Framework FrontEnd

Programación Web Avanzada, Programación

índice

¿Qué es un Framework?

Para comenzar a entender Bootstrap, primero revisaremos algunos conceptos importantes. Particularmente un framework Front-End.

Según Wikipedia un Framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

En otras palabras: Sirve para poder estructurar de una mejor manera nuestro código, escribir menos líneas. La mayoría de las veces podemos reutilizar código en otros proyectos.

¿Por qué debemos elegir un framework a la hora de ponernos a programar?

Evitar escribir código repetitivo: Los proyectos tienen partes comunes para su funcionamiento, ejemplo acceso a las BD, validación de formularios o seguridad, el framework nos evita tener que programar estas partes.

Utilizar buenas prácticas: Los frameworks están basados en patrones de desarrollo, normalmente MVC (Modelo-Vista-Controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario. Vamos, que gracias a ellos, lo tenemos todo más ordenado.

Permitir hacer cosas avanzadas que nosotros no haríamos: Está claro que un framework siempre te va permitir hacer cosas de una manera fácil y segura, que a uno te costaría mucho tiempo hacerlas.

Desarrollar más rápido: Ya que no necesitamos reescribir código, podemos hacer cosas más complejas y seguras, y al contar con un modelo MVC, tenemos todo más ordenado. Por estos motivos es que nos permiten un desarrollo más rápido.

En el campo del desarrollo web, las tecnologías se dividen técnicamente en dos, Backend y Frontend para poder mantener separadas las partes entre lo que se utiliza para la interacción con el usuario y el procesamiento y manipulación de los datos. ¿Qué es esto?, es muy sencillo de explicar.

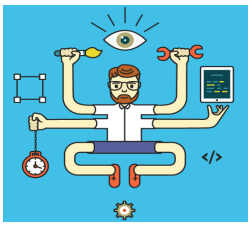
Frontend

El frontend son todas aquellas tecnologías que corren del lado del cliente, es decir, que corren del lado del navegador web.
El programador del FrontEnd debe de saber un poco de diseño, este se va a encargar de que la página no solo se vea bonita para el usuario, si no que sea cómoda de utilizar, cómoda de navegar e intuitiva.

Backend

El programador backend es aquel que se encuentra del lado del servidor, se encarga de lenguajes como PHP, Python, .Net, Java, etc.. Se encarga de interactuar con BD, verificar manejo de sesiones de usuarios, montar la página en un servidor, y desde éste "servir" todas las vistas que el FrontEnd crea.

Full Stack



Es un programador multiusos, responsable del desarrollo del proyecto, desde el montaje de los servidores, hasta el diseño con CSS. Tal y como evoluciona hoy en día la tecnología, es casi inviable tener una pata en cada lado (Cliente/Servidor/Arquitectura) y dominar todos. Generalmente, los programadores Full Stack están más centrados en una de las dos partes, es decir, dominan una de las partes y de la otra tienen nociones.

Teniendo estos conceptos más o menos claros podemos avanzar un poco más en detalle en los Frameworks Front End.

Un framework front-end es una herramienta que se integra con nuestro proyecto web para conseguir que el desarrollo front-end (interfaz, animaciones, etc.) sea más fácil, rápido y robusto.

Los Frameworks Front End son una herramienta útil para desarrolladores que recién comienzan o programadores con poco tiempo y poca experiencia en diseño. Algunos prefieren no usarlos para poder diferenciarse del resto en cuanto a diseño, ya que los frameworks tienen ciertas características que hacen notar cuando una página está desarrollada con un determinado frameworks por el estilo que estos manipulan.

Algunas de las ventajas que tiene utilizar este tipo de marcos es:

- Aunque no se use en el producto final se puede usar como prototipo ya que obtenemos resultados en menos tiempo.
- Permiten conocer a fondo la manera en que se implementan ciertos componentes
- Los frameworks son construidos por desarrolladores expertos, que cuidan cada detalle. Los elementos ya implementados que ofrece un framework permite crear sitios web seguros, que validan estándares y cargan rápido.

Como desventajas tenemos podemos mencionar que sin aprendizaje no podremos crear sitios realmente personalizados sobre la base que ofrece el framework. Se utilizan sin preocuparse por entender el código que están observando. Algo no menor es que los nombres de las variables suelen ser genéricos sin significado, lo que rompe con las buenas prácticas de programación a nivel de

comprensión de código. Además, las soluciones automatizadas tienden a utilizar demasiado código para implementar componentes sencillos. Y si nuestra tarea es modificar un sitio web ya en funcionamiento, incorporar un framework traerá muchos dolores de cabeza, por lo que no es recomendable aplicar un frameworks a sistemas funcionando.

Framework FrontEnd

Existen muchos Framework FrontEnd. Alguno de ellos los nombramos a continuación:

Foundation

(https://get.foundation/) Foundation es una familia de framework front-end responsive que facilitan el diseño de hermosos sitios web, aplicaciones y correos electrónicos responsive que se ven increíbles en cualquier dispositivo. Foundation es semántica, legible, flexible y completamente personalizable. Tiene cursos propios pero pagos.

Semantic UI

(https://semantic-ui.com/) Semantic es un framework de desarrollo que permite crear diseños flexibles utilizando HTML amigable, trata palabras y clases como conceptos intercambiables. Las clases usan la sintaxis de los lenguajes naturales como las relaciones sustantivo/modificador, orden de las palabras, y la pluralidad para vincular conceptos intuitivamente. Gran dependencia de Javascript. Orientado a Smartphone.

Pure

(https://purecss.io/) Un conjunto de módulos CSS pequeños y responsive que puede usar en cada proyecto web. Es ridículamente pequeño. El conjunto completo de módulos registra 3,8 KB * minimizado y comprimido . Diseñado teniendo en cuenta los dispositivos móviles. Pure es más minimalista y más acotado en cuanto a interfaz. Es utilizado por Yahoo.

Materialize

(https://materializecss.com/) Materialise es un marco front-end receptivo de código abierto que ofrece un diseño elegante de material listo para usar . Esta herramienta se basa en el lenguaje de diseño de materiales de Google . Es principalmente un marco UX que dicta cómo aparecerán y se comportarán los elementos en la página.

Bootstrap

(https://getbootstrap.com/) Bootstrap es un kit de herramientas de código abierto para desarrollar con HTML, CSS y JS. Realice rápidamente prototipos de sus ideas o cree toda su aplicación con nuestras variables y mixins Sass, sistema de cuadrícula sensible, componentes precompilados extensos y complementos potentes creados en jQuery.

Bootstrap

Bootstrap es un framework desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Cuenta con un Sistema GRID, lo que significa que nuestra web tendrá un ancho de página de 12 columnas, por lo tanto podremos añadir como máximo 12 columnas de tamaño 1 en nuestra página web.

Código basado en HTML5 y CSS3 que tiene en cuenta el formato de visualización de las principales herramientas de navegación (***responsive** design*): smartphones, tablets. Permite crear de forma sencilla webs de **diseño adaptable**, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es Open Source o código abierto, por lo que lo podemos usar de forma gratuita y sin restricciones.

Los navegadores no adoptan todos los mismos valores por defecto para los estilos de los elementos HTML. En función del navegador utilizado, esto puede provocar sorpresas al visualizar las páginas web. Por otro lado, algunos navegadores presentan defectos de consideración de algunos elementos. Por ello es que Bootstrap hace uso de `normalize.css`. Normalize es un pequeño fichero `CSS` que establece reglas para obtener una visualización idéntica independientemente del navegador empleado. En lugar de actuar con brutalidad como los resets `CSS`, que restablecen todos los valores a cero, normalize actúa de manera inteligente conservando lo que es útil y jugando sutilmente con los elementos. Aquí encontraréis las reglas aplicadas con su explicación en detalle.

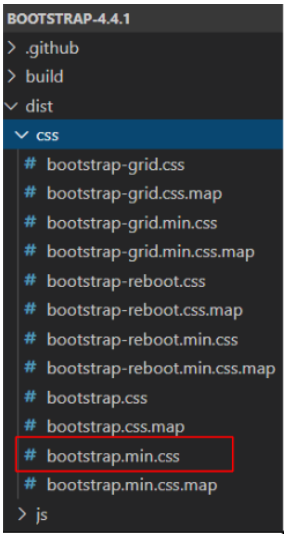
¿Por qué Bootstrap?

- Buena documentación;
- Garantía de evolución permanente;
- Mina de recursos variados en la web;
- Arquitectura basada en LESS (“Less is More”), una herramienta muy práctica que amplía las posibilidades de `CSS`
- también existe una adaptación en SASS (“Sass es una extensión de CSS que añade potencia y elegancia al lenguaje básico”)

Instalar Bootstrap 4.4.1

Instalar Bootstrap es simple: haga clic en el botón de descarga en el sitio web del frameworks . Hay dos botones disponibles:

- "Compiled CSS and JS" que te proporciona los archivos necesarios para ejecutar Bootstrap.
- "Source files", que además de proporcionar los archivos necesarios para su ejecución, contiene todos los archivos fuente.



Cuando descarga la biblioteca con el botón "Source files", obtiene un archivo `zip` que contiene un directorio bootstrap-4.4.1 que contiene varios archivos y carpetas, entre ellas una llamada `dist` como se muestra en la siguiente figura.

El fichero `bootstrap.css.map` permiten encontrar la ubicación original de una línea de código a partir del código minimizado. Esta funcionalidad se puede utilizar con las últimas versiones de Chrome y de Firefox. Estos ficheros no son indispensables para el funcionamiento de Bootstrap.
El corazón de este framework es un archivo `CSS`.

Luego debemos enlazar a nuestra página con el archivo `bootstrap.min.css` y ya podemos utilizar esta librería.

Para que Bootstrap funcione correctamente, las páginas `HTML`, que deben tener formato `HTML 5`. Es decir deben comenzar con:

```
<!DOCTYPE html>
...
```

Luego debe declarar al menos el archivo `bootstrap.min.css` (o `bootstrap.css`) en el encabezado de la página web:

```
<head>
  ...
  <link href="css/bootstrap.min.css" rel="stylesheet">
</head>
```

Después de estos pasos, todas las clases deberían ser accesibles. **¡No olvide modificar el enlace para tener en cuenta la ubicación de sus archivos!**

Ejemplo

La primera clase que podemos probar es class="container" (esta muestra todo el contenido del div centrado), además el framework ya define estilos por defecto para una gran cantidad de elementos HTML de la página.

Dicho ejemplo lo colocaremos, en la carpeta `dist` para no tener enlaces muy largos, pero podría estar en cualquier lugar.

```
<!DOCTYPE html>
<html>
<head>
  <title>Prueba de Bootstrap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link href="css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container" style="background-color:#ccc">
    <h1>Titulo</h1>
    <p>Esto es una prueba de bootstrap 4.</p>
  </div>
</body>
</html>
```

Como pueden observar, en el `<head>` tenemos el título de nuestro proyecto y se mostrará en la pestaña de la página y la codificación que se va a utilizar.

```
  <title>Prueba de Bootstrap</title>
  <meta charset="utf-8">
```

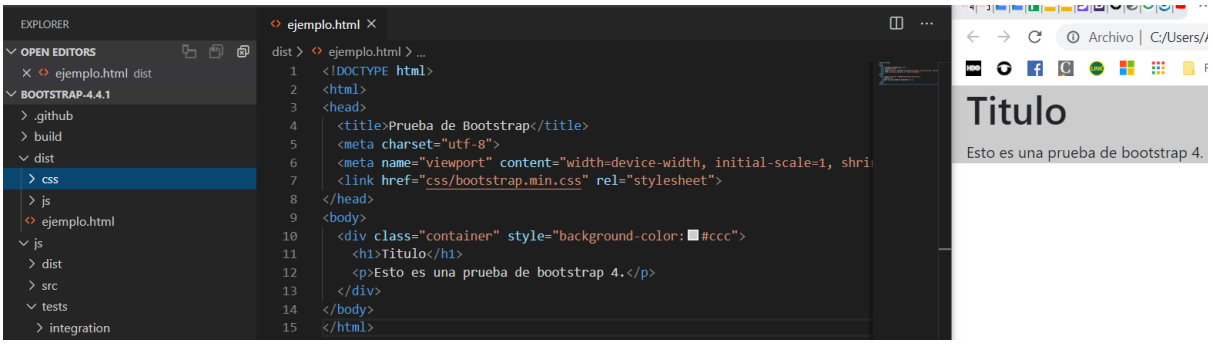
Luego encontramos esta línea:

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

El uso de la etiqueta viewport "width=device-width" (ancho=ancho del dispositivo) hace que la página se reduzca para ajustarse al contenido que desborda los límites del viewport.
Y especifica que la pantalla ocupará todo el espacio disponible con un tamaño de 1 (`initial-scale=1`).
Y como final del header tenemos la conexión con bootstrap

```
<link href="css/bootstrap.min.css" rel="stylesheet">
```

En la siguiente imagen podemos ver donde se encuentra el archivo `ejemplo.html`, al lado el código de ejemplo y al final tenemos el resultado de ejecutar nuestro ejemplo en el navegador Web.



CDN - Bootstrap 4.X.X

Hay una segunda forma de trabajar con Bootstrap sin tener que descargar el framework y consiste en utilizar un servidor donde se alojan todos los archivos de Bootstrap, y es en CDN.

CDN es el acrónimo de "Content delivery network"; es una red de servidores que ofrece librerías. Esto hace que no tenga sentido tener librerías en el servidor propio, solo es necesario "apuntar" hacia ellas.

Ventajas

- Reduce la carga de nuestros servidores sobretodo si utilizamos un hosting compartido, ayuda a evitar la saturación de la página web.
- Tiene la capacidad de absorber los picos de tráfico.
- Facilita que Bootstap 4 quede en caché del navegador y se vuelva a cargar cada vez que se visita una web que lo utilice.
- Reduce la latencia. Menor tiempo de carga.
- Reducción de costos de nuestros servidores por requerir menor ancho de banda y tráfico.
- Disminuye el tiempo de transferencia de datos.
- Atenúa la carga del servidor. Si tienes un pico de tráfico es la CDN quien se encarga, por lo que el servidor funcionará mejor.
- Es capaz de bloquear ataques. Por ejemplo, DDos o spam.

Desventajas

- Está la posibilidad que el servidor donde se aloja Bootstrap 4 se caiga, si bien es mucho más probable que nuestro servidor (sobre todo si es un hosting compartido) es el que tenga problemas.
- No podemos probar en forma local nuestro sitio web si no se encuentra conectado a internet.

Si vamos a utilizar un CDN (Content Delivery Network - red de entrega de contenidos), en vez de utilizar el enlace local de Bootstrap:

```
<link href="css/bootstrap.min.css" rel="stylesheet">
```

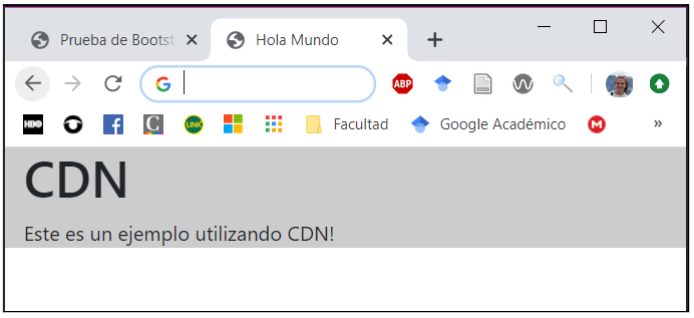
Utilizaremos el siguiente:

```
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

Nuestro esqueleto de página html tendrá una estructura similar a esta:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <title>Hola Mundo</title>
  </head>
  <body>
     <div class="container" style="background-color:#ccc">
       <h1>CDN</h1>
       <p>Este es un ejemplo utilizando CDN!</p>
     </div>
  </body>
</html>
```

Visualmente se verá así:



Grid, ¡olvídate de las tablas!

Bootstrap tiene un sistema de grillas que es importante que se entienda.

Para armar nuestro esquema de la página debemos pensar que tenemos la posibilidad de definir filas y en cada fila definir de 1 hasta 12 columnas. Cada columna con un ancho relativo a ese número 12.

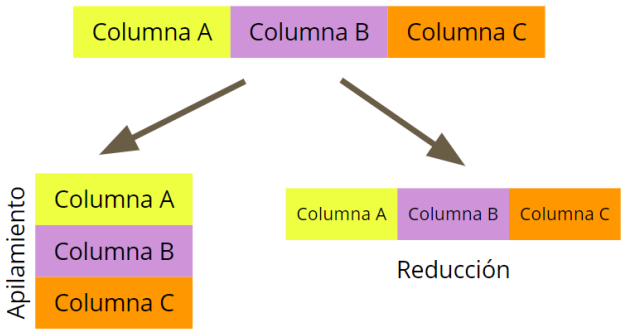
Luego Bootstrap se encarga de colapsar las columnas cuando se accede al sitio desde un dispositivo con una capacidad limitada en cuanto al ancho en píxeles.

Entonces, lo primero que tenemos que decidir es qué tipo de ancho queremos para nuestra web. En Bootstrap se diseña por filas, por lo que podemos tener un menú que ocupe todo el ancho y el cuerpo de la web sea fijo. Es decir, podremos combinar los dos sin ningún tipo problema.

En Bootstrap existen varias clases para ello: xs (para teléfonos móviles), sm (para tablets), md (para computadoras), lg (para computadoras con pantalla más grande).

Pero, ¿por qué 4 clases para las columnas?
Recordemos que Bootstrap es `responsive`, lo que quiere decir que se adapta al tamaño de la pantalla. Permite una visualización tanto en una pantalla grande como en un teléfono móvil.

Pero, ¿qué sucede con los elementos de una página web cuando la ventana se reduce o se expande? Existen dos hipótesis: los elementos se redimensionan y permanecen en su posición o se apilan cuando la ventana se vuelve más estrecha y se sitúan unos junto a otros cuando se expande.



Bootstrap considera 4 tipos de soportes: los pequeños, de tipo smartphone (menos de 768 píxeles), los medianos, de tipo tablet (menos de 992 píxeles), las pantallas medianas (menos de 1200 píxeles) y, finalmente, las pantallas grandes (más de 1200 píxeles). En la tabla que ilustra a continuación se ven las diferencias de reacción en función de la categoría.

	Pantalla pequeña (smartphone)	**Pantalla reducida (tablet)**	**Pantalla mediana (escritorio)**	**Pantalla grande (escritorio)**
Comporta-miento	Redimensionamiento	Redimensionamiento	Redimensionamiento	Apilamiento y redimensionamiento
Clase	col-sm-*	col-md-*	col-lg-*	col-xl-*
Valor de referencia	< 540 px	>= 720 px	>= 992 px	>= 1200 px

Clase .container

En Bootstrap, existían dos clases para contenedores: `.container` y `.container-fluid`. La diferencia entre ellas estriba en que la primera establece un ancho máximo concreto para el contenedor según el tamaño de la página (por ejemplo, para tamaños pequeños sm es de 540px o para pantallas extra grandes `lg` es de 1140px) mientras que el contenedor fluido siempre establece el ancho máximo en el 100%, de modo que se adapta a la pantalla, pero puede crecer mucho en algunos casos.

Bien, en Bootstrap 4.4 añaden unas nuevas clases de contenedor a las que le podemos añadir el nombre de un ancho (`sm`, `md`, `lg`, `xl`), de modo que se comportan como un híbrido de las dos anteriores: muestran un ancho máximo del 100% hasta alcanzar la medida indicada, y luego mantienen un ancho máximo fijo.

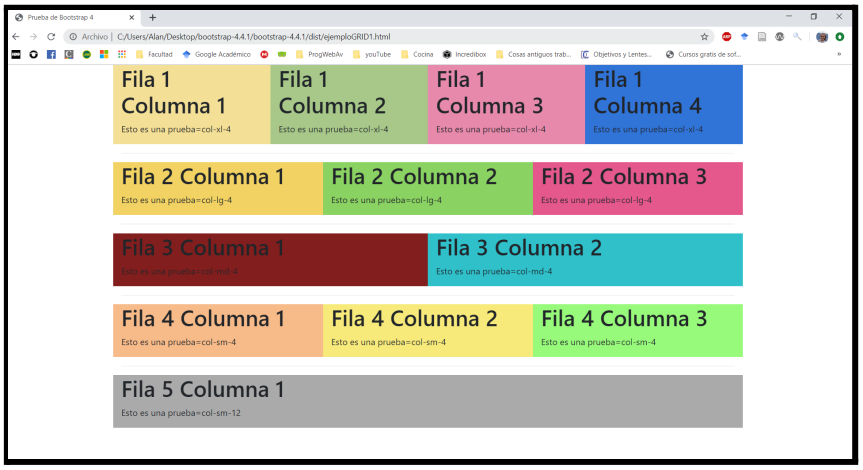
Por ejemplo, `.container-sm` tendrá un ancho máximo del 100% para pantallas pequeñas (hasta 576px) y un ancho fijo de 540px para las que lo superen. Y en el otro extremo `.container-xl` tendrá un ancho máximo del 100% hasta el tamaño extralargo (hasta 1200px) pero limitará el ancho a 1140px a partir de esas medidas del viewport.

Esto nos evita tener que estar limitando el ancho a mano en nuestras hojas de estilo si solo usamos el contenedor fluido. Obtenemos un comportamiento híbrido de los anteriores que nos permite tener contenedores fluidos limitando al mismo tiempo su ancho máximo.

Vamos a ir mostrando ejemplos en los cuales veremos las diferencias entre los diferentes tipos de columnas que podemos tener usando solo la clase `.container` . Y cómo se comportan de acuerdo al formato que le demos y a los tamaños de pantalla que tengamos.

El resto de las clases quedan a investigación de los estudiantes.

Para ver y entender cómo creamos las columnas en cada fila y cómo colapsan según el ancho del dispositivo vamos a ir viendo poco a poco el código de la siguiente página.



Ejemplo

La primera parte de nuestro código html ya lo hemos visto con anterioridad, así que solo lo mostramos.

```
<!doctype html>
<html>
<head>
  <title>Prueba de Bootstrap 4</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
```

Luego en el `body` comenzamos colocando la clase `container`:

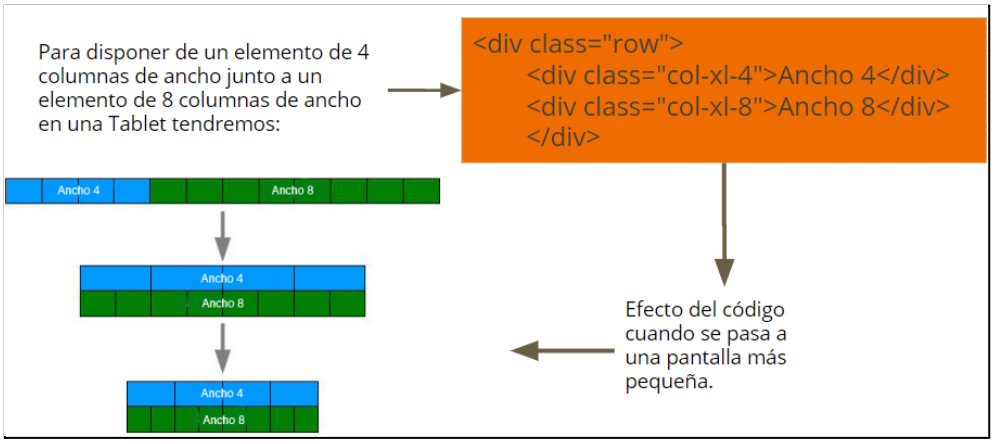
```
<body>
  <div class="container">
```

Luego, como ya explicamos, Bootstrap trabaja con un sistema de grillas, y la manera de trabajarlo es con filas y columnas.

Debemos utilizar la clase `row` para indicar el comienzo de una fila. Dentro de la fila dispondremos tantos `div` como columnas tenga dicha fila.

col-xl-*

Las columnas que se programan con `col-xl`, son para las pantallas de computadoras que tienen <=1200px de pantalla. Al utilizar este formato, cuando se achique un poco la pantalla del navegador, enseguida colapsará y las columnas pasan a apilarse.



Comenzamos a trabajar con el ejemplo. Para indicar cada columna, debemos utilizar la siguiente sintaxis:

```
<div class="col-xl-*"  ....>
```

Y donde aparece el asterisco lo reemplazamos por un valor entre 1 y 12. Teniendo en cuenta que la suma de todas las columnas que tengamos, tiene que ser 12, sino quedarán columnas vacías.

Si en el asterisco colocamos un 3, nos quedan 9 unidades de columnas para repartir entre las otras columnas.

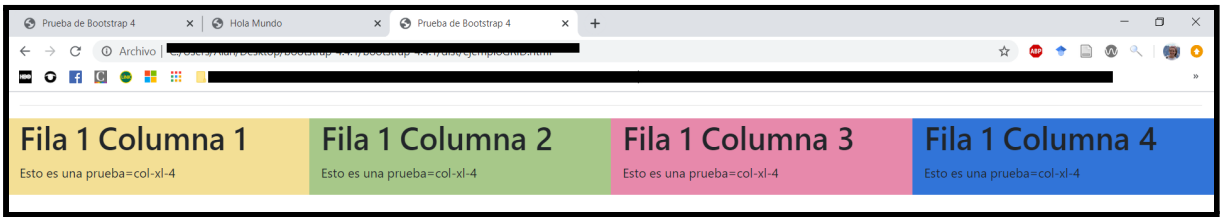
Entonces, para la primer fila de nuestro ejemplo asignamos un tamaño de 3 a cada columna, por lo que tendremos 4 columnas del mismo tamaño.

```
<body>
<hr>
  <div class="container">
    <div class="row">
      <div class="col-xl-3"  style="background-color:rgb(243, 223, 149)">
        <h1>Fila 1 Columna 1</h1>
        <p>Esto es una prueba=col-xl-3</p>
      </div>
      <div class="col-xl-3"  style="background-color:rgb(167, 200, 137)">
        <h1>Fila 1 Columna 2</h1>
        <p>Esto es una prueba=col-xl-3</p>
      </div>
      <div class="col-xl-3"  style="background-color:rgb(231, 137, 171)">
        <h1>Fila 1 Columna 3</h1>
        <p>Esto es una prueba=col-xl-3</p>
      </div>
      <div class="col-xl-3"  style="background-color:rgb(49, 116, 216)">
        <h1>Fila 1 Columna 4</h1>
        <p>Esto es una prueba=col-xl-3</p>
```

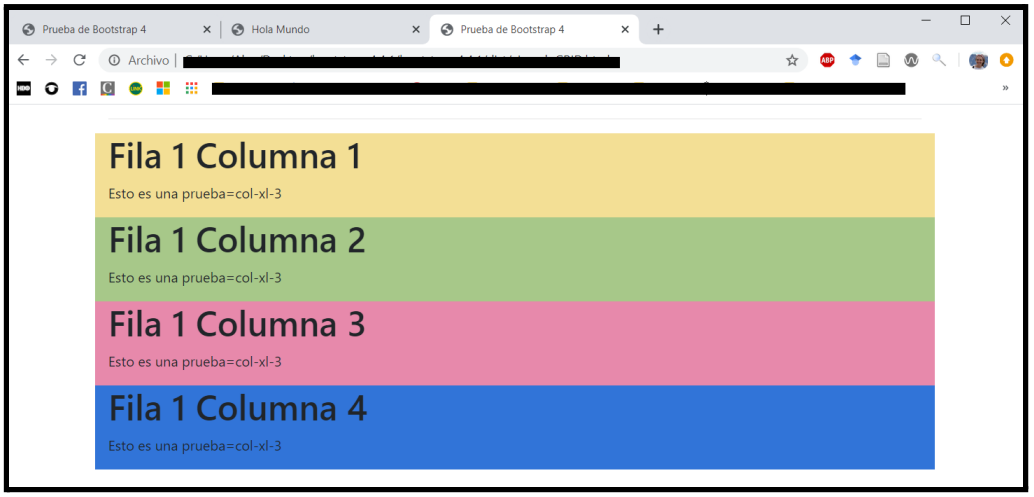
```
    </div>
  </div>
```

En el navegador veremos algo así:

Note que al utilizar la clase `container` nuestras columnas se colocan centradas en la pantalla porque el navegador está en pantalla completa. Si en vez de utilizar la clase `container` utilizamos `container-fluid` se verían las columnas expandidas, como se muestra en la siguiente imagen.



Continuando con el ejemplo anterior (utilizando la clase `container`) medida que achiquemos la ventana del navegador veremos que, cuando el ancho del navegador es menor a 1200 px, colapsan las columnas de la primer fila y se muestran una debajo de otra (esto sucede en cualquier dispositivo cuyo ancho en píxeles es menor a 1200).



col-lg-*

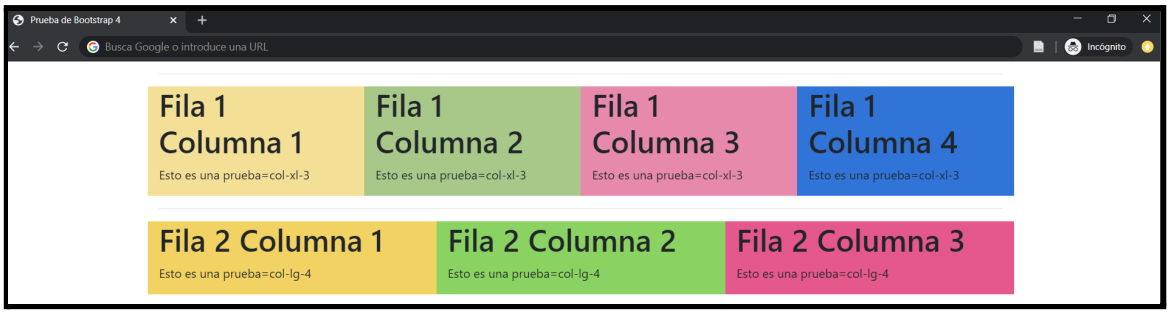
Para seguir con el ejemplo, ahora colocaremos una nueva fila, con 3 columnas y que dichas columnas colapsen cuando el tamaño del navegador sea menor a 992px.
Para ello trabajaremos con la clase:

```
<div class="col-lg-*"  ...">
```

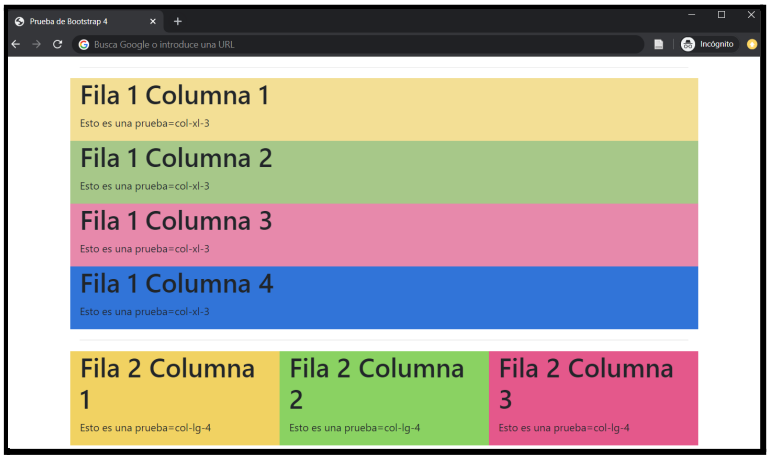
En este caso, como queremos 3 columnas en nuestra fila, en el asterisco colocaremos el número 4. El código que agregamos es el siguiente:

```
<hr>
    <div class="row">
      <div class="col-lg-4"  style="background-color:rgb(241, 210, 98)">
        <h1>Fila 2 Columna 1</h1>
        <p>Esto es una prueba=col-lg-4</p>
      </div>
      <div class="col-lg-4"  style="background-color:rgb(138, 210, 98)">
        <h1>Fila 2 Columna 2</h1>
        <p>Esto es una prueba=col-lg-4</p>
      </div>
      <div class="col-lg-4"  style="background-color:rgb(228, 88, 139)">
        <h1>Fila 2 Columna 3</h1>
        <p>Esto es una prueba=col-lg-4</p>
      </div>
    </div>
```

Cuando se muestra en pantalla completa vemos lo siguiente:



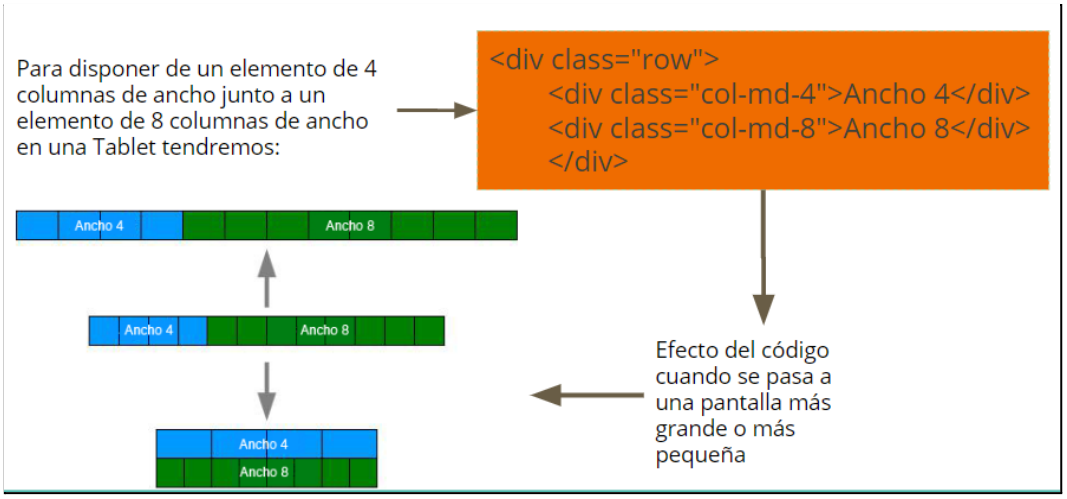
Y cuando se comienza a achicar la pantalla podemos notar que la primer fila colapsa y se apila, mientras que la segunda solo se redimensiona.



Cuando utilizamos las clases `col-lg-*` la fila colapsará cuando el ancho del navegador se reduzca a menos de 992px, es decir, reduciendo aún más la pantalla. Y cuando la pantalla llegue a menos de 992px, ambas filas colapsan.

col-md-*

Cuando se utiliza col-md (medium), que equivale a una pantalla de una tablet y se quiere visualizar en una pantalla más grande o más chica tenemos lo siguiente:



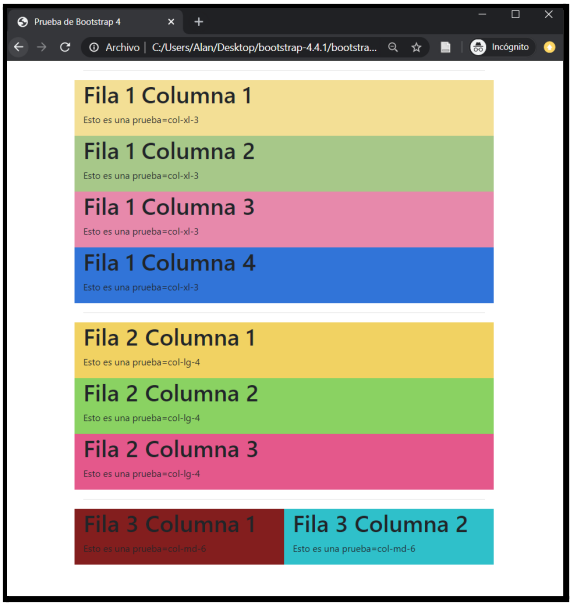
Para seguir con el ejemplo, ahora colocaremos una nueva fila, con 2 columnas y que dichas columnas colapsen cuando el tamaño del navegador sea menor a 720px.
Para ello trabajaremos con la clase:

```
<div class="col-md-*"  ...">
```

Cómo serán solo dos columnas, en el asterisco colocamos el valor 6. Para que cada columna ocupe 6 unidades de columna.

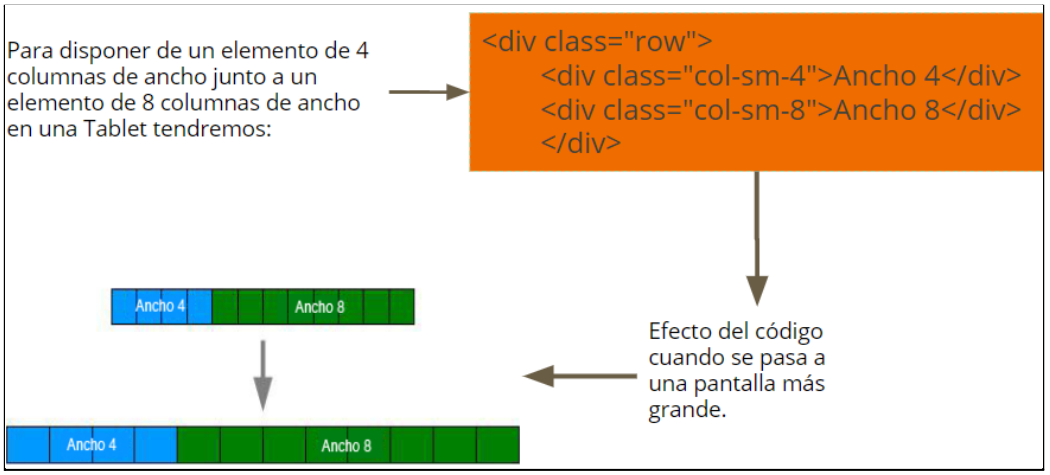
```
<div class="row">
      <div class="col-md-6"  style="background-color:rgb(131, 30, 30)">
        <h1>Fila 3 Columna 1</h1>
        <p>Esto es una prueba=col-md-6</p>
      </div>
      <div class="col-md-6"  style="background-color:rgb(47, 192, 202)">
        <h1>Fila 3 Columna 2</h1>
        <p>Esto es una prueba=col-md-6</p>
      </div>
    </div>
```

Mientras la pantalla no llegue a una resolución menor a 720px, las columnas no se apilaran, pero nótese, que tanto la fila 1, como la fila 2 ya colapsaron y se apilaron porque el tamaño del navegador es menor que 992px.



col-sm-*

Cuando se utiliza col-sm, es decir el tamaño small, que equivale a una pantalla de un smartphone y se quiere visualizar en una pantalla más grande tenemos lo siguiente:



Seguimos con nuestro ejemplo. En la cuarta fila utilizamos:

```
<div class="col-sm-*"  ...">
```

En este caso también utilizamos columnas de tamaño 4, para que puedan notar que no importa el tamaño que tengan dichas columnas, sino de la clase que se utilice para que estas colapsen y se apilen.

```
<div class="row">
        <div class="col-sm-4"  style="background-color:rgb(247, 186, 137)">
          <h1>Fila 4 Columna 1</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
        <div class="col-sm-4"  style="background-color:rgb(247, 234, 122)">
          <h1>Fila 4 Columna 2</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
        <div class="col-sm-4"  style="background-color:rgb(152, 250, 123)">
          <h1>Fila 4 Columna 3</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
      </div>
```

Visualmente se verán así:

Dichas columnas colapsan al llegar a un valor menor de 540px.

Por ello es que en la pantalla podemos ver tanto la fila 1 como la fila 2 y 3 que están colapsadas y apiladas ya que el tamaño del navegador es menor que 720px, pero todavía no llega a los 540px para colapsar además la fila 4.

col-*

Por último cuando utilizamos la clase col-*

```
<div class="col-*"  ...">
```

Esta clase, la particularidad que tiene es que las columnas nunca colapsarán. Se reducirán pero nunca se apilaran.

```
<div class="row">
       <div class="col-4"  style="background-color:rgb(247, 186, 137)">
         <h1>Fila 5 Columna 1</h1>
```

```
      <p>Esto es una prueba=col-4</p>
    </div>
    <div class="col-4"  style="background-color:rgb(247, 234, 122)">
      <h1>Fila 5 Columna 2</h1>
      <p>Esto es una prueba=col-4</p>
    </div>
    <div class="col-4"  style="background-color:rgb(152, 250, 123)">
      <h1>Fila 5 Columna 3</h1>
      <p>Esto es una prueba=col-4</p>
    </div>
  </div>
```

Noten, que se empiezan a superponer las letras de las columnas, y genera que se cree una barra de desplazamiento, ya que las columnas no entran en el tamaño final que utilizamos (<=540px).

Ahora depende del tipo de página que uno tiene que implementar el definir en qué momento queremos que nuestra estructura de página colapse sus columnas. Si no queremos que colapse luego empleamos col-* para definir las distintas columnas.

Código final

El código final de este ejemplo es:

```
<!doctype html>
<html>
<head>
  <title>Prueba de Bootstrap 4</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <hr>
        <div class="row">
          <div class="col-xl-3"  style="background-color:rgb(243, 223, 149)">
            <h1>Fila 1 Columna 1</h1>
            <p>Esto es una prueba=col-xl-3</p>
          </div>
          <div class="col-xl-3"  style="background-color:rgb(167, 200, 137)">
            <h1>Fila 1 Columna 2</h1>
            <p>Esto es una prueba=col-xl-3</p>
          </div>
          <div class="col-xl-3"  style="background-color:rgb(231, 137, 171)">
            <h1>Fila 1 Columna 3</h1>
            <p>Esto es una prueba=col-xl-3</p>
          </div>
          <div class="col-xl-3"  style="background-color:rgb(49, 116, 216)">
            <h1>Fila 1 Columna 4</h1>
            <p>Esto es una prueba=col-xl-3</p>
          </div>
        </div>
    <hr>
    <div class="row">
        <div class="col-lg-4"  style="background-color:rgb(241, 210, 98)">
          <h1>Fila 2 Columna 1</h1>
          <p>Esto es una prueba=col-lg-4</p>
        </div>
        <div class="col-lg-4"  style="background-color:rgb(138, 210, 98)">
          <h1>Fila 2 Columna 2</h1>
          <p>Esto es una prueba=col-lg-4</p>
        </div>
        <div class="col-lg-4"  style="background-color:rgb(228, 88, 139)">
          <h1>Fila 2 Columna 3</h1>
          <p>Esto es una prueba=col-lg-4</p>
        </div>
      </div>
      <hr>
```

```
    <div class="row">
      <div class="col-md-6"  style="background-color:rgb(131, 30, 30)">
        <h1>Fila 3 Columna 1</h1>
        <p>Esto es una prueba=col-md-6</p>
      </div>
      <div class="col-md-6"  style="background-color:rgb(47, 192, 202)">
        <h1>Fila 3 Columna 2</h1>
        <p>Esto es una prueba=col-md-6</p>
      </div>
    </div>
    <hr>
    <div class="row">
        <div class="col-sm-4"  style="background-color:rgb(247, 186, 137)">
          <h1>Fila 4 Columna 1</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
        <div class="col-sm-4"  style="background-color:rgb(247, 234, 122)">
          <h1>Fila 4 Columna 2</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
        <div class="col-sm-4"  style="background-color:rgb(152, 250, 123)">
          <h1>Fila 4 Columna 3</h1>
          <p>Esto es una prueba=col-sm-4</p>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-4"  style="background-color:rgb(247, 186, 137)">
          <h1>Fila 5 Columna 1</h1>
          <p>Esto es una prueba=col-4</p>
        </div>
        <div class="col-4"  style="background-color:rgb(247, 234, 122)">
          <h1>Fila 5 Columna 2</h1>
          <p>Esto es una prueba=col-4</p>
        </div>
        <div class="col-4"  style="background-color:rgb(152, 250, 123)">
          <h1>Fila 5 Columna 3</h1>
          <p>Esto es una prueba=col-4</p>
        </div>
      </div>
</body>
</html>
```

Alineación de filas y columnas

Bootstrap 3

Con las versiones de Bootstrap anteriores a la 4.4 las columnas se podían alinear a través de css. Por

ejemplo, cuando trabajamos saltandonos columnas, lo hacíamos de la siguiente manera.

Para esta imagen:

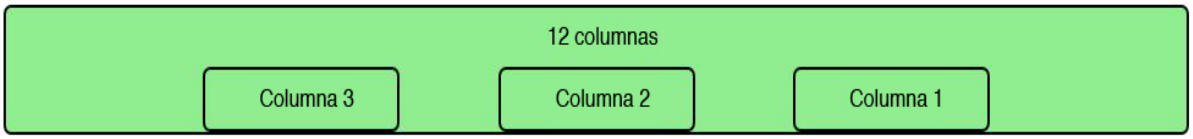
Teníamos el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-lg-3">3 columnas</div>
            <div class="col-lg-6">6 columnas</div>
            <div class="col-lg-3">3 columnas</div>
        </div>
        <div class="row">
            <div class="col-lg-3">3 columnas</div>
            <div class="col-lg-offset-6 col-lg-3">3 columnas</div>
        </div>
    </div>
</body>
</html>
```

Enlazado a un CSS assets/css/tuto.cs:
Estas clases simplemente añaden un margen a la izquierda. En este caso, 6 columnas representan la mitad de la ventana, es decir, 50%.

```
.col-lg-offset-6 {
        margin-left: 50%;
        }
```

O por ejemplo, para poder tener alineadas la columnas contenidas dentro de otra columna lo hacíamos con pull y push.
La clase `col-lg-push-*` permite separar una columna hacia la derecha y la `clase col-lg-pull-*` hace lo contrario.
Es decir, para la siguiente imagen:



Teníamos que colocar, para una alineación a izquierda:

```
<div class="col-lg-2 col-lg-pull-2">Columna 3</div>
```

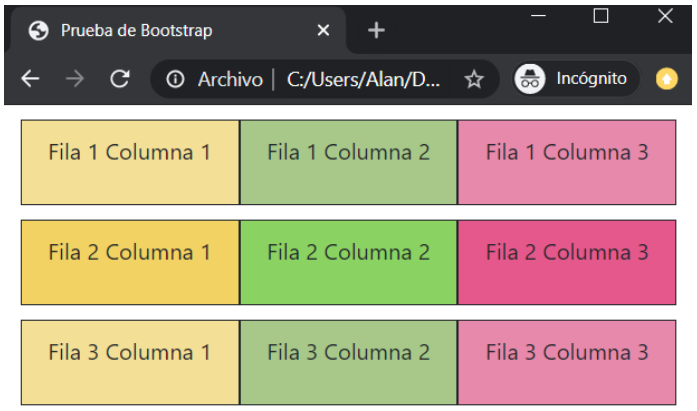
O para una alineación a derecha:

```
<div class="col-lg-2 col-lg-push-8">Columna 1</div>
```

Bootstrap 4

En Bootstrap 4 esto cambió bastante la forma en la que se tratan las tablas.
Es posible elegir la alineación vertical de cada fila. Esto es, añadiendo a la clase `.row` alguna de las clases `.align-items-start`, `.align-items-center` o `.align-items-end`.

Por ejemplo, para la siguiente página:



Utilizamos los 3 alineamientos que mencionamos.
Por un lado, para tener una alineación en el inicio de la página utilizamos:

```
<div class="row align-items-start">
```

Para tener una alineación en el centro de la página utilizamos:

```
<div class="row align-items-center">
```

Y para tener una alineación al final de la página utilizamos:

```
<div class="row align-items-end">
```

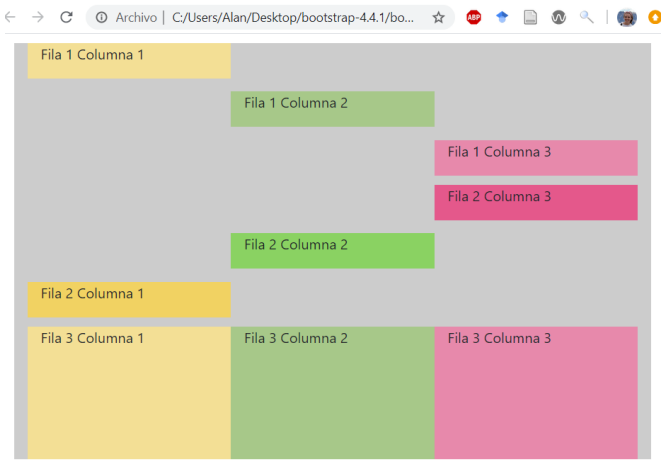
El código del ejemplo es el siguiente:

```
<body>
    <div class="container">
        <div class="row align-items-start">
            <div class="col-4"  style="background-color:rgb(243, 223, 149)">
                <p>Fila 1 Columna 1</p>
            </div>
            <div class="col-4" style="background-color:rgb(167, 200, 137)">
                <p>Fila 1 Columna 2</p>
            </div>
            <div class="col-4"  style="background-color:rgb(231, 137, 171)">
                <p>Fila 1 Columna 3</p>
            </div>
        </div>
        <div class="row align-items-center">
            <div class="col-4"  style="background-color:rgb(241, 210, 98)">
                <p>Fila 2 Columna 1</p>
            </div>
            <div class="col-4"  style="background-color:rgb(138, 210, 98)">
                <p>Fila 2 Columna 2</p>
            </div>
            <div class="col-4"  style="background-color:rgb(228, 88, 139)">
                <p>Fila 2 Columna 3</p>
            </div>
        </div>
        <div class="row align-items-end">
            <div class="col-4"  style="background-color:rgb(243, 223, 149)">
                <p>Fila 3 Columna 1</p>
            </div>
            <div class="col-4" style="background-color:rgb(167, 200, 137)">
                <p>Fila 3 Columna 2</p>
            </div>
            <div class="col-4"  style="background-color:rgb(231, 137, 171)">
                <p>Fila 3 Columna 3</p>
            </div>
        </div>
    </div>
</body>
```

De igual forma, cada columna puede adaptar su alineación vertical mediante las siguientes clases:

```
<div class="col-*  align-self-start" ...>
<div class="col-*  align-self-center" ...>
<div class="col-*  align-self-end" ...>
```

Por ejemplo, si modificamos el ejemplo anterior utilizando estas clases, obtenemos lo siguiente:



Donde la primer y segunda fila tienen aplicada estas clases y la tercer fila no. Para que pueda notarse la diferencia entre cada una.
El código es el siguiente:

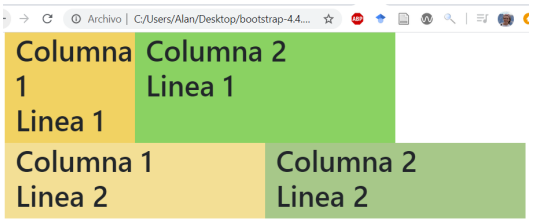
```
<!DOCTYPE html>
<html>

<head>
    <title>Prueba de Bootstrap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <style type="text/css">
        .container { background: #ccc; }
        .row { margin: 10px 0; height: 150px; }
    </style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-4  align-self-start"  style="background-color:rgb(243, 223, 149)">
                <p>Fila 1 Columna 1</p>
            </div>
            <div class="col-4  align-self-center" style="background-color:rgb(167, 200, 137)">
                <p>Fila 1 Columna 2</p>
            </div>
            <div class="col-4  align-self-end"  style="background-color:rgb(231, 137, 171)">
                <p>Fila 1 Columna 3</p>
            </div>
        </div>
        <div class="row">
            <div class="col-4  align-self-end"  style="background-color:rgb(241, 210, 98)">
                <p>Fila 2 Columna 1</p>
            </div>
            <div class="col-4  align-self-center"  style="background-color:rgb(138, 210, 98)">
                <p>Fila 2 Columna 2</p>
            </div>
            <div class="col-4   align-self-start"  style="background-color:rgb(228, 88, 139)">
                <p>Fila 2 Columna 3</p>
```

```
            </div>
        </div>
        <div class="row">
            <div class="col-4"  style="background-color:rgb(243, 223, 149)">
                <p>Fila 3 Columna 1</p>
            </div>
            <div class="col-4" style="background-color:rgb(167, 200, 137)">
                <p>Fila 3 Columna 2</p>
            </div>
            <div class="col-4"  style="background-color:rgb(231, 137, 171)">
                <p>Fila 3 Columna 3</p>
            </div>
        </div>
    </div>
</body>
</html>
```

Varias líneas en una misma fila "row"

Con Bootstrap 4 se ha agregado la posibilidad de que en una fila definida por la clase "row" especifique varias columnas y mediante la clase "w-100" permitir hacer un salto de línea.

Con un ejemplo debe quedar claro cómo podemos definir varias líneas en una misma fila:



Podemos comprobar que luego que se aplica la clase "w-100" a un `div` vacío el siguiente `div` aparece en una nueva línea.

Offset

En Bootstrap 3 se utilizan los offset para saltarnos algunas columnas, como vimos en ejemplos anteriores, ahora en Bootstrap 4.4, si bien existe el `offset`, son los márgenes los encargados de gestionar el espacio expreso entre cada columna a través de las clases.
Disponemos las siguientes clases para implementar el desplazamiento (offset):

```
offset-*
offset-sm-*
offset-md-*
offset-lg-*
offset-xl-*
```

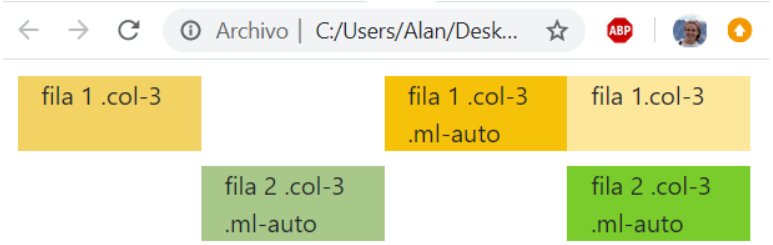
Recordar que el asterisco representa un valor entre 1 y 12.

Si bien se puede seguir usando offset, las clases que nombramos a continuación son más prácticas dependiendo del modelado que queramos realizar. En algunos casos completos el concepto de offset nos puede ayudar más que el desplazamiento horizontal.
Las nuevas clases son:

```
<div class="col-3" ml-auto
```

y
.mr-auto

combinables igualmente con interfijos de tamaños que dan lugar a clases como .ml-md-auto:



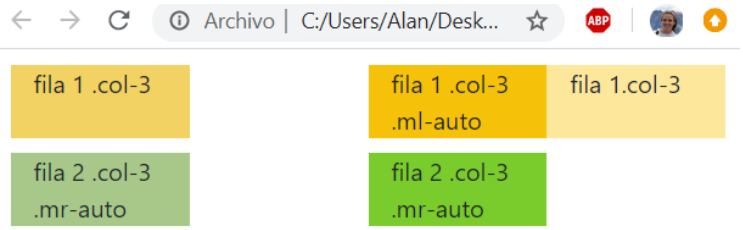

```
</head>

<body>
        <div class="container">
                <div class="row">
                        <div class="col-3" style="background-color:rgb(241, 210, 98)">
                                fila 1 .col-3
                        </div>
                        <div class="col-3 ml-auto" style="background-color:rgb(245, 194, 9)">
                                fila 1 .col-3 .ml-auto
                        </div>
                        <div class="col-3" style="background-color:rgb(252, 231, 155)">
                                fila 1.col-3
                        </div>
                </div>
                <div class="row">
                        <div class="col-3 ml-auto" style="background-color:rgb(167, 200, 137)">
                                fila 2 .col-3 .ml-auto
                        </div>
                        <div class="col-3 ml-auto" style="background-color:rgb(121, 204, 44)">
                                fila 2 .col-3 .ml-auto
                        </div>
                </div>
        </div>
</body>
</html>
```

Y cambiando algunos `.ml-auto` por `.mr-auto` para la segunda fila por ejemplo:

```
<div class="row">
        <div class="col-3 mr-auto" style="background-color:rgb(167, 200, 137)">
                fila 2 .col-3 .mr-auto
        </div>
```

```
        <div class="col-3 m-auto" style="background-color:rgb(121, 204, 44)">
                fila 2 .col-3 .ml-auto
        </div>
</div>
```

Obtenemos la siguiente vista.



Anidamiento de columnas

Otro tema perfectamente resuelto por Bootstrap 4 es el anidamiento de columnas.
Por ejemplo, para implementar una página que muestre dos columnas y dentro de la segunda columna otras dos columnas internas.

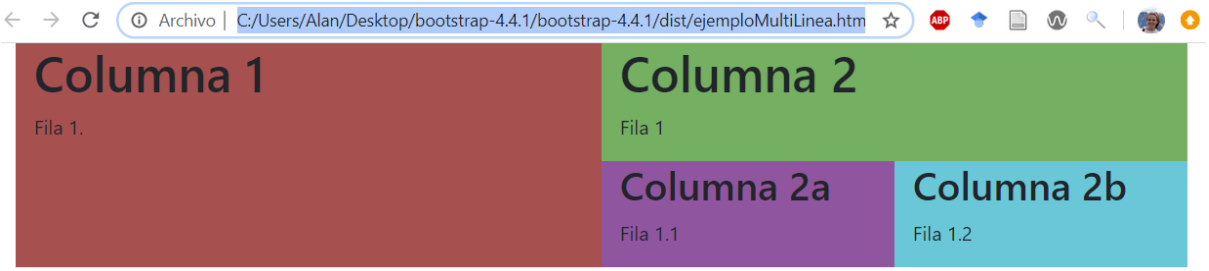
```
<body>
    <div class="container">
        <div class="row">
            <div class="col-lg-6" style="background-color:rgb(167, 80, 80)">
                <h1>Columna 1</h1>
                <p>Fila 1.</p>
            </div>
            <div class="col-lg-6" style="background-color:rgb(116, 175, 98)">
                <h1>Columna 2</h1>
                <p>Fila 1</p>
                <!-- acá se crea otra clase row, que pertenece la la clase row de más arriba -->
                <div class="row">
                    <div class="col-lg-6" style="background-color:rgb(143, 85, 158)">
                        <h2>Columna 2a</h2>
                        <p>Fila 1.1</p>
                    </div>
                    <div class="col-lg-6" style="background-color:rgb(106, 199, 216)">
                        <h2>Columna 2b</h2>
                        <p>Fila 1.2</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

Lo importante es analizar la segunda columna donde vemos que insertamos además de un título y un párrafo tenemos una nueva fila (row) y dos nuevos div que debemos repartir las doce columnas:

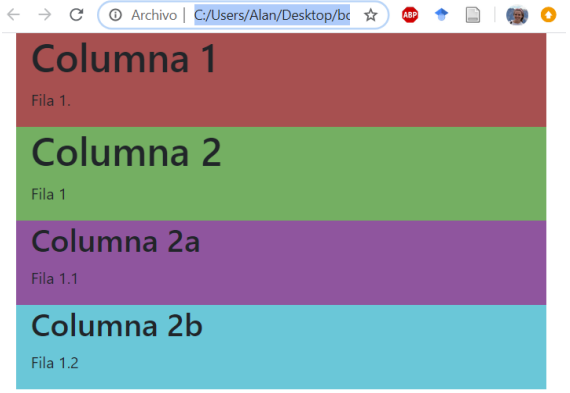
```
...
<div class="col-lg-6" style="background-color:rgb(116, 175, 98)">
    <h1>Columna 2</h1>
```

```
    <p>Fila 1</p>

    <div class="row">
        <div class="col-lg-6" style="background-color:rgb(143, 85, 158)">
            <h2>Columna 2a</h2>
            <p>Fila 1.1</p>
        </div>
        <div class="col-lg-6" style="background-color:rgb(106, 199, 216)">
            <h2>Columna 2b</h2>
            <p>Fila 1.2</p>
        </div>
    </div>

</div>
```

Podemos observar que la segunda columna contiene a su vez otras dos columnas:



También podemos ver que si reducimos el ancho del navegador colapsan todas las columnas (externas e internas):



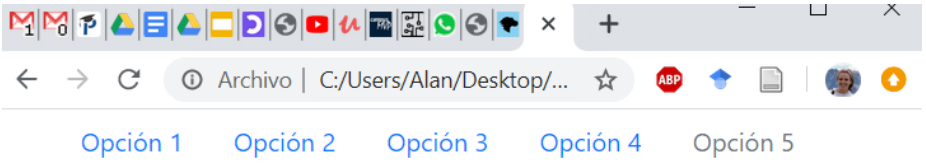
Y así se puede ir jugando con las diferentes columnas y filas para poder armar una vista.
Por otro lado, se puede crear componentes de navegación con Bootstrap. La manera de crearlas en con la clase .nav.

.nav

Para poder crear elementos de navegación se utiliza la clase .nav y después aplica la clase .nav-tabs para mostrar sus enlaces en forma de pestaña.

Para crear un simple menú horizontal en Bootstrap 4 disponemos de la clase "nav", y nav-link.

Vemos un menú de navegación básico implementado con la clase "nav":



El menú de navegación dispone de 5 enlaces y el cuarto se muestra desactivo debido a la clase `disabled`.

```
<!doctype html>
<html>

<head>
    <title>Prueba de Bootstrap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
    <div class="container">
        <nav class="nav">
            <a class="nav-link" href="#">Opción 1</a>
            <a class="nav-link" href="#">Opción 2</a>
            <a class="nav-link" href="#">Opción 3</a>
            <a class="nav-link" href="#">Opción 4</a>
            <a class="nav-link disabled" href="#">Opción 5</a>
        </nav>
    </div>
</body>

</html>
```

Los `href = "#"` serán completados con los enlaces que queramos para nuestro menú de opciones.
Si queremos un menú vertical, en vez de pestañas, para mostrar las opciones una debajo de la otra debemos agregar la clase `flex-column` a la etiqueta HTML "nav":

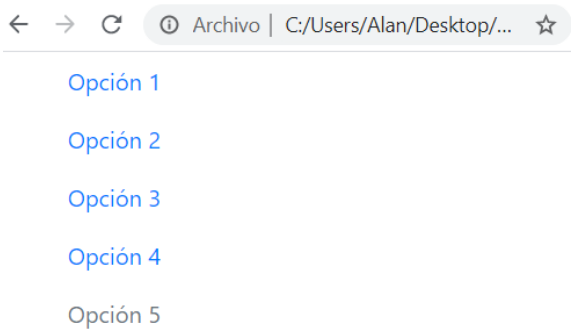
Es decir, en vez de utilizar solo la clase:

```
<nav class="nav">
```

Utilizamos:

```
<nav class="nav flex-column">
```

Y dichas opciones se muestran así:

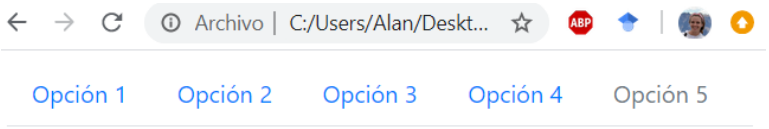


Podemos fácilmente transformar nuestro menú de opciones a pestañas agregando la clase `nav-tabs`, de forma similar si queremos que se muestran como pastillas debemos emplear en su lugar la clase `nav-pills`.

Utilizando:

```
<nav class="nav nav-tabs mt-2"> <!-- Agregamos la clase mt-2 para dar un margen superior y evitar que salga pegado. -->
```

El menú se se ve como sigue:



Si queremos que alguna de las opciones este activa, es decir que se muestre al inicio, le colocamos al hipervínculo la palabra `active`.

```
<a class="nav-link active" href="#">Opción 1</a>
```

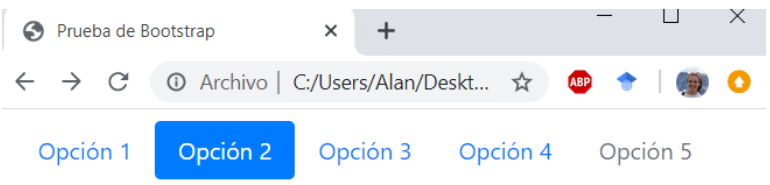
Y si cambiamos la clase

```
<nav class="nav nav-tabs ...>
```

por la clase:

```
<nav class="nav nav-pills ...>
```

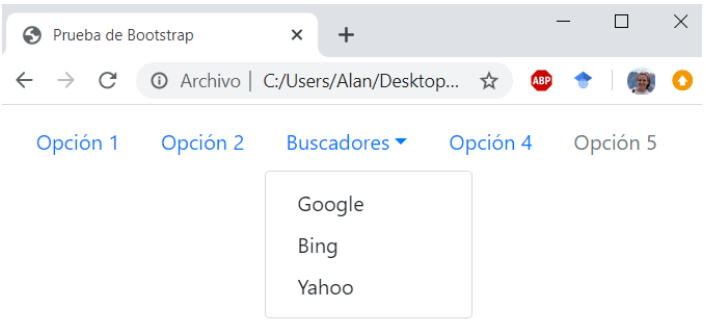
Visualmente se verá como muestra la siguiente imagen, note además que en este caso el botón activo es Opción 2 y el botón Opción 5 está deshabilitado:



La componente `nav` nos permite definir una serie de hipervínculos. Ahora veremos que en lugar de

alguno de esos hipervínculos podemos disponer una o más componentes de tipo `dropdown (listas desplegables)`.

Las componentes de tipo "dropdown" (listas desplegables) requieren los archivos de Javascript propuestos por Bootstrap 4.
Por ejemplo, para crear una lista desplegable y que se vea de la siguiente manera:



La sintaxis de un menú horizontal con una opción de tipo "dropdown":

```
<body>
    <div class="container">
        <ul class="nav mt-2">
            <li class="nav-item">
                <a class="nav-link" href="#">Opción 1</a>
            </li>
            <li class="nav-item">
                <a class="nav-link active" href="#">Opción 2</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#">Buscadores</a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" href="http://www.google.com">Google</a>
                    <a class="dropdown-item" href="http://www.bing.com">Bing</a>
                    <a class="dropdown-item" href="http://www.yahoo.com">Yahoo</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Opción 4</a>
            </li>
            <li class="nav-item">
                <a class="nav-link disabled" href="#">Opción 5</a>
            </li>
        </ul>
    </div>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
        integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>
</body>
```

Utilizamos la etiqueta HTML "ul":

```
<ul class="nav mt-2">
```

Cada opción es un "li" con la clase "nav-item":

```
<li class="nav-item">
```

Los enlaces agregan la clase "nav-link":

```
<a class="nav-link" href="#">Opción 1</a>
```

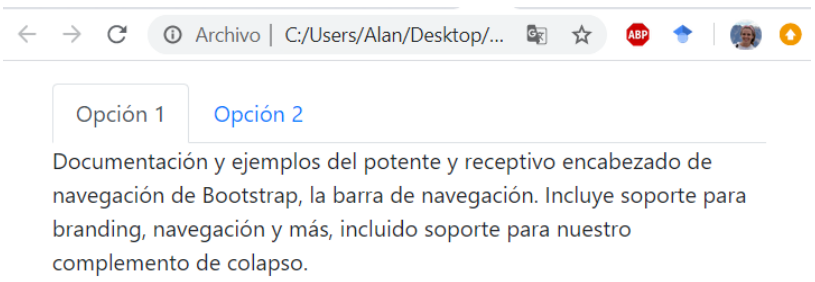
El menú desplegable (dropdown) también es un "li" que le definimos las clases "nav-item" y "dropdown":

```
<li class="nav-item dropdown">
       <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#">Buscadores</a>
       <div class="dropdown-menu">
         <a class="dropdown-item" href="http://www.google.com">Google</a>
         <a class="dropdown-item" href="http://www.bing.com">Bing</a>
         <a class="dropdown-item" href="http://www.yahoo.com">Yahoo</a>
       </div>
     </li>
```

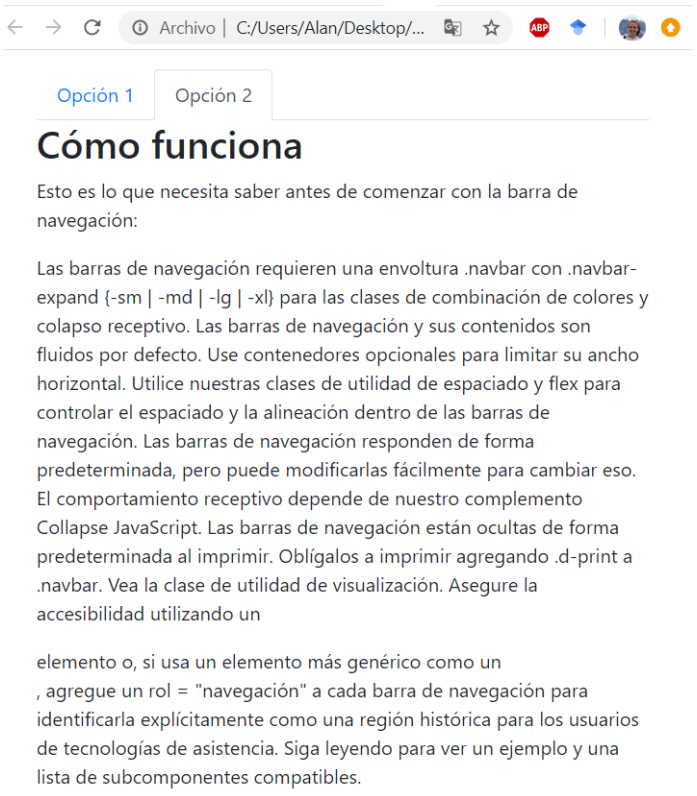
Una funcionalidad más compleja que podemos utilizar para los menús con pestañas o píldoras es asociarle un div con contenido.

Solo se muestra la información de la opción seleccionada. Mediante Javascript Bootstrap 4 resuelve el problema de hacer visible el div de la opción que se selecciona y ocultar la otra.

Veamos un ejemplo con 2 pestañas que muestran distintos contenidos al ser presionados:
La pestaña “Opción 1” contiene la siguiente información:



Mientras que la “Opción 2” contiene la información que se encuentra a continuación:



Definimos una lista no ordenada (ul) con las clases "nav" y "nav-tabs":

```
<ul class="nav nav-tabs mt-3">
```

Cada "li" debe agregar la clase "nav-item" y el ancla contenida debe definir la propiedad data-toggle con el valor "tab" y la propiedad href con un valor que debe coincidir con el div definido más abajo:

```
<li class="nav-item">
    <a class="nav-link active"  data-toggle="tab" href="#opcion1" >Opción 1</a>
</li>
```

De forma similar definimos la otra pestaña:

```
<li class="nav-item">
   <a class="nav-link"  data-toggle="tab" href="#opcion2">Opción 2</a>
</li>
```

Luego debemos declarar un div con la clase "tab-content" y dentro de este div tantos div como pestañas tenga nuestro menú definiendo la propiedad id con los valores iniciados en la propiedad href de las anclas, además a cada div debemos inicializarlos con la clase "tab-pane":

```
<div class="tab-content">
   <div class="tab-pane fade show active" id="opcion1">
      ...
   </div>
   <div class="tab-pane fade" id="opcion2" >
```

```
    ...
```

El código final entonces queda de la siguiente manera:

```
<!doctype html>
<html>

<head>
    <title>Prueba de Bootstrap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
    <div class="container">
        <ul class="nav nav-tabs mt-3">
            <li class="nav-item">
                <a class="nav-link active" data-toggle="tab" href="#opcion1">Opción 1</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" data-toggle="tab" href="#opcion2">Opción 2</a>
            </li>
        </ul>
        <div class="tab-content">
            <div class="tab-pane fade show active" id="opcion1">
                Documentation and examples for Bootstrap's powerful, responsive navigation header,
the navbar. Includes
                support for branding, navigation, and more, including support for our collapse
plugin.
            </div>
            <div class="tab-pane fade" id="opcion2">
                <h2>How it works</h2>
                <p>Here's what you need to know before getting started with the navbar:</p>
                <p>Navbars require a wrapping .navbar with .navbar-expand{-sm|-md|-lg|-xl} for
responsive collapsing and
                    color scheme classes.
                    Navbars and their contents are fluid by default. Use optional containers to
limit their horizontal
                    width.
                    Use our spacing and flex utility classes for controlling spacing and alignment
within navbars.
                    Navbars are responsive by default, but you can easily modify them to change
that. Responsive
                    behavior depends on our Collapse JavaScript plugin.
                    Navbars are hidden by default when printing. Force them to be printed by adding
.d-print to the
                    .navbar. See the display utility class.
                    Ensure accessibility by using a <nav> element or, if using a more generic
element such as a <div>,
                            add a role="navigation" to every navbar to explicitly identify it as a
landmark region for
                            users of assistive technologies.
                            Read on for an example and list of supported sub-components.
                </p>
            </div>
        </div>
    </div>
```

```
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
        integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
        integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
        crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
        crossorigin="anonymous"></script>
</body>

</html>

</html>
```

Además del conenido de este documento, en https://getbootstrap.com/docs/4.0/getting-started/introduction/ se encuentra toda la información necesaria para poder tabajar con Bootstrap sin inconvenientes.
También en sitios como https://openclassrooms.com/en/courses/3273226-dive-into-bootstrap se encuentran ejemplos para poder realizarlos de forma interactiva. E investigando un poco más, en la Web encontrarán miles de ejemplos que les servirán para poder realizar el trabajo práctico completo de Bootstrap.