resoluciones menores (en este caso, una columna debajo de la otra).

En resoluciones **menores a 992px**, cuando nuestro sistema de columnas es del tipo **col-md-**, colapsa para que pueda ser visualizado de otra forma en otras resoluciones como podemos observar en la siguiente imagen:



Para la **sección B** de nuestro ejemplo, como dijimos anteriormente, necesitamos 3 columnas y una fila, pero una de las columnas, la del medio, en su interior contiene otras dos columnas más a modo de subtítulo.

Entonces, si son 3 columnas sobre 12 filas disponibles, vamos a necesitar que nuestras filas sean de 4, porque 12 sobre 3 nos da como resultado 4. Observemos el siguiente código:

```
<div class="row">
  <div class="col-md-4" style="background-
color:#FFBF00" >
    </div>
    <div class="col-md-4" style="background-
color:#00BFFF" >

      </div>
      <div class="col-md-4" style="background-
color:#FFBF00" >
        </div>
</div>
```

Para terminar nuestro diseño, ahora solo nos resta agregarle las columnas necesarias a la columna del medio.

La columna del medio en su interior tiene otras dos columnas que hacen de subtítulo, tal cual nos fue pedido desde el diseño. Entonces, para esto vamos a necesitar dos columnas de 6, ya que 6 es el resultado de dividir 12 sobre 2. A esto se lo llama **anidar columnas**, cuando tenemos que incluir otra clase **row** dentro de una clase **row** ya definida.

```
<div class = "row">
  <div class="col-md-6" style="background-
color:#E0F2F7">
    <h5>SubTítulo </h5>
    <p>Lorem Ipsum es simplemente el texto de
relleno de las imprentas y archivos de texto. </p>
```



```
</div>
<div class="col-md-6" style="background-
color:#E0F2F7">
    <h5>SubTítulo </h5>
    <p>Lorem Ipsum es simplemente el texto de
relleno de las imprentas y archivos de texto. </p>
</div>
</div>
```

```
<div class="content">
    <div class="row">
        <div class="col-md-6"
style="background-color:#aaa" >
            <h3>Título 1</h3>
            <p>Lorem Ipsum es simplemente el
texto de relleno de las imprentas y archivos de
texto. Lorem Ipsum ha sido el texto de relleno
estándar de las industrias desde el año 1500,
cuando un impresor (N. del T. persona que se
dedica a la imprenta) desconocido usó una galería
de textos y los mezcló de tal manera que logró
hacer un libro de textos especimen.</p>
        </div>
        <div class="col-md-
6"style="background-color:#bbb" >
            <h3>Título 2</h3>
            <p>Lorem Ipsum es simplemente el
texto de relleno de las imprentas y archivos de
texto. Lorem Ipsum ha sido el texto de relleno
estándar de las industrias desde el año 1500,
cuando un impresor (N. del T. persona que se
dedica a la imprenta) desconocido usó una galería
de textos y los mezcló de tal manera que logró
hacer un libro de textos especimen.</p>
        </div>
    </div>

    <div class="row">
        <div class="col-md-4"
style="background-color:#FFBF00" >
            <h3>Título 3</h3>
            <p>Lorem Ipsum es simplemente el
texto de relleno de las imprentas y archivos de
```

```

texto. Lorem Ipsum ha sido el texto de relleno
estándar de las industrias desde el año 1500,
cuando un impresor (N. del T. persona que se
dedica a la imprenta) desconocido usó una galería
de textos y los mezcló de tal manera que logró
hacer un libro de textos especimen.</p>
    </div>
    <div class="col-md-
4"style="background-color:#00BFFF" >
        <h3>Título 4</h3>
        <div class ="row">
            <div class="col-md-6"
style="background-color:#E0F2F7">
                <h5>SubTítulo </h5>
                <p>Lorem Ipsum es
simplemente el texto de relleno de las imprentas y
archivos de texto. </p>
            </div>
            <div class="col-md-6"
style="background-color:#E0F2F7">
                <h5>SubTítulo </h5>
                <p>Lorem Ipsum es
simplemente el texto de relleno de las imprentas y
archivos de texto. </p>
            </div>
        </div>
    </div>
    <div class="col-md-
4"style="background-color:#FFBF00" >
        <h3>Título 5</h3>
        <p>Lorem Ipsum es simplemente el
texto de relleno de las imprentas y archivos de
texto. Lorem Ipsum ha sido el texto de relleno
estándar de las industrias desde el año 1500,
cuando un impresor (N. del T. persona que se
dedica a la imprenta) desconocido usó una galería
de textos y los mezcló de tal manera que logró
hacer un libro de textos especimen.</p>
    </div>

</div>
</div>

```

Este es el ejemplo final terminado:



En resumen: Bootstrap nos simplifica la vida a los programadores que no sabemos tanto de diseño, o bien si queremos realizar un trabajo **front-end** rápido y que quede bien sin tener que lidiar trabajando con estilos, y por sobre todo que sea adaptable a todo tipo de pantalla.

En la página oficial de Bootstrap contamos con numerosos ejemplos de componentes, tanto para formularios como para un proyecto en general: <http://getbootstrap.com/components/>

Y pronto estará disponible la **versión 4** con muchas mejoras, nuevos estilos y componentes.

Toda la información fue extraída del libro **Bootstrap** (2015) de editorial Users que es de mi autoría.

Actividades

1. Realizar un formulario, que utilice los componentes de formularios de Bootstrap. <http://getbootstrap.com/components/#input-groups>
2. Realizar cualquier ejercicio de los vistos anteriormente, que incluyan componentes HTML, con Bootstrap.

Autoevaluación

1. ¿Qué es Bootstrap?
2. ¿Qué es el diseño web adaptable o responsive?
3. ¿Qué ventaja nos ofrece el framework de Bootstrap?
4. ¿Cómo funciona el sistema de columnas del framework?
5. ¿Se puede adaptar un diseño realizado anteriormente a Bootstrap?
6. ¿Qué significa enlazar una librería por CDN?
7. Explique cómo agregar las librerías de Bootstrap en el documento HTML.

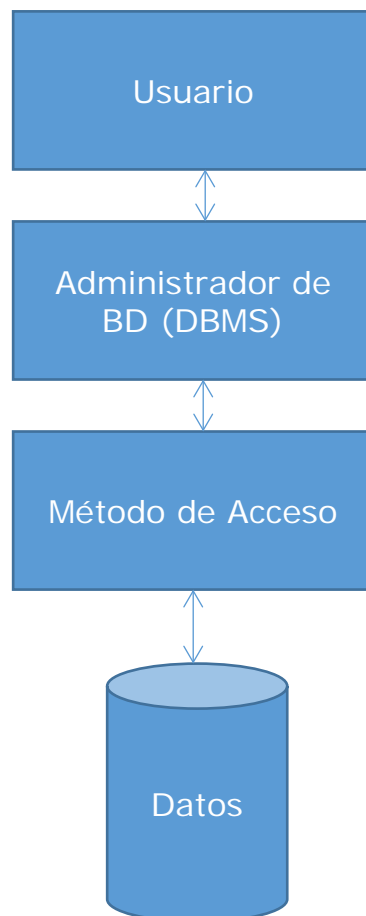
Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 12- Base de datos - MySQL.

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, siendo este un componente electrónico, por tanto se ha desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD (del inglés Database Management System o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.



Definición

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de

datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

Muchos **SGBD** proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base de datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por **Sun Microsystems** en 2008, y ésta a su vez fue comprada por **Oracle Corporation** en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

MySQL es usado por muchos sitios web grandes y populares, como **Wikipedia**, **Google** (aunque no para búsquedas), **Facebook**, **Twitter**, **Flickr**, y **YouTube**.

MySQL es muy utilizado en aplicaciones web, como **Joomla**, **Wordpress**, **Drupal** o **phpBB**, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a **PHP**, que a menudo aparece en combinación con **MySQL**.

Sus inicios

MySQL fue creado por una compañía sueca MySQL AB en 1995. Los desarrolladores de la plataforma fueron Michael Widenius (@montywi), David Axmark y Allan Larsson. El objetivo principal era ofrecer opciones eficientes y fiables de gestión de datos para los usuarios domésticos y profesionales. Más de la mitad de una docena de versiones alfa y beta de la plataforma fueron lanzados en 2000. Estas versiones son compatibles con casi todas las principales plataformas.

Open Source

Originalmente la propiedad era de **MySQL AB**, la plataforma de código abierto fue a partir del 2000 y comenzó a seguir los términos de GPL. Ser open source (código abierto) dio lugar a una disminución significativa de los ingresos, sin embargo, se

recuperó con el tiempo. La naturaleza de código abierto de MySQL ha hecho que terceros desarrolladores contribuyan al proyecto.

Expansión de Negocios

MySQL fue ganado constante popularidad entre los usuarios domésticos y profesionales, y en 2001, la plataforma tenía 2 millones de instalaciones activas. En 2002, la compañía amplió su alcance y abrió sede en EE.UU. Además de la sede en Suecia. El mismo año, se anunció que el número de miembros de las plataformas era más de 3 millones de usuarios con unos ingresos por valor de \$ 6.500.000.

Cambio en la Estrategia

La plataforma continúa ganando popularidad en relación a fines de 2003, se podría presumir el ingreso total de \$ 12 millones, con 4 millones de instalaciones activas. En 2004, la empresa decidió centrarse más en los ingresos del usuario final en lugar de cuota de licencia por instalación. La estrategia resultó ser rentable y el año terminó con un ingreso neto de \$ 20 millones.

Adquisición de Innobase por Oracle

En el 2005, Oracle compró Innobase, la empresa que gestiona el almacenamiento backend Innobase de MySQL. Este motor de almacenamiento de MySQL permite la implementación de funciones importantes, como las transacciones y claves foráneas. El mismo año, MySQL Network desarrolla en las líneas de RedHat Red, esto dio lugar a MySQL 5, que amplió considerablemente el conjunto de características disponibles para los usuarios de la empresa. Después de años, se renovó el contrato entre MySQL y Innabose.

MySQL adquirida por Sun Microsystems

En enero de 2008, MySQL fue adquirida por Sun Microsystems por \$ 1 mil millones. La decisión fue criticada por Michael Widenius y David Axmark, los co-fundadores de MySQL AB. En ese momento MySQL ya era la primera opción de las grandes corporaciones, bancos y empresas de telecomunicaciones. El CEO de Sun Microsystems, Jonathan Schwartz, llamó a MySQL “la acción de la raíz” (“The root stock”) de la economía de Internet.

La adquisición de Sun y MySQL por Oracle

La adquisición de MySQL por Sun no resultó muy fructífera y en abril de 2009, se llegó a un acuerdo entre Sun Microsystems y Oracle Corporation, según la cual Oracle fue compra a Sun Microsystems, junto con los derechos de autor y marca registrada de MySQL. El acuerdo fue aprobado por el gobierno de los EE.UU el 20 de Agosto de 2009. Como resultado de la petición en línea iniciada por uno de los fundadores de MySQL Monty Widenius, Oracle se enfrentó a algunas complicaciones legales con la Comisión Europea. Sin embargo, los problemas se resolvieron y en Enero de 2010, la adquisición de MySQL por parte de Oracle se convirtió oficial.

MySQL Forks

Michael Widenius dejó Sun Microsystems después de que fuera adquirida por Oracle y con el tiempo desarrolló una copia (mini-copia) (fork) de MySQL llamado MariaDB. Los Forks son proyectos relacionados que se pueden considerar mini-versiones de MySQL estándar. Hasta la fecha, varios de tales versiones se han puesto en marcha cuyo objetivo es proporcionar una funcionalidad específica. Maria DB es un fork de propiedad comunitaria que significa que no tendría restricciones de licencia habituales que tiene la versión estándar de MySQL. Es compatible con MySQL binary Library de modo que no hay ninguna diferencia entre los comandos y las API.

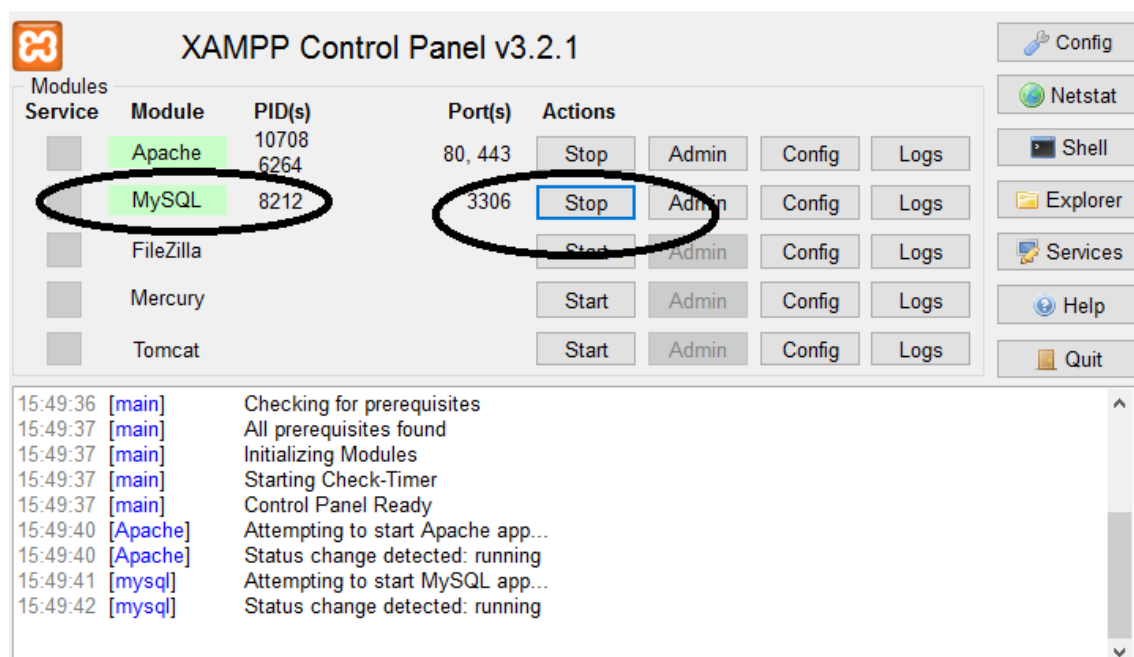
MySQL and Cloud Computing

Las versiones antiguas de MySQL sólo se han desarrollado para máquinas convencionales. Sin embargo, con el advenimiento de la computación en la nube (cloud computing), MySQL se hizo también compatible con diversos servicios de computación en la nube, como Amazon EC2. Varios modelos de implementación se han utilizado para la implementación de MySQL en plataformas de cloud computing. Tal vez el más popular de estos modelos es 'imágenes en máquinas virtuales', que permite el uso de una imagen ya hecha máquina donde esta MySQL preinstalado.

Un segundo modelo de cloud computing es gestionar MySQL cloud hosting donde la base de datos no está disponible como un servicio pero que está alojado y administrado en el nombre del propietario. Este modo, sin embargo, se ofrece sólo para algunas empresas. Con la expansión de la computación en nube y la tecnología relacionada, también se espera que las versiones de **MySQL** para la computación en nube puedan aumentar en número.

Configuración (primeros pasos)

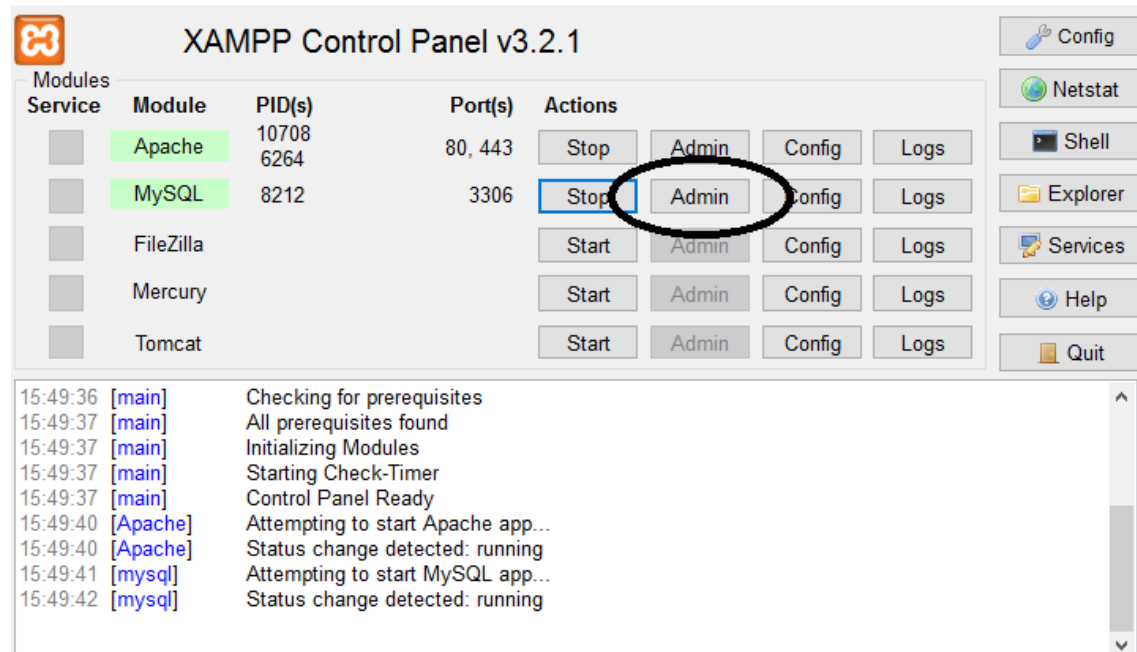
Para comenzar a utilizar MySQL, debemos tener los servicios instalados. Si instalaron correctamente XAMPP, esos servicios se instalan solos. Lo que sí debemos verificar es que el servicio este activo. Para esto vamos al control panel de XAMPP y verificamos que el servicio este activo, caso contrario hacemos clic en el botón **Start** para iniciar el mismo. Si está activo debemos tener una imagen similar a la siguiente figura:



Luego debemos ir al administrador para configurar el usuario y la base de datos.

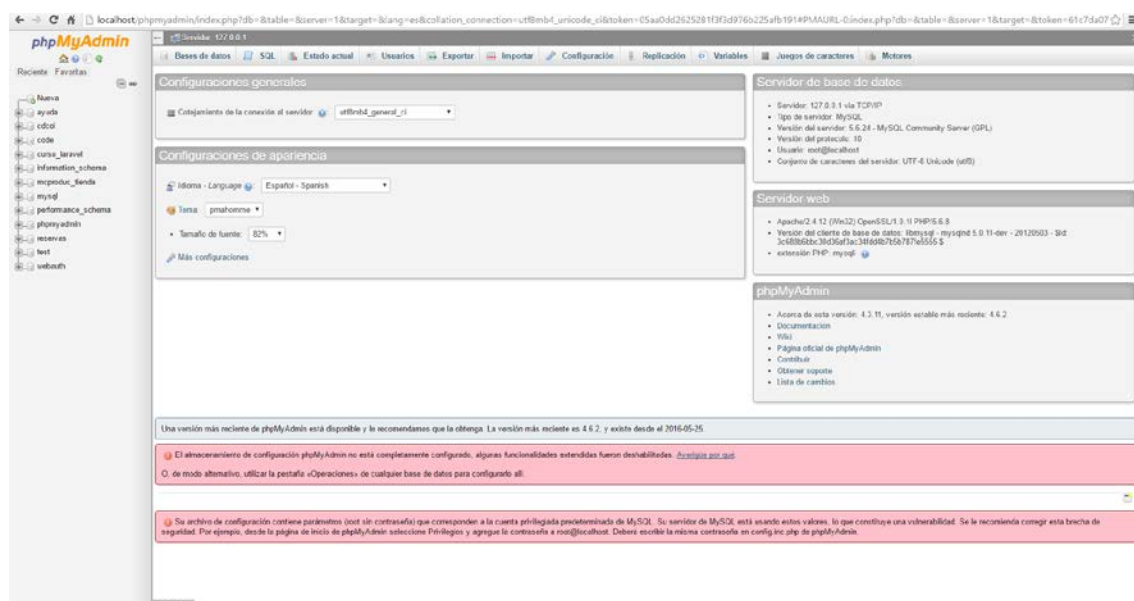
Podemos utilizar un administrador propio como puede ser MySQLFront, que lo podemos bajar de mi página <http://www.murcielagoblanco.com.ar/index.php/descargas> o bien utilizar PHPMyAdmin, que ya viene disponible en el paquete que instalamos con XAMPP. Hay muchos más, cualquiera es bueno.

Nosotros vamos a utilizar PHPMyAdmin, ya que lo tenemos instalado. Para esto podemos ir al panel de control de XAMPP y hacer clic en la opción admin:



Otra opción, más rápida (si el servicio esta iniciado) es desde el navegador, ir directamente al sitio: <http://localhost/phpmyadmin>

Y de acuerdo a la versión que tengamos instalada, vamos a ver algo parecido a esto (digo “parecido” porque cambian algunas cosas de acuerdo a la versión):



Desde el administrador vamos a poder administrar usuarios, crear base de datos, tablas, hacer consultas entre otras cosas. Nosotros sólo vamos a hacer la mayoría de las cosas desde un script en PHP, pero tengan en cuenta que muchas cosas se pueden hacer simplemente desde el administrador.

Para comenzar a utilizar base de datos, es decir conectar desde un script vamos a necesitar crear un usuario que cuente con todos los privilegios.

De acuerdo a la versión que tengamos instalada del administrador, debemos dirigirnos a la solapa “usuarios” o “privilegios” para poder crear un usuario nuevo.



Vista global de usuarios

Usuario	Servidor	Contraseña	Privilegios globales	Conceder	Acción
<input type="checkbox"/> cualquiera	%	--	USAGE	No	Editar los privilegios Exportar
<input type="checkbox"/> cualquiera	localhost	No	USAGE	No	Editar los privilegios Exportar



Vista global de usuarios

Usuario	Servidor	Contraseña	Privilegios globales	Grupo de usuario	Conceder	Acción
<input type="checkbox"/> cualquiera	%	--	USAGE		No	Editar los privilegios Exportar
<input type="checkbox"/> cualquiera	localhost	No	USAGE		No	Editar los privilegios Exportar

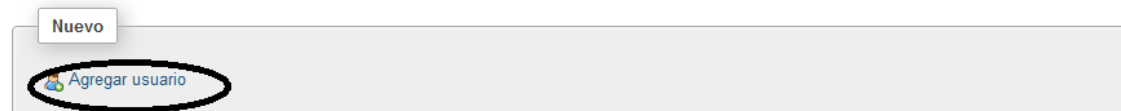
Generalmente el usuario que viene por defecto es **root** y de contraseña nada, es decir sin contraseña. Algunos caso me ha tocado usuario: “**root**” y contraseña: “**1111**”. Pero como dije anteriormente depende mucho de la versión que tengamos instalada.

Para crear un usuario nuevo vamos a “**Vista global de usuarios**” y hacemos clic en **Nuevo -> Agregar usuario**.

Vista global de usuarios

	Usuario	Servidor	Contraseña	Privilegios globales	Grupo de usuario	Conceder	Acción
<input type="checkbox"/>	cualquiera	%	--	USAGE		No	Editar los privilegios Exportar
<input type="checkbox"/>	cualquiera	localhost	No	USAGE		No	Editar los privilegios Exportar
<input type="checkbox"/>	pma	localhost	No	USAGE		No	Editar los privilegios Exportar
<input type="checkbox"/>	root	127.0.0.1	No	ALL PRIVILEGES		Sí	Editar los privilegios Exportar
<input type="checkbox"/>	root	:::1	No	ALL PRIVILEGES		Sí	Editar los privilegios Exportar
<input type="checkbox"/>	root	localhost	No	ALL PRIVILEGES		Sí	Editar los privilegios Exportar

↑ ☐ Marcar todos Para los elementos que están marcados: Exportar



Al hacer clic se nos va a abrir una ventana nueva con toda la configuración del usuario nuevo:

Agregar usuario

Información de la cuenta

Nombre de usuario:

Servidor:

Contraseña:

Debe volver a escribir:

Generar contraseña:

Base de datos para el usuario

☐ Crear base de datos con el mismo nombre y otorgar todos los privilegios

☐ Otorgar todos los privilegios al nombre que contiene comodín (usuarname_%)

Privilegios globales

Ver los nombres de los privilegios de rights están representados en right

Datos

- ☐ SELECT
- ☐ INSERT
- ☐ UPDATE
- ☐ DELETE
- ☐ TRIGGER

Estructura

- ☐ CREATE
- ☐ ALTER
- ☐ DROP
- ☐ RENAME
- ☐ CREATE TEMPORARY TABLES
- ☐ SHOW VIEW
- ☐ CREATE VIEW
- ☐ ALTER VIEW
- ☐ EXECUTE
- ☐ CREATE VIEW
- ☐ TRIGGER

Administración

- ☐ SHUTDOWN
- ☐ RESET
- ☐ ANALYZE
- ☐ BACKUP
- ☐ RECOVER
- ☐ LOCK TABLES
- ☐ UNLOCK TABLES
- ☐ REPLICATION CLIENT
- ☐ CREATE USER

Límites de recursos

Para el campo de parámetro de valor (valor a 0 (cero), reemplaza el 0 por el valor)

MAX_CONNECTIONS_PER_HOUR:

MAX_QUERIES_PER_HOUR:

MAX_UPDATES_PER_HOUR:

MAX_CONNECTIONS_PER_HOUR:

MAX_USER_CONNECTIONS:

Continuar

En el campo **nombre de usuario** asignamos un nombre al usuario nuevo, en **servidor** si es local debemos escribir **localhost** o bien desde el combo elegimos la opción **local** y agregamos una **contraseña**.

En **Base de datos para el usuario**, hacemos clic en la opción **Crear base de datos con el mismo nombre y otorgar todos los privilegios**. Y en **Privilegios globales** hacemos clic en **Marcar todos**, de esta forma le damos todos los privilegios al usuario nuevo.

De esta forma al hacer clic en **continuar**, ya tendremos nuestro usuario creado. **Recordemos el usuario y contraseña porque es lo que vamos a utilizar al crear el script de conexión a la base.**

Actividades

1. Iniciar el servicio de base de datos MySQL y entrar al administrador de la misma.
2. Crear un usuario nuevo con todos los privilegios de root.

Autoevaluación

1. ¿Qué es MySQL?
2. ¿Actualmente que empresa es propietaria de MySQL?
3. ¿Defina las formas de entrar al administrador de la base?
4. ¿Es necesario crear un usuario nuevo al ingresar?
5. Diga por qué a la respuesta de la pregunta 4.

Tecnicatura Superior en Análisis de Sistemas

PROGRAMACIÓN V

Tema 13- PHP - MySQL.

Una vez que ya hemos creado nuestro usuario de la base de datos, podemos comenzar a armar el script en PHP que nos va a servir para conectar a la base de datos, crear tablas, insertar datos, modificar y eliminar datos.

Lo primero que vamos a utilizar es la función **mysql_connect()**, que debemos pasarle por parámetros el servidor, usuario y contraseña de nuestra base de datos. Lo que hace esta función es abrir la conexión al servidor **MySQL**.

Esta función quedo obsoleta a partir de la versión 5.5 de PHP y ya fue eliminada de la versión 7. La forma de utilizarla es:

```
$conexion =  
mysql_connect("localhost","usuario","pass");
```

Creo una variable llamada **\$conexion**, y le paso la conexión al servidor, con los datos que tenemos del mismo.

Si utilizamos la versión 7 de PHP o queremos probar la nueva forma de conectar creada a partir de la versión 5.5 hacemos lo siguiente:

```
$conexion = mysqli_connect("localhost", "usuario",  
"contraseña", "bd");
```

Podemos observar que a diferencia de la función **mysql_connect**, **mysqli_connect** agrega un parámetro más (optativo) que es el nombre de la base de datos a la que vamos a conectar.

Si queremos probar la conexión al servidor, para saber si es correcta, debemos preguntar si se ha podido conectar, de la siguiente manera:

```
if (!$conexion){  
die('No se ha conectado: '.mysql_error());  
}  
else{  
echo "Se ha conectado al servidor";  
}
```

Esto seria, que si no se pudo conectar sale del script con la función **die()** y muestra el error de porque no se ha podido conectar con la función **mysql_error()**, caso contrario muestra un mensaje de que se ha podido conectar.

Para la creación de la base de datos también lo hacemos con una pregunta, si ya fue creada seguramente nos va a dar un error, ya que no pueden haber dos bases de datos con el mismo nombre. Y si se produce también, algún otro tipo de error nos lo va a decir con la función **mysql_error()**.

```

if (mysql_query("CREATE DATABASE Empresa",
$conexion))
{
    echo "Se ha creado la bd";
}
else{
    echo "No se ha podido crear la bd por el
siguiente error: ".mysql_error();
}

```

Si utilizamos la versión más **nueva de PHP** debemos cambiar el `mysql_query` por **`mysqli_query`**. Y la variable **`$conexion`** debe ir antes de la consulta SQL. Por ejemplo:

```

mysqli_query($conexion,"CREATE DATABASE Empresa");

```

Debemos tener presente que este script que estamos creando se puede ejecutar todo junto o por partes en archivos separados, eso como más nos guste.

Luego para crear la tabla hacemos lo siguiente:

```

mysql_select_db("Empresa",$conexion);

$sql = "CREATE TABLE Clientes
(
Id int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(Id),
Nombre varchar(60),
Direccion varchar(60),
Telefono varchar (20),
Email varchar (60)
)";

mysql_query($sql, $conexion);

```

Primero preparo la petición al servidor, que en este caso sería crear una tabla. Selecciono la base de datos en la cual voy a trabajar y creo una variable **`$sql`** donde voy a crear la consulta. Por ultimo ejecuto la petición al servidor con la función **`mysql_query()`**.

Para insertar datos es prácticamente lo mismo, selecciono la base, creo una consulta y la ejecuto:

```
mysql_select_db("Empresa",$conexion);

mysql_query("INSERT INTO Clientes(Nombre,
Direccion, Telefono, Email)
VALUES ('Microsoft', 'Bouchard 800', '46666666',
'microsoft@microsoft.com')");

mysql_query("INSERT INTO Clientes(Nombre,
Direccion, Telefono, Email)
VALUES ('Google', 'Rivadavia 1040', '46666666',
'google@gmail.com')");

mysql_query("INSERT INTO Clientes(Nombre,
Direccion, Telefono, Email)
VALUES ('Apple', 'Esmeralda 1800', '46666666',
'apple@icloud.com')");
```

En el caso de que quisiéramos hacer una actualización o eliminar algún dato, debemos hacer lo mismo, crear la consulta y ejecutarla.

Para finalizar, debemos cerrar la conexión con el servidor.

```
mysql_close($conexion);
```

En el caso de que usemos una versión más nueva de PHP:

```
mysqli_close($con);
```

Les paso un ejemplo completo de un script terminado y listo para ejecutarse:

```
//Conexión al servidor:
$conexion =
mysql_connect('localhost','usuario','pass');

    if (!$conexion){
        die ('No se ha podido conectar! '
.mysql_error());
    }else{
        echo('Se ha establecido la conexión al
servidor!');
    }
Crea base:
```



```

if (mysql_query('CREATE DATABASE AGENDA_II',
$conexion)){
    echo 'La base se ha creado correctamente';
}else{
    echo 'No se ha creado la base por:
'.mysql_error();
}
//Pregunta si esta creada sino la crea:
if(!mysql_select_db('AGENDA_II', $conexion)){
    if (mysql_query('CREATE DATABASE
AGENDA_II', $conexion)){
        echo 'La base se ha creado
correctamente! <br>';
    }else{
        echo 'No se ha creado la base por:
'.mysql_error();
    }
}else{
    echo 'La base Agenda ya ha sido creada con
anterioridad!';
}
//Creo tabla:
mysql_select_db("AGENDA_II",$conexion);
$sql = "CREATE TABLE mi_agenda
(
    personaID int NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(personaID),
    Nombre varchar(15),
    Apellido varchar(15),
    Edad int,
    Telefono int
)";
mysql_query($sql, $conexion);
Hago un insert:
mysql_select_db("AGENDA_II",$conexion);
mysql_query("INSERT INTO mi_agenda(Nombre,
Apellido, Edad, Telefono)
VALUES ('Pepe', 'Rodriguez', 25, 156888999)");

mysql_query("INSERT INTO mi_agenda(Nombre,
Apellido, Edad, Telefono)
VALUES ('Alberto', 'Alvarez', 28,
155858999)");
Muestro:

```

```

mysql_select_db("AGENDA_II",$conexion);
$consulta = mysql_query("SELECT * FROM
mi_agenda");

while ($fila = mysql_fetch_array($consulta)){
    echo $fila['Nombre']." ".$fila['Apellido']."
    ".$fila['Edad']." ".$fila['Telefono'];
    echo "<br>";
}
//Actualizar:
mysql_select_db("AGENDA_II",$conexion);

mysql_query("UPDATE miAgenda SET Edad = '21' WHERE
Nombre = 'Alberto' AND Apellido = 'Alvarez'");
//Eliminar:
mysql_select_db("AGENDA_II",$conexion);

mysql_query("DELETE from miAgenda WHERE Nombre =
'Alberto' AND Apellido='Alvarez'");

mysql_close($conexion);

```

De este último script debemos tener en cuenta que si queremos mostrar datos, podemos hacerlo mediante un ciclo **while** utilizando un **array** (`mysql_fetch_array()`) y luego mostramos los datos a medida que recorre el ciclo. También esto podemos utilizarlo creando una tabla. Esta función de MySQL (**`mysql_fetch_array()`**) recupera una fila de resultados como un array, eso nos permite recorrerlo e ir mostrando los datos que contiene el mismo.

Para versiones más nuevas de PHP utilizaremos **`mysqli_fetch_array()`**.

Actividades

1. Crear un script de conexión a una base de datos, creando la misma e informando si fue creada con éxito o si se produjo algún error.
2. A la base de datos anterior agregarle una tabla e insertar datos en la misma.
3. Mostrar los datos de la base en una tabla HTML, utilizando una clase de Bootstrap.

Autoevaluación

1. ¿Qué función se utiliza en MySQL para crear una conexión al servidor?
2. ¿Qué función se utiliza para capturar un error en MySQL?
3. ¿Qué función se utiliza para crear una consulta en MySQL en una versión superior a la 5.6 de PHP?
4. ¿Qué función se utiliza para cerrar la conexión al servidor?