Table of Contents

Contenidos

El diseño de webs tipo "responsive" permite crear webs adaptables a diferentes tamaños de pantalla, desde webs para pantallas tamaño escritorio, pasando por tablets, hasta webs para móviles. Este tipo de diseño se basa en crear un único código y utilizar reglas y estilos CSS para adaptar los contenidos a los diferentes tamaños de pantalla.

Los contenidos principales del libro son:

Introducción al diseño "*responsive*"

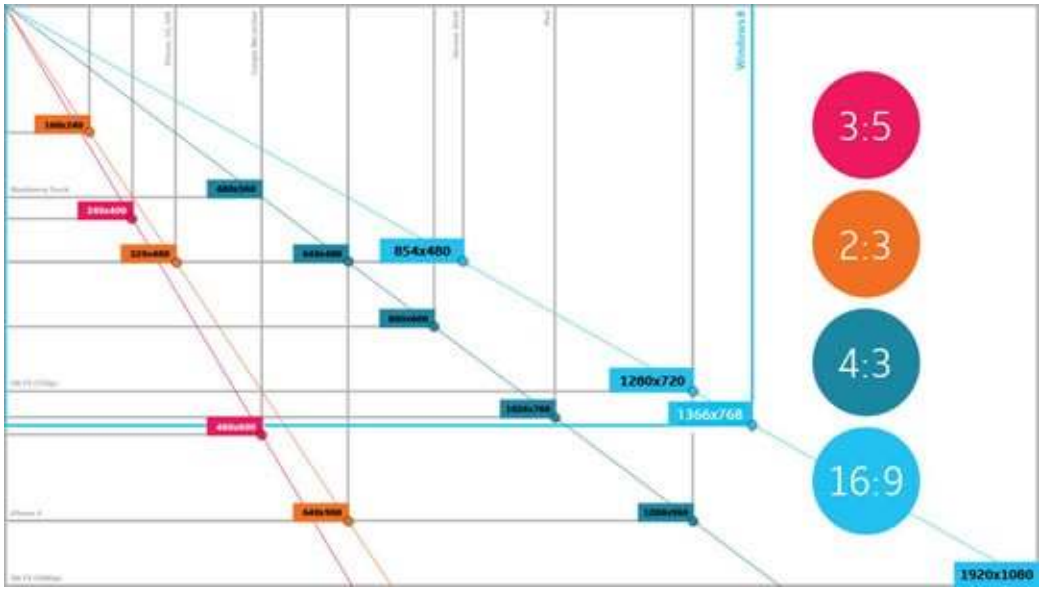
El diseño web *responsive*, adaptable o adaptativo, conocido por las siglas *RWD* (del inglés, *Responsive Web Design*) es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla. Hoy día las páginas web se visualizan en multitud de tipos de dispositivos como tabletas, *smartphones*, libros electrónicos, portátiles, PCs, etc. Esta tecnología pretende que con un solo diseño web tengamos una visualización adecuada en cualquier dispositivo.

El diseño *responsive* se basa en proporcionar a todos los usuarios de una web los mismos contenidos y una experiencia de usuario lo más similar posible, frente a otras aproximaciones al desarrollo web móvil como la creación de *apps*, el cambio de dominio o webs servidas dinámicamente en función del dispositivo.

Aunque todas tienen pros y contras, la web *responsive* es considerada por muchos expertos como la mejor práctica posible, al unificar la web, reducir tiempos de desarrollo y ofrecer grandes ventajas para SEO móvil.

Variabilidad en las resoluciones de pantalla

Durante muchos años el desarrollo web se ha basado en la resolución estándar de 1024×768 (hace apenas 3 años aproximadamente el 40% de los usuarios tenía esta resolución). Pero en la actualidad existe una amplia variedad de resoluciones, no solo en pantallas de ordenadores de escritorio sino también para *tablets* y dispositivos móviles.



Es muy importante conocer todas estas estadísticas así como cuales son las dimensiones de pantalla de los usuarios, a qué público vamos dirigidos, etc. y así poder tenerlo en cuenta en la usabilidad de nuestra web. Ya no es posible centrar el desarrollo pensando que los usuarios van a tener (en un alto porcentaje) una única resolución de pantalla.

Desde hace ya unos años en el desarrollo web se ha sustituido en cierta medida el problema de la compatibilidad de navegadores (gracias a que poco a poco todas las compañías se están ciñendo a los estándares con HTML5/CSS3 y otras se basan directamente en web-kit) por el problema de las resoluciones de los dispositivos.

En la siguiente tabla se pueden ver las últimas estadísticas (2014) de las resoluciones de pantalla más utilizadas:

Resolución	**% utilización**
> 1920x1080	34%
1920x1080	13%
1366x768	31%
1280x1024	8%
1280x800	7%
1024x768	6%
800×600	0.5%
< 800×600	0.5%

En la actualidad ya no es 1024x768 la resolución más utilizada, sino que es 1366×768 y resoluciones superiores a 1920x1080.

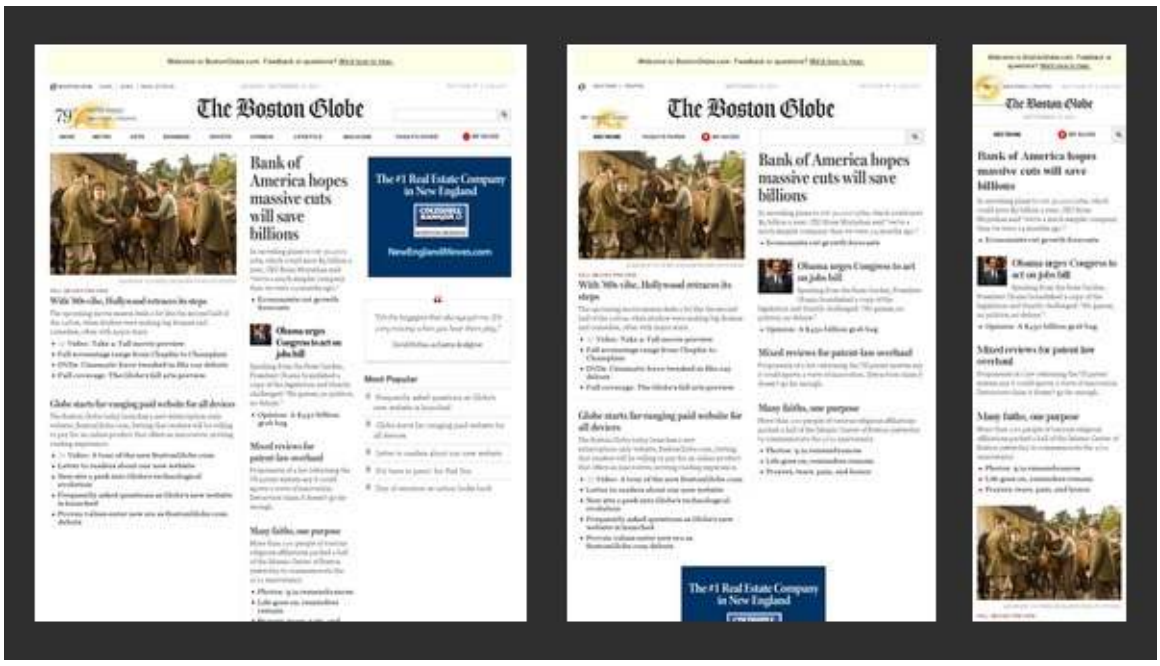
Es fundamental tener en cuenta que en el diseño *responsive*, al variar tanto las posibles resoluciones en las que se verá nuestra web deberemos mostrar en primer lugar los contenidos más importantes e imprescindibles.

Ejemplos de sitios web creados con tecnología *Responsive*

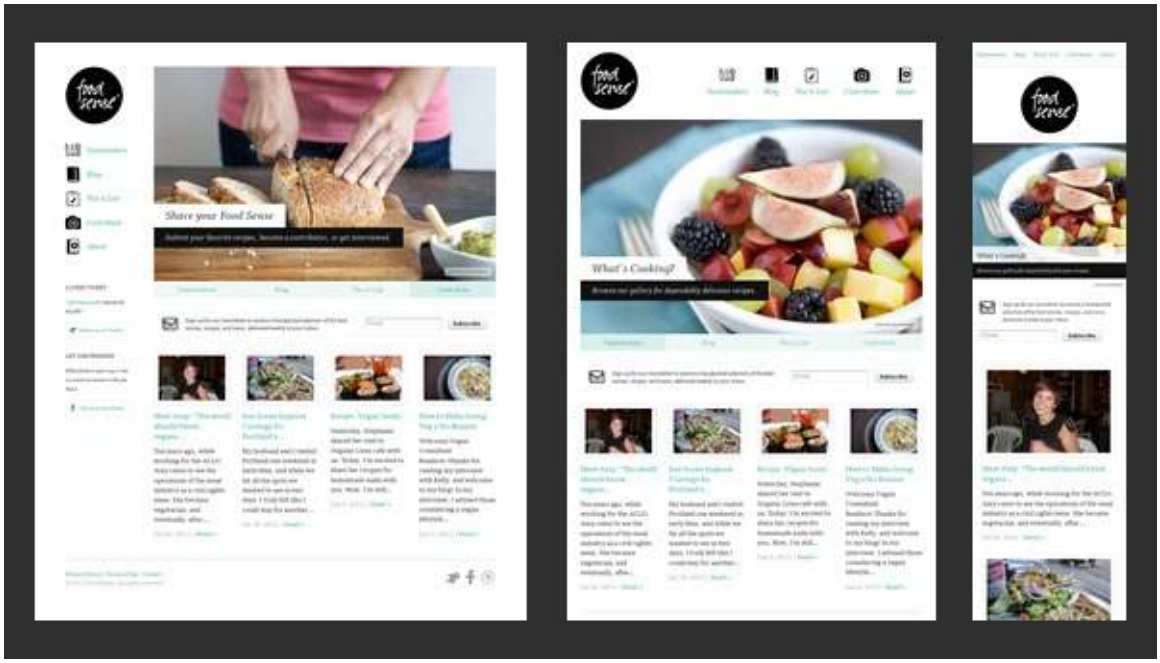
En un artículo llamado: "*Responsive Web Design: 50 Examples and Best Practices*" muestra excelentes ejemplos de la aplicación de esta tecnología. Algunos de estos ejemplos son:

dConstruct 2011

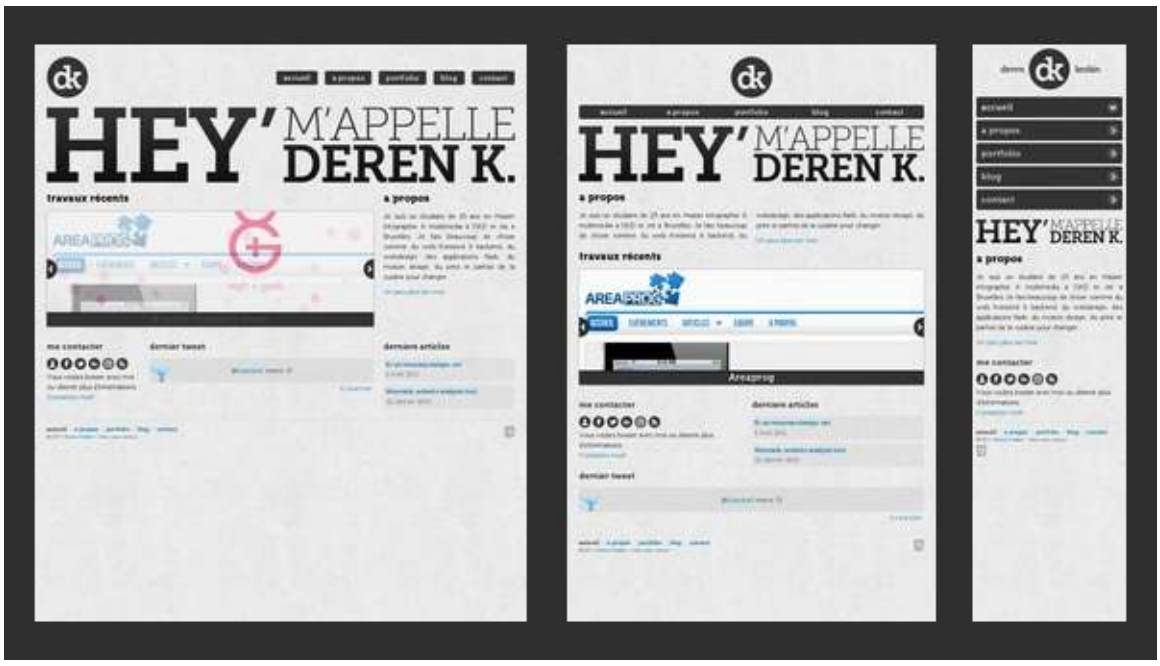
Boston Globe



Food Sense



Deren keskin



Frameworks responsive

Como se suele decir, en vez de reinventar la rueda y programar nosotros todo el diseño *responsive*, podemos aprovechar algunos de los *frameworks* que existen en el mercado para este propósito. Nos ahorrarán muchísimo tiempo, partiremos de código ampliamente probado, y de unos diseños base de todos los elementos web bastante más bonitos que la que tendrían de forma nativa.

Actualmente existen en el mercado una amplia variedad de este tipo *frameworks responsive*, algunos de los más utilizados son:

- **Bootstrap** (http://getbootstrap.com/): Este framework es uno de los más populares del mercado, habiendo sido desarrollado por el equipo de Twitter. Bootstrap ha sido creado pensando en ofrecer la mejor experiencia de usuario tanto a usuarios de PC (IE7 incluido!), como a smartphones y tabletas. Utiliza un grid responsive de 12 columnas y trae integrado decenas de complementos, plugins de JavaScript, tipografía, controladores de formularios y mucho más. Además utiliza el preprocesador de CSS LESS.
- **Foundation** (http://foundation.zurb.com/): Junto con Bootstrap es uno de los *frameworks* más avanzados que existen en la actualidad. Ha sido desarrollado con SASS, un potente preprocesador de CSS que hace de Foundation un *framework* fácilmente personalizable. Además saca partido de las nuevas tecnologías y funciona con IE8+.
- **Skeleton** (http://getskeleton.com/): Skeleton es un *boilerplate* que ofrece un grid responsive basado en una resolución de 960px que se ajusta al tamaño de los dispositivos móviles. Tiene poco peso e incluye una colección de archivos CSS y JS para facilitarnos el diseño de nuestra web.
- **HTML5 Boilerplate** (http://html5boilerplate.com/): Al igual que los demás nos ofrece un set de utilidades para construir nuestra web responsive de forma rápida y sencilla, con la ventaja de ser uno de los que menos ocupan.

En este curso nos vamos a centrar en **Bootstrap** por ser uno de los *frameworks* más completos, más utilizados y que mejor funcionan. En las siguientes secciones estudiaremos en detalle el funcionamiento de esta librería.

Bootstrap

Como ya hemos comentado antes, Bootstrap es uno de los *frameworks* más populares y utilizados del mercado para la creación de páginas *responsive*, habiendo sido desarrollado por el equipo de Twitter.

Entre los navegadores soportados se encuentran Chrome, Firefox, Opera, Safari e Internet Explorer a partir de la versión 8 (aunque en la versión 7 también funciona correctamente).

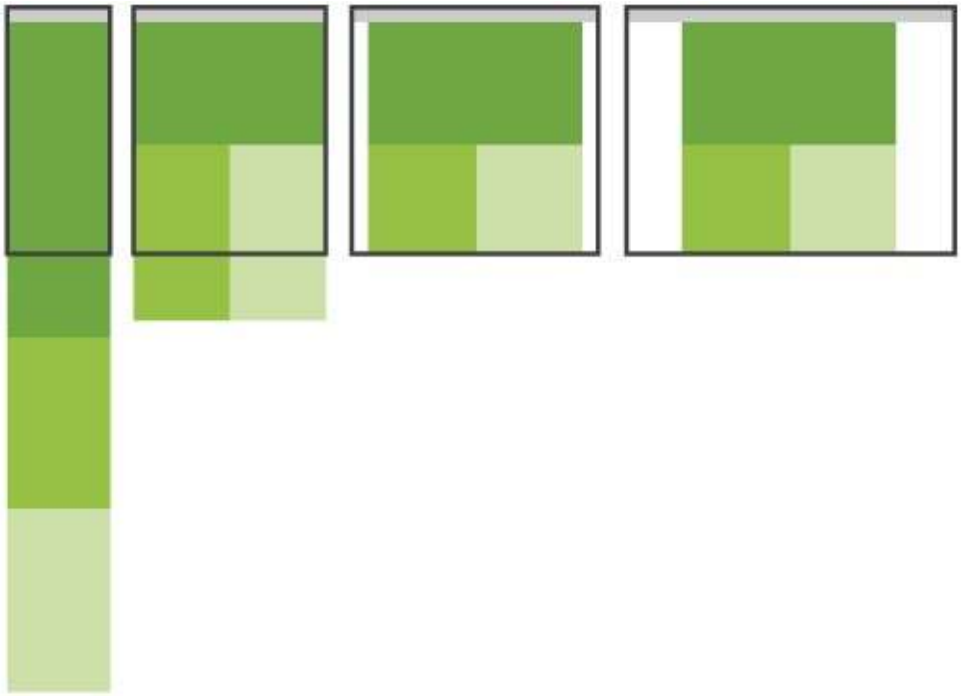
Está preparado para funcionar tanto en navegadores de PCs y portátiles con cualquier tamaño de pantalla así como para *tablets* y *smartphones* de tamaños mucho más reducidos.

Para conseguir que una misma web se pueda visualizar correctamente en todos esos tamaños de pantalla ha diseñado un avanzado sistema de rejilla dividido en columnas para el posicionamiento de los elementos de nuestra web. Además incorpora otras muchas utilidades y complementos (formularios, botones, barras de navegación, etc.) para simplificar el desarrollo de una web *responsive*.

Funcionamiento del diseño adaptable

El diseño *responsive* se basa en adaptar dinámicamente el diseño web en función de la resolución de la pantalla del visitante. De esta forma adaptamos nuestras webs a dispositivos móviles sin necesidad de tener dos sitios separados y al mismo tiempo también podemos adaptar la web a resoluciones grandes para mejorar la experiencia de usuario.

Antiguamente se pensaba en hacer 2 diseños, uno para móviles y otro para web, sin embargo, el diseño *responsive* trata de estructurar o adaptar el contenido que ya tienes en el diseño original a otros formatos diferentes: móviles, *tablets* y versión de escritorio, como bien muestra esta imagen:



La solución técnica que se le ha dado en el desarrollo web al problema de esta diversidad de resoluciones web se llama *Responsive Web Design* y nos permite hacer interfaces adaptadas al entorno del usuario mediante estructuras, bloques, columnas e imágenes fluidas gracias a *media-queries* de CSS.

A partir de CSS 2.1 las hojas de estilo han incluido los *media types*, lo cual nos ha facilitado, por ejemplo, proveer un estilo distinto para el diseño de impresión:

```
<link rel="stylesheet" type="text/css" href="core.css" media="screen" />
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

A partir de CSS 3 el W3C creó las *media queries*. Una media query nos permite apuntar no sólo a ciertas clases de dispositivos, sino realmente inspeccionar las características físicas del dispositivo que está renderizando nuestro trabajo. Para utilizarlas podemos incorporar una *query* al atributo media de un *link* a una hoja de estilos:

```
<link rel="stylesheet" type="text/css" href="shetland.css"
        media="screen and (max-device-width: 480px)" />
```

La *query* contiene dos componentes:

- Un media type (*screen*, *print* o *all*).
- La consulta entre paréntesis, conteniendo una característica a inspeccionar (max-device-width o min-device-width) seguida por el valor al que apuntamos (480px).

También es posible utilizarlas directamente en el CSS como parte de una regla `@media`:

```
@media screen and (max-device-width: 480px) {
    .column {
        float: none;
    }
}
```

Por ejemplo, si quisiéramos crear un estilo de bloques *fluidos* que para pantallas grandes se muestre uno a continuación del otro y para pantallas pantallas cambie a mostrarse de forma apilada, uno encima de otro, podríamos hacer algo como:

```
@media all and (max-width: 800px) {
    .bloque{
        display: block !important;
        /* Cuando el ancho sea inferior a 800px el elemento será un bloque */
        width: auto !important;
    }
}
.bloque {
    display: inline-block;    /* Para que se muestren los bloques en línea */
    height:300px;
    width: 300px;
    border:1px solid #333;
    background: #999;
    margin:20px;
}
```

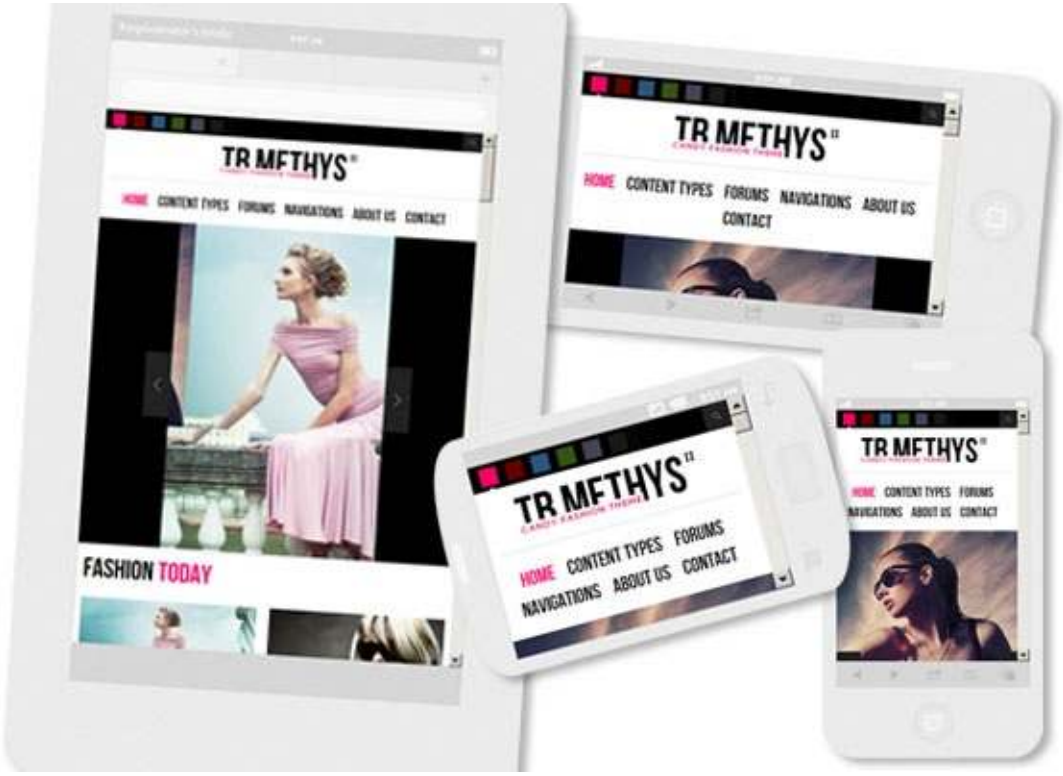
Para más información podéis consultar: http://www.w3.org/TR/css3-mediaqueries/

Probando el responsive

Para probar nuestros diseños *responsive* tenemos varias opciones, una de ellas es usar algunas de las webs que existen para tal fin. Como por ejemplo:

Responsinator (http://www.responsinator.com)

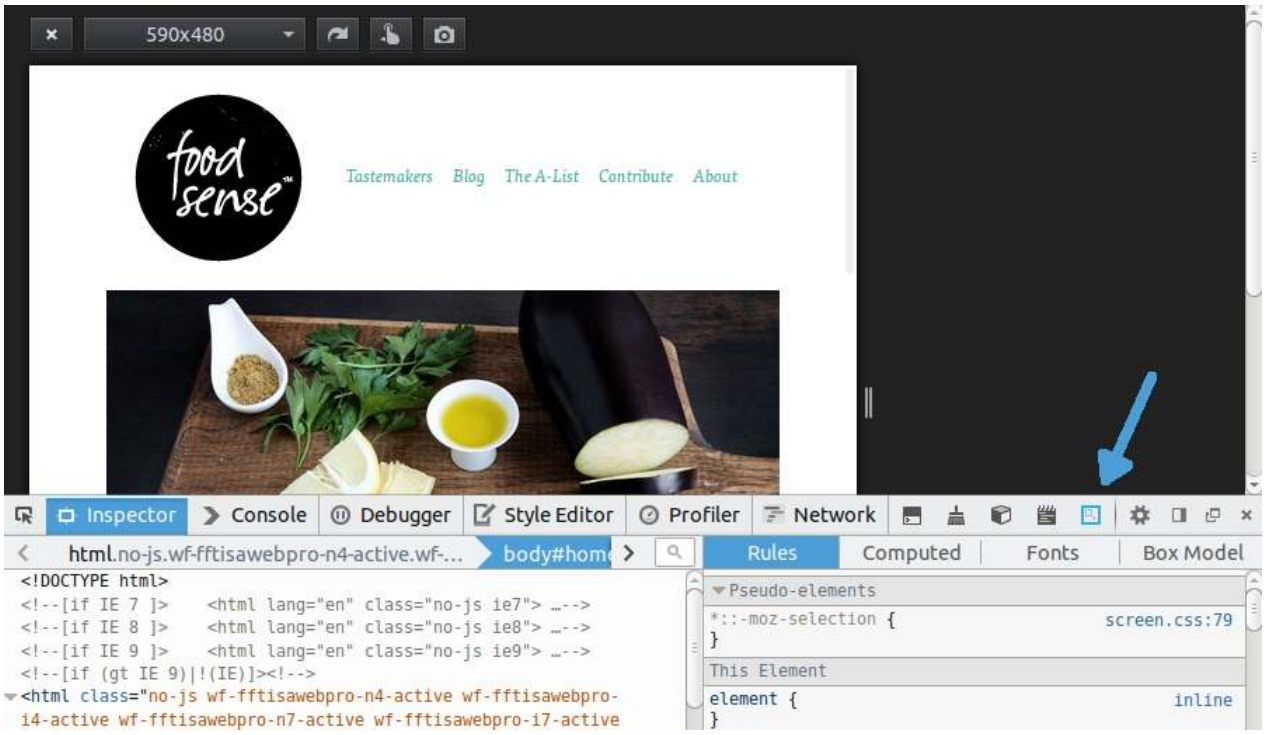
Esta herramienta está disponible solamente de forma *online*, pero nos permite ver de un solo vistazo como se mostraría nuestra web con el tamaño de los *smarthones* y *tablets* más populares, como por ejemplo las diferentes versiones de iPhone, iPad, Kindle y algunas versiones de teléfonos Android.



El problema de estas herramientas es que tenemos que acceder a una versión publicada de nuestra web (no permiten localhost) y son un poco más lentas para realizar pruebas continuas, por esta razón es mucho más recomendable utilizar alguno de los kits de herramientas para el desarrollador web que existen para los diferentes navegadores.

Herramientas del navegador para el desarrollador

Tanto en Firefox como Chrome viene instalado por defecto una serie de herramientas de ayuda para el desarrollador que nos permiten, entre otras cosas, ver la consola de mensajes, inspeccionar el código o ver la secuencia de llamadas al servidor.



Además de estas también existen otras herramientas más avanzadas que podemos instalar como una extensión de nuestro navegador, como por ejemplo Firebug.

La ventaja de estas herramientas frente a las anteriores es que son muchos más rápidas, nos permiten probar nuestra página en local y además podemos inspeccionar el código y modificar los estilos en tiempo real. Usando el inspector de estas herramientas nos podemos ahorrar mucho tiempo a la hora de realizar pruebas sobre la propia página cargada, ya que de otra forma tendríamos que modificar el código directamente, recargar la página y volver a probarlo.

Página básica

Bootstrap utiliza ciertos elementos HTML y propiedades CSS que requieren el uso del *doctype* de HTML 5 para que funcionen, por lo que es importante añadirlo a todas nuestras páginas:

```
<!DOCTYPE html>
<html lang="en">
  ...
</html>
```

Además para asegurar que se muestra correctamente en dispositivos móviles y que permite la utilización del zoom al arrastrar tenemos que añadir la siguiente etiqueta `meta` dentro de la cabecera `<head>` :

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

La propiedad `width` controla el tamaño del viewport. Puede definirse con un número en pixeles como `width=600` o con un valor especial como `device-width` que es el equivalente al ancho de la pantalla en píxeles del dispositivo que cargue la web. El atributo `shrink-to-fit="no"` configura este mismo comportamiento para los navegadores Safari anteriores a la versión 9.

La propiedad `initial-scale` del viewport controla el nivel de zoom cuando la página se carga por primera vez. Las propiedades `maximum-scale` , `minimum-scale` , y `user-scalable` controlan la forma en cómo se permite a los usuarios aumentar o disminuir el zoom en la página. Si añadimos a la etiqueta *meta* del _viewport _el atributo `user-scalable=no` (como se puede ver en el ejemplo inferior) podemos deshabilitar el zoom para dispositivos móviles. De esta forma los usuarios únicamente podrán usar el scroll de la aplicación, haciendo tu web más similar a una aplicación nativa. Sin embargo, hay que usar esta característica con cuidado ya que no es recomendable para todos los sitios.

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

                              maximum-scale=1, user-scalable=no">
```

A continuación se incluye la plantilla HTML base para cualquier proyecto con Bootstrap, a partir de la cual se tendrán que ir añadiendo el resto de elementos:

```
<!doctype html>
<html lang="en">
  <head>
    <title>Hello, world!</title>
    <!-- Required meta tags -->
    <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-beta.
2/css/bootstrap.min.css" integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPv
rszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-K
J3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymou
s"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.mi
n.js" integrity="sha384-vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpF
gh" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.m
in.js" integrity="sha384-alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZ
flJ" crossorigin="anonymous"></script>
  </body>
</html>
```

Como se puede ver hemos añadido la etiqueta meta de cabecera y hemos cargado la hoja de estilo de Bootstrap y las librerías Javascript que vamos a necesitar (JQuery, Bootstrap y Popper). También hemos añadido la etiqueta h1 con "Hello world! dentro del cuerpo de la web, que será donde podremos empezar a escribir el contenido de nuestro sitio web responsive.

Sistema de rejilla

El sistema de rejilla de Bootstrap se basa en la creación o disposición del contenido de nuestra web dentro de rejillas flexibles, las cuales se escalarán al tamaño y posición adecuada de forma automática dependiendo del tamaño de la pantalla en la que se rendericen.

Elemento contenedor

El sistema de rejilla tiene que ser utilizado dentro de uno de los dos elementos contenedores que provee Bootstrap: `container` ó `container-fluid`. Es importante tener en cuenta que estos elementos se utilizan como raíz de la rejilla y no se podrán anidar unos dentro de otros.

Si lo que queremos es que el contenido de nuestra web aparezca centrado y con un ancho fijo entonces podemos utilizar la etiqueta `.container`, de la forma:

```
<div class="container">
  ...
</div>
```

Por el contrario, si queremos que el contenido de nuestra web pueda ocupar todo el ancho disponible (hay que tener en mente todos los tamaños de pantalla, incluso las muy grandes), podemos usar la etiqueta `.container-fluid`:

```
<div class="container-fluid">
  ...
</div>
```

En las siguientes imágenes se ejemplifica el resultado obtenido para un mismo ancho al aplicar los dos tipos de contenedores, container en el primer caso y container-fluid en el segundo. El comportamiento de estos elementos ante distintos tamaños de pantalla es el siguiente: el elemento "container-fluid" siempre se adapta al 100% del tamaño de la pantalla, mientras que el tipo "container" tiene un tamaño máximo, por lo que si el ancho de la pantalla es superior a este ancho el contenido aparecerá centrano, dejando un margen a cada lado, y si el ancho de la pantalla es igual o inferior al tamaño máximo del contenedor, entonces se adaptará al ancho disponible.

Funcionamiento del sistema de rejilla

El sistema de rejilla está pensado para ayudarnos en la disposición de los contenidos de nuestra web y su adaptación a los diferentes tamaños de pantalla de forma automática. Para ello tenemos que poner el contenido dentro de celdas o columnas que irán dentro de filas. Cada fila se puede dividir hasta en 12 columnas, pero seremos nosotros los que definiremos el número de columnas deseado para cada tamaño de pantalla.

A continuación se detalla el funcionamiento de este sistema:

- Las columnas irán agrupadas dentro de filas (`.row`).
- Las filas (`.row`) se deben colocar dentro de una etiqueta contenedora: `.container` (para ancho fijo) o `.container-fluid` (para poder ocupar todo el ancho), esto permitirá alinear las celdas y asignarles el espaciado correcto.
- El contenido se debe disponer dentro de columnas o celdas, las cuales deben de ser el único hijo posible de las filas (`.row`), las cuales, a su vez, serán el único hijo posible del contenedor (`.container` o `.container-fluid`).
- Al seguir este orden el sistema de rejilla funcionará correctamente, creando el espaciado interior y los márgenes apropiados dependiendo de las dimensiones de la pantalla.
- Cada fila se puede dividir hasta un máximo de 12 columnas, pero somos nosotros los que tendremos que definir el número de columnas en el que queremos dividir cada fila y su ancho para cada tamaño de pantalla. Por ejemplo: 3 columnas de igual ancho.
- Si el tamaño total de las columnas de una fila excede de 12 el tamaño sobrante se colocará en la siguiente fila.
- El tamaño de las columnas se especificará con clases css que Bootstrap define para cada tamaño de pantalla, por ejemplo `.col-md-XX` , donde `XX` es el tamaño de la columna, que podrá tomar valores entre 1 y 12.

En la siguiente tabla se muestra un resumen del sistema de rejilla de Bootstrap, su comportamiento según el tamaño del dispositivo y las clases CSS que nos permiten controlarlo:

Pantalla	**Dimensiones**	**Prefijo de la clase**	**Ancho del contenedor**
Tamaño extra pequeño	< 576 px	**.col-**	Ninguno (automático)
Tamaño pequeño	≥ 576 px	.col-sm-	540px
Tamaño medio	≥ 768 px	.col-md-	720px
Tamaño grande	≥ 992 px	.col-lg-	960px
Tamaño extra grande	≥ 1200 px	.col-xl-	1140px

Es importante destacar que al definir estas clases no solo se aplican para ese tamaño de pantalla sino para los superiores también. Por ejemplo, al indicar el tamaño de las columnas con las clases para *tablets* (*.col-sm-*), también se aplicará para los tamaños de pantalla medianos y grandes (si no hubieran otras clases para estos tamaños que los sobreescribieran). Es decir, nos tenemos que fijar que en la tabla anterior el tamaño se indica con el símbolo de mayor o igual (≥) (o de menor para el caso de xs) a un tamaño dado, y por lo tanto se aplicará esa disposición a partir de ese tamaño, a no ser que se indique otra cosa.

Bootstrap está diseñado pensando en los dispositivos móviles primero (o como ellos indican: siguiendo la estrategia mobile first). Por lo tanto todos los tamaños y dimensiones están pensadas para los dispositivos móviles, y para tamaños más grandes lo que hacen es adaptar o escalar estos tamaños.

Si nos fijamos en la tabla anterior podremos ver que para el tamaño extra pequeño el prefijo de la clase que se define es "**.col-**" (a diferencia de los demás que añaden un sufijo para el tamaño de pantalla). Cuando indiquemos el tamaño de las columnas usando esta clase se aplicará para todos los tamaños, a no ser, como ya hemos dicho, que se indique otra clase para otro tamaño mayor que defina otra disposición.

A continuación veremos diferentes formas de indicar el número de columnas que conforman cada fila, usando el sistema automático, especificando el ancho o bien usando un sistema mixto.

Columnas de ancho específico

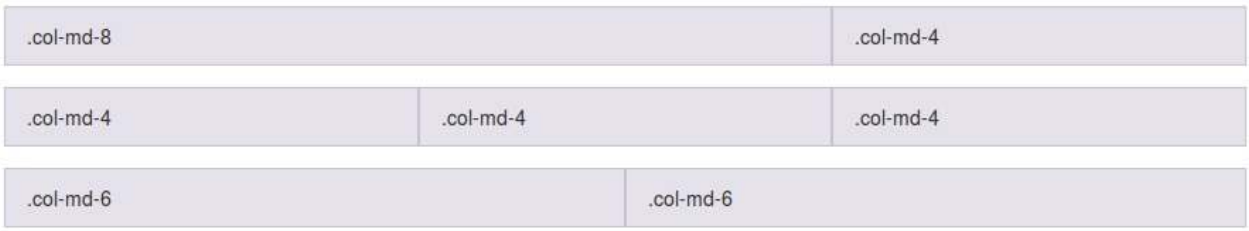
A continuación se incluyen algunos ejemplos de uso del sistema de rejilla que nos ayudarán a comprender mejor su funcionamiento.

Selección de tamaño de las columnas solo para pantallas medianas

En el siguiente ejemplo se han creado 3 filas, la primera dividida en 2 columnas de tamaño desigual, la segunda en 3 columnas de igual tamaño y la tercera en 2 columnas también de igual tamaño.

```
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

En la siguiente imagen se puede ver el resultado para pantallas de tamaño mediano (tamaños de md en adelante):



> Para poder visualizar las columnas se ha añadido una clase CSS que establece color para el borde y el fondo de las cajas. Por defecto, ni la etiqueta DIV ni las etiquetas .col-* establecen apariencia (ni color de borde ni de fondo), solamente establecen la anchura, y por lo tanto al renderizar el código anterior tal cual las cajas se verán transparentes.

Dado que las columnas se han especificado únicamente mediante las clases `.col-md-*` esto creará estas divisiones solo para las pantallas medianas y grandes, pero no para los tamaños de pantalla más pequeños. En este último caso las columnas se ampliarán para ocupar todo el ancho y por lo tanto se mostrarán apiladas de la forma:

.col-md-8

.col-md-4

.col-md-4

.col-md-4

.col-md-4

.col-md-6

.col-md-6

Selección de dos tamaños de columna: pequeño y mediano

Si no queremos que las columnas se muestren apiladas para tamaños de pantalla pequeños podemos indicar también la disposición para esos casos mediante las clases `.col-*` además de las que ya teníamos con `.col-md-*`. Por ejemplo:

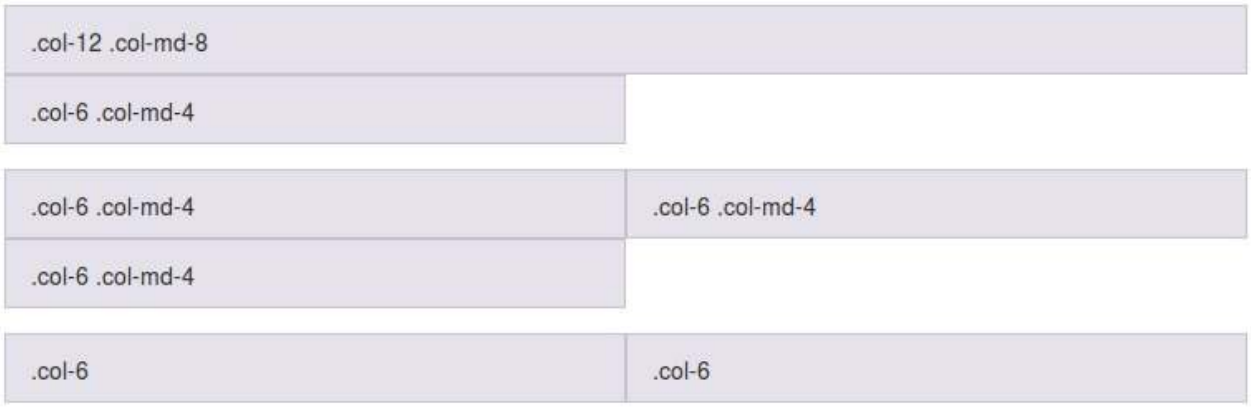
```
<!-- En pantallas pequeñas aparecerá una columna que ocupará todo el ancho
y otra que ocupará la mitad de la pantalla -->
<div class="row">
  <div class="col-12 col-md-8">.col-12 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- En pantallas medianas se indica que cada columna ocupe la mitad
del ancho disponible -->
<div class="row">
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>

<!-- Como no se indica el tamaño para pantallas grandes las columnas
siempre ocuparán el 50% -->
<div class="row">
  <div class="col-6">.col-6</div>
  <div class="col-6">.col-6</div>
</div>
```

En la siguiente imagen se puede ver como quedaría el código de ejemplo para pantallas medianas y grandes (tamaños de md en adelante):

En el caso de pantallas pequeñas las columnas se verían de la forma:



Selección de tres tamaños: extra pequeño, pequeño y mediano

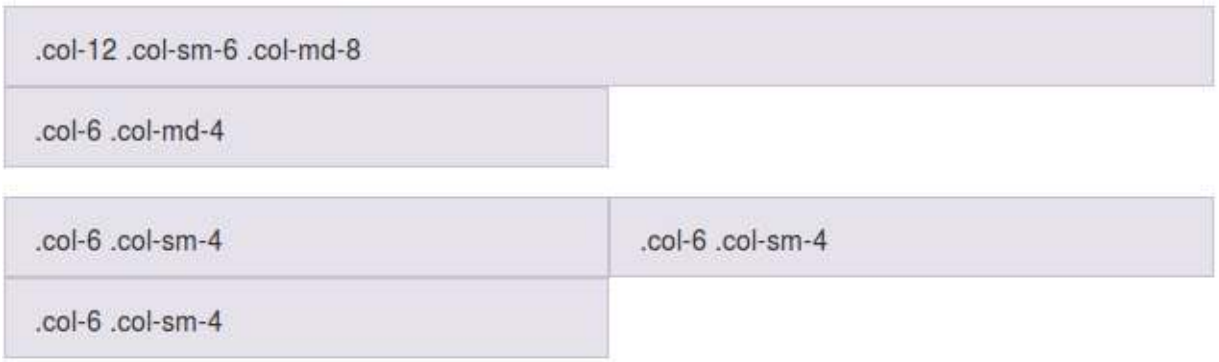
Si queremos tener un mayor control podemos especificar también el tamaño de las columnas para las pantallas tipo *small* con las clases `.col-sm-*`. Por ejemplo:

```
<div class="row">
  <div class="col-12 col-sm-6 col-md-8">.col-12 .col-sm-6 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
<div class="row">
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
</div>
```

A continuación se incluye una previsualización de este código de ejemplo para pantallas medianas y grandes (tamaños md, lg y xl):

El mismo código pero en pantallas tipo *small* (tamaño sm) se mostraría como:

Y en el caso de pantallas pequeñas se vería de la forma:



Además de los tres tamaños indicados en este último ejemplo para la primera columna (`.col-12 .col-sm-6 .col-md-8`) podríamos añadir también, si lo necesitamos, el tamaño para pantallas grandes y extra grandes con col-lg y col-xl-. Por ejemplo, podríamos haber definido la siguiente columna:

```
<div class="col-12 col-sm-6 col-md-8 col-lg-9 col-xl-10">...</div>
```

En resumen, podemos indicar **para cada columna** todos los tamaños que queramos de entre los disponibles (con .col-, .col-sm-, .col-md-, .col-lg- y .col-xl-). Sin embargo, **esto solo lo tendremos que hacer** cuando necesitemos establecer un ancho de columna distinto para cada tamaño de pantalla. Si para todos los tamaños de pantalla necesitamos el mismo ancho entonces utilizaremos solamente la clase .col-. Es decir, no tendría sentido escribir algo como "col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6", ya que se conseguiría el mismo efecto que si hubieramos puesto solamente "col-6".

Por lo tanto, solo añadiremos más de una clase cuando necesitemos establecer anchos distintos entre esos tamaños, y además si solo vamos a diferenciar entre 2 tamaños solo será necesario usar 2 etiquetas. Por ejemplo si queremos diferenciar solamente entre movil y escritorio solamente tendríamos que añadir la clase .col- y la clase .col-md-.

Columnas de ancho automático

A partir de la versión 4 de Bootstrap podemos utilizar las columnas de ancho automático, es decir, indicar únicamente el número de columnas que queremos y el sistema calculará automáticamente su anchura. Para esto podremos usar **la clase "** `.col` **"**, sin número de columnas ni tamaño de pantalla, por ejemplo:

```
<div class="container">
  <div class="row">
    <div class="col">1 of 2</div>
    <div class="col">2 of 2</div>
  </div>
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col">2 of 3</div>
    <div class="col">3 of 3</div>
  </div>
</div>
```

Con este código obtendríamos un resultado similar al de la siguiente figura, donde en primer lugar se crea una fila con dos columna de igual ancho, y a continuación se añade una segunda fila con tres columnas de igual ancho.

El número de columnas del ejemplo anterior se mantendrá para todos los tamaños de pantalla, adaptando el ancho de las columnas para cada uno de ellos.

A esta clase podemos añadir el sufijo para establecer el tamaño de pantalla, por lo que disponemos de las clases `.col, .col-sm, .col-md, .col-lg, .col-xl`. En todos los casos estaremos indicando que queremos una columna de ancho automático desde el tamaño de pantalla indicado en adelante. Debemos de tener en cuenta que si indicamos algo como `<div class="col col-sm col-md">` el resultado que obtendríamos sería el mismo en todos los tamaños, una columna de ancho automático, equivalente a haber indicado únicamente `<div class="col">`. Por lo tanto, el uso de estos sufijos solo se justicará cuando queramos una columna de ancho automático solamente de un tamaño en adelante, y que por lo tanto, para los tamaños inferiores se cree una columna que ocupe todo el ancho. Por ejemplo, veamos el siguiente código:

```
<div class="row">
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
</div>
```

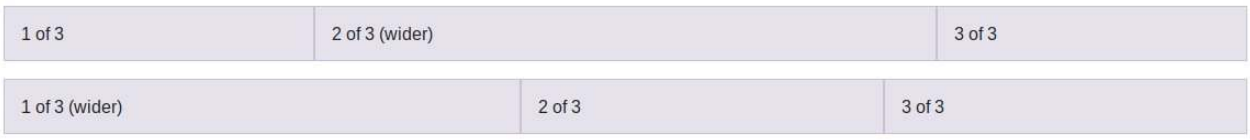
Estamos indicando que queremos una fila con tres columnas de ancho automático para los tamaños desde sm en adelante, y por lo tanto, para el tamaño extra pequeño estas tres columnas pasarán a ocupar todo el ancho, transformándose en tres filas completas.

Modo mixto

Estas columnas de ancho automático se pueden mezclar **en una misma fila** con las columnas de ancho específico que hemos visto antes. La forma de calcular el ancho de cada columna será el siguiente: En primer lugar se calculará el tamaño de las columnas de ancho específico y a continuación se rellenará el espacio restante usando las columnas de ancho automático. Por ejemplo, a continuación vamos a definir dos filas mezclando ambos tipos de columnas:

```
<div class="container">
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-6">2 of 3 (wider)</div>
    <div class="col">3 of 3</div>
  </div>
  <div class="row">
    <div class="col-5">1 of 3 (wider)</div>
    <div class="col">2 of 3</div>
    <div class="col">3 of 3</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura:



Como podemos ver en este ejemplo, los sistemas para de definir las columnas **se pueden mezclar como queramos**, por ejemplo usando el de ancho fijo entre dos columnas de ancho automático, o estableciendo una columna de ancho específico al principio y después dos de ancho automático.

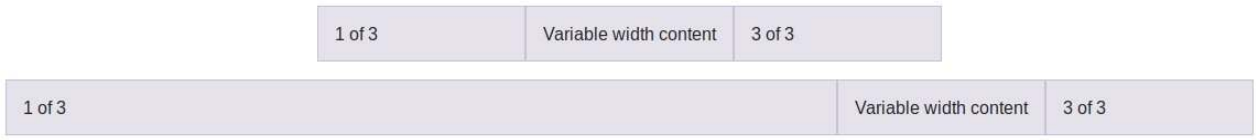
Ancho de columna variable

Con Bootstrap 4 también se introdujeron las columnas de ancho variable, las cuales ocuparán el ancho justo que se necesite según el contenido de la columna. Para utilizarlas disponemos de las clases " `.col-*-auto` ", donde * puede ser culquiera de los sufijos de tamaño de pantalla que hemos visto antes *sm*, *md*, *lg*, *xl*, o ningún sufijo (`.col-auto`) para indicar todos los tamaños.

Estas etiquetas, igual que las de ancho automático, se pueden mezclar con las de ancho específico, por ejemplo:

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">1 of 3</div>
    <div class="col-md-auto">Variable width content</div>
    <div class="col col-lg-2">3 of 3</div>
  </div>
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-md-auto">Variable width content</div>
    <div class="col col-lg-2">3 of 3</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura:



En el código anterior se ha usado la clase " `.justify-content-md-center` " para alinear el contenido dentro de una fila, estas etiquetas las veremos en la sección "Alineación". A continuación vamos a ver otras utilidades del sistema de rejilla, como el anidamiento de columnas, o cómo forzar el cambio de fila.

Forzar el cambio de fila

Mediante la clase `.w-100` podemos forzar el cambio de fila cuando nosotros queramos:

```
<div class="row">
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

  <!-- Force next columns to break to new line -->
  <div class="w-100"></div>

  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
</div>
```

Con lo que obtendríamos dos filas con dos columnas cada una:

.col-6 .col-sm-3	.col-6 .col-sm-3
.col-6 .col-sm-3	.col-6 .col-sm-3

Esta clase también nos puede ser útil para forzar el cambio de fila solo para determinados tamaños de pantalla. Para esto tenemos que combinarla con otras clases de Bootstrap que nos permiten mostrar u ocultar elementos según el tamaño de pantalla. A continuación se incluye un ejemplo:

```
<div class="row">
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

  <!-- Force next columns to break to new line at md breakpoint and up -->
  <div class="w-100 d-none d-md-block"></div>

  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
</div>
```

Donde la clase " `.d-none` " significa que no se muestre ese elemento (para ningún tamaño), y la clase " `.d-md-block` " indica que se muestre a partir del tamaño de pantalla "*md*" en adelante. Por lo tanto, el campo div marcado con "w-100" permanecerá oculto para los tamaños extra pequeño y pequeño, y por lo tanto no se activará el cambio de fila para esos dos tamaños, pero sí para los tamaños desde "md" en adelante.

En la sección "Utilidades Responsive" se explicarán las etiquetas ".d-*" que nos permirán controlar la visibilidad de cualquier elemento HTML en función del tamaño de pantalla.

Anidamiento de columnas

Una característica muy potente del sistema de rejilla es que se pueden anidar columnas dentro de otras columnas. Para esto solamente tenemos que crear una nueva fila dentro de una columna, y dentro de esta nueva fila podremos subdividirla usando también hasta 12 columnas.

Por ejemplo, en el siguiente código se crea una primera fila con una columna de tamaño 9, dentro de la cual se añade una segunda fila con dos columnas:

```
<div class="row">
  <div class="col-sm-9">
    Level 1: .col-sm-9
    <div class="row">
      <div class="col-8 col-sm-6">Level 2: .col-8 .col-sm-6</div>
      <div class="col-4 col-sm-6">Level 2: .col-4 .col-sm-6</div>
    </div>
  </div>
</div>
```

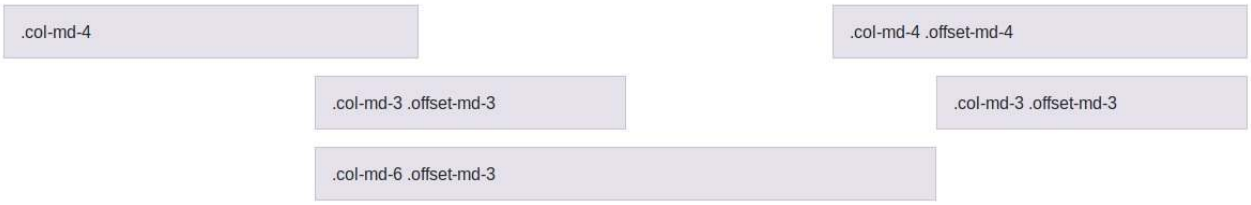
Al visualizar este código obtendríamos:

Espaciado entre columnas

Es posible crear un espaciado entre las columnas o dicho de otra forma, mover o desplazar una columna **hacia la derecha**, añadiendo un *offset* inicial mediante las clases: `.offset-*`. Por ejemplo `.offset-4` creará un espacio a la izquierda de la columna de tamaño 4 (como si se creara una columna oculta de tipo `.col-4`). En el siguiente código podemos ver un ejemplo más completo:

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
</div>
<div class="row">
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
</div>
<div class="row">
  <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
</div>
```

El cual se renderizaría de la forma:



Como se puede ver en el ejemplo anterior, también podemos especificar el offset según el tamaño de pantalla. Si usamos, por ejemplo, la clase " `offset-4` " estaremos indicando que se cree un espacio de 4 para **todos** los tamaños de pantalla; mientras que con " `offset-md-4` " se creará este espacio a partir del tamaño de pantalla "md" en adelante.

Si en algún caso necesitamos eliminar el *offset* podemos utilizar el tamaño cero (0). Por ejemplo, si especificamos un *offset* de 2 para tamaños pequeños y no queremos que dicho *offset* se aplique para pantallas medianas ni grandes tendríamos que hacer:

```
  <div class="col-sm-5 offset-sm-2 col-md-7 offset-md-0">...</div>
```

Márgenes

Además de la clase *offset* también disponemos de las clases para crear márgenes de espacio variable tanto al lado izquierdo (con " `.ml-auto` ") como al lado derecho (con `.mr-auto`) de una columna. A continuación se incluye un ejemplo:

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 ml-auto">.col-md-4 .ml-auto</div>
</div>
<div class="row">
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
</div>
<div class="row">
  <div class="col-auto mr-auto">.col-auto .mr-auto</div>
  <div class="col-auto">.col-auto</div>
</div>
```

Con lo que obtendríamos el siguiente resultado:

Como se puede ver, en la primera fila se crea un marge automático por la izquierda (y para todos los tamaños) de la segunda columna, lo que provoca el desplazamiento de esta hasta alinearla a la derecha. En la segunda fila se añade margen por la izquierda a las dos columnas para tamaños de pantalla de "md" en adelante. Y en la última fila se crea un margen automático por la derecha de la primera columna (para todos los tamaños), esto provoca un efecto similar al obtenido en la primera fila.

Ordenación de columnas

También podemos modificar el orden visual de las columnas mediante la clase `.order-`. Esta clase permite indicar la posición a la cual queremos desplazar la columna (del 1 al 12, por ejemplo `.order-1`, `.order-2`, etc.). También podemos especificar el tamaño de pantalla para el que queremos que se aplique (por ejemplo `.order-md-12`). A continuación se incluye un ejemplo:

```
<div class="container">
  <div class="row">
    <div class="col">First, but unordered</div>
    <div class="col order-12">Second, but last</div>
    <div class="col order-1">Third, but first</div>    <!-- ¡¡CUIDADO!! -->
  </div>
</div>
```

Obteniendo como resultado:

Si nos fijamos en el resultado obtenido podemos ver que **no** se obtiene el resultado esperado, la tercera columna (en color rojo) aparece en la segunda posición en lugar de en la primera como se había indicado con "`order-1`". Esto es debido a un pequeño error al cambiar el orden de derecha izquierda. En los casos en los que simplemente queramos mover una columna hacia la derecha no se producirá este error, **pero si queremos mover hacia la izquierda será necesario que establezcamos el order de todas las columnas.** Por lo tanto, para que funcione correctamente el ejemplo anterior tendríamos que escribir el siguiente código:

```
<div class="container">
  <div class="row">
    <div class="col order-2">First, but unordered</div>   <!-- Añadimos el orden de es
ta columna -->
    <div class="col order-12">Second, but last</div>
    <div class="col order-1">Third, but first</div>
  </div>
</div>
```

Obteniendo ahora sí el resultado esperado:

Como se puede ver también en este ejemplo, no es necesario que los números de columna para la ordenación sean consecutivos, simplemente se ordenarán de mayor a menor.

Hay que tener cuidado con estas clases si hay un salto de línea dentro de una misma fila (debido a que el número de columnas ocupe más de 12), ya que en estos casos el orden no funcionarán correctamente.

También disponemos de la clase " `.order-first` ", la cual nos permitirá situar cualquier elemento en primer lugar. Además, esta clase sí que funciona aunque haya un salto de línea.

Alineación

Con la nueva versión de Bootstrap también han aparecido nuevas clases que nos permiten especificar la alineación de las columnas tanto en horizontal como en vertical.

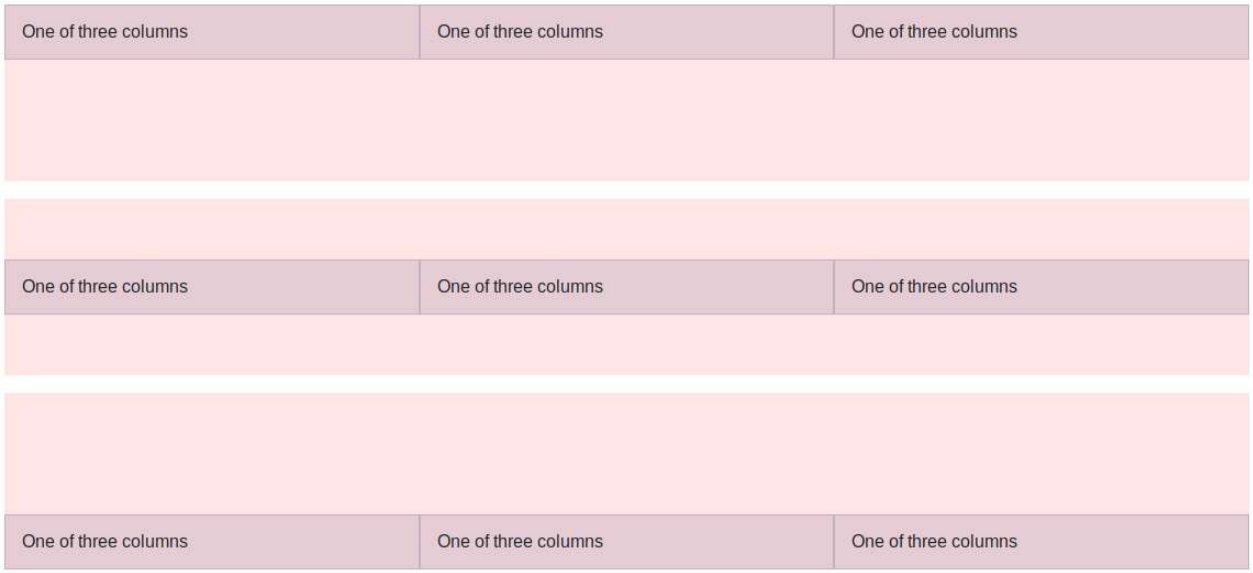
Alineación vertical

Para indicar la alineación en vertical tenemos dos opciones: indicar la misma alineación para todos los elementos de una fila o indicar la alineación a nivel de columna, lo que nos permitirá establecer distintas alineaciones para cada columna.

En el primer caso la clase CSS para la alineación la tendremos que añadir a la fila usando la etiqueta " `.align-items-*` ", donde "*" podrá ser " `start` " (al principio o pegada a la parte superior de la fila), " `center` " (alineación centrada en vertical) o " `end` " (alineación pegada al final o a la parte inferior de la fila). A continuación se incluye un ejemplo de los tres tipos de alineación:

```
<div class="container">
  <div class="row align-items-start">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
  <div class="row align-items-center">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
  <div class="row align-items-end">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado como el de la siguiente figura:



En el segundo caso, si queremos indicar por separado la alineación vertical de cada una de las columnas de una fila, tendremos que usar la clase CSS `.align-self-*`, donde "*" podrá adoptar los mismos valores: `start`, `center` o `end`. A continuación se incluye un ejemplo en el que se indican los tres tipos de alineaciones dentro de una misma fila:

```
<div class="container">
  <div class="row">
    <div class="col align-self-start">One of three columns</div>
    <div class="col align-self-center">One of three columns</div>
    <div class="col align-self-end">One of three columns</div>
  </div>
</div>
```

Con lo que obtendremos el siguiente resultado:

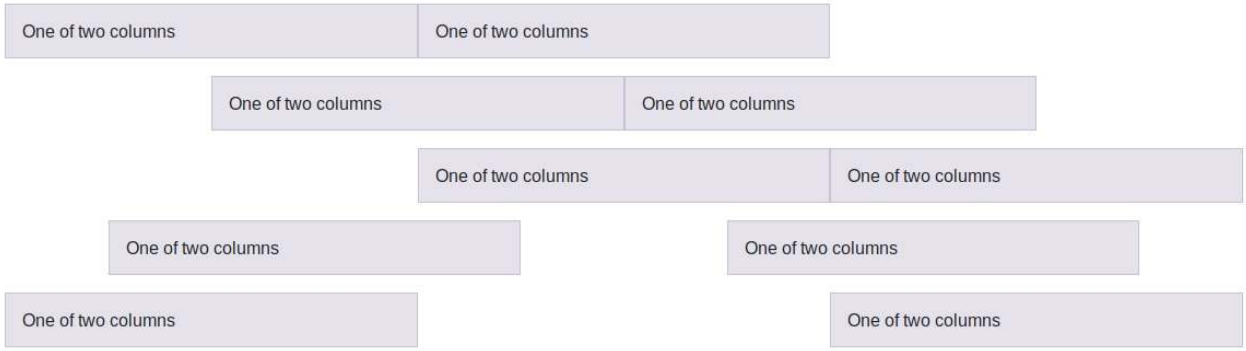
> Es importante destacar que al utilizar cualquierda de estas etiquetas de alineación, la altura de las columnas **se ajustará al contenido**, mientras que si no utilizamos ninguna etiqueta de alineación, la altura de la celda se **extenderá hasta ocupar todo el espacio disponible** en la fila.

Alineación horizontal

También podemos especificar la alineación horizontal de los elementos de una fila. Para esto disponemos de la clase " `.justify-content-*` ", donde "*" podrá ser " `start` " (izquierda), " `center` " (centrado), " `end` " (derecha), " `around` " (añadirá el mismo espacio **a ambos lados** de la columna) y " `between` " (añade espacio **entre** las columnas). A continuación se incluye un ejemplo de cada uno de estos tipos de alineación horizontal:

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura, con la fila alineada a la izquerda, la segunda centrada, la tercera alineada a la derecha, la cuarta con el espaciado "al rededor" (o a ambos lados) de las columnas, y la última con el espaciado entre las columnas.



Alineación responsive

En caso de que lo necesitemos podremos añadir también el tamaño de pantalla a las distintas clases de alineación que hemos visto: `align-items-*`, `align-self-*` y `justify-content-*`. Para esto tendremos que añadir primero el tamaño de pantalla (sm, md, lg o xl), a continuación un guión (-), y después el tipo de alineación deseado (de entre los que hemos visto), por ejemplo: `align-items-md-center`, `align-self-sm-end`, `justify-content-lg-end`, etc.

Al indicar el tamaño de pantalla dicha alineación se aplicará solamente a partir de dicho tamaño en adelante, aunque también podemos indicar distintas alineaciones para un mismo campo según el tamaño de la pantalla, por ejemplo:

```
<div class="row justify-content-center justify-content-md-start">
     ...
</div>
```

En el ejemplo anterior el contenido se alinearía de forma centrada para los tamaños de pantalla extra pequeños y pequeños, y cambiará a alineazión izquierda a partir del tamaño de pantalla "md".

Utilidades *Responsive*

Bootstrap también incluye una serie de clases para ayudarnos a mostrar u ocultar contenidos según el tamaño del dispositivo. En primer lugar vamos a ver las clases base que utlizaremos para estas acciones:

- `.d-none` : **Oculta** el elemento sobre el que se aplique.
- `.d-inline` : **Muestra** el elemento de forma "inline", es decir, permitiendo otros elementos por los lados y ocupando el ancho justo.
- `.d-block` : **Muestra** el elemento en forma de bloque, ocupando todo el ancho disponible y sin permitir otros elementos por los lados.
- `.d-inline-block:` **Muestra** el elemento en forma de bloque, pero ocupando el ancho justo y permitiendo otros elementos por los lados.

A continuación podemos ver un ejemplo del efecto obtenido al aplicar las distintas etiquetas de las que disponemos para mostrar elementos:

La diferencia entre las etiquetas " `d-inline"` y " `d-inline-block` " es el comportamiento de bloque que adopta el elemento, el cual respetará todos los márgenes y alguras que le indiquemos.

Al aplicar estas etiquetas sobre un elemento lo mostraremos u ocultaremos **para todos los tamaños**, sin embargo, si queremos podemos añadirles modificadores para indicar el tamaño **a partir del cual** queremos que se muestren u oculten. En este último caso tendremos que añadir el tamaño de pantalla entre el prefijo " `d-` " y el sufijo `none` , `inline` , `block` o `inline-block` , es decir, siguiendo el patrón " `d-*-*` ". Por ejemplo, podremos indicar `d-sm-none` para que se oculte a partir del tamaño pequeño de pantalla, `d-xl-none` para que se oculte para las pantallas extra granes, o `d-md-block` para que se muestre en forma de bloque a partir del tamaño md.

Es importante que nos fijemos que estas utilidades *responsive* se aplicarán a partir del tamaño indicado en adelante, sin embargo, ¿cómo podríamos hacer para que solamente se oculte o se muestre para un tamaño de pantalla? Para esto podemos **combinar varias clases**, por ejemplo, para que solo se oculte para el tamaño extra pequeño tendríamos que poner " `d-none d-sm-block` ", o para que solo se muestre para el tamaño pequeño usaríamos " `d-none d-sm-block d-md-none` ".

A continuación se incluye una tabla resumen de las etiquetas que tendríamos que aplicar para mostrar u ocultar solamente para un tamaño de pantalla:

Tamaños de pantalla	Mostrar	Ocultar
Solo para tamaños extra pequeños	`d-block d-sm-none`	`d-none d-sm-block`
Solo para tamaños pequeños (sm)	`d-none d-sm-block d-md-none`	`d-sm-none d-md-block`
Solo para tamaños medianos (md)	`d-none d-md-block d-lg-none`	`d-md-none d-lg-block`
Solo para tamaños grandes (lg)	`d-none d-lg-block d-xl-none`	`d-lg-none d-xl-block`
Solo para tamaños extra grandes (xl)	`d-none d-xl-block`	`d-xl-none`

Media queries

En la mayoría de los casos gracias a todas las clases que provee Bootstrap nos será suficiente para componer nuestra web. Sin embargo, en algunas situaciones es posible que queramos modificar dicho comportamiento, por ejemplo para aplicar determinados estilos CSS (colores, alineación interna, etc.) que cambien según el tamaño de pantalla. En estos casos será necesario que creemos nuestra propia *media query* para aplicar los estilos deseados.

Una *media query* se define de la forma:

```
@media (min-width: TAMAÑO-EN-PÍXELES) {
    /* Los estilos aquí contenidos solo se aplicarán a partir
    del tamaño de pantalla indicado */
}
```

En este caso, los estilos que estén dentro de esta *media query* se aplicarán **solo a partir del tamaño en píxeles indicado**. Además del tamaño mínimo, también podemos indicar el tamaño máximo o el rango de tamaño en el que se aplicarán los estilos, de la forma:

```
@media (max-width: TAMAÑO-EN-PÍXELES) {
    /* Estos estilos solo se aplicarán hasta el tamaño indicado */
}
@media (min-width: TAMAÑO-EN-PÍXELES) and (max-width: TAMAÑO-EN-PÍXELES) {
    /* Solo se aplicarán entre los tamaños indicados */
}
```

Recordamos que los rangos que define Bootstrap son:

- Pantallas extra pequeñas (móviles) < 576px
- Pantallas pequeñas (_sm, tablets _en vertical) ≥ 576px
- Pantallas medianas (md, para tablets en horizontal) ≥ 768px
- Pantallas grandes (lg, tamaño escritorio) ≥ 992px
- Pantallas extra grandes (xl, escritorio grande) ≥ 1200px

Es importante tenerlos en cuenta a la hora de definir nuestras propias *media queries*, para crear los puntos de ruptura o cambio con las mismas dimensiones. Si no lo hicieramos así estaríamos añadiendo todavía más casos a los cinco tamaños de pantalla que contempla Boostrap, y por lo tanto complicando tanto la programación como el mantenimiento del código.

Ejemplos de uso

Si por ejemplo queremos que en las pantallas extra pequeñas (xs) el color de fondo que aplica la clase `.miestilo` sea rojo y para el resto de tamaños sea verde, podríamos hacer:

```
.miestilo {
    background-color: green;
}
@media (max-width: 768px) {
    .miestilo {
        background-color: red;
    }
}
```

O si por ejemplo queremos variar la alineación del texto que se aplica en una clase a partir de las pantallas tipo escritorio:

```
.miestilo {
    text-align: center;
}
@media (min-width: 992px) {
    .miestilo {
        text-align: left;
    }
}
```

Estos sencillos ejemplos nos muestran la idea básica que tenemos que seguir para complementar el código de Bootstrap para hacer que la web se comporte como nosotros queramos. De esta forma podemos llegar a hacer cosas muy avanzadas y personalizar completamente el aspecto de una web según el tamaño del dispositivo.

Otros ejemplos de personalizaciones que podemos hacer usando las *media queries* son:

- Cambiar el tamaño y la posición de una imagen. Por ejemplo hacer que la imagen de cabecera sea muy pequeña para dispositivos móviles y mucho mayor para pantallas de escritorio.
- Cambiar la posición de cualquier elemento. Si por ejemplo tenemos un elemento que no se ve bien con una alineación en pantallas pequeñas podemos colocarlo en otro lugar.
- Cambiar el tamaño de letra, la fuente o su color. Podemos reducir el tamaño de letra de las cabeceras para pantallas xs o aumentarlo para pantallas más grandes.
- Aplicar combinaciones de estilos avanzados. Por ejemplo, podemos crear un estilo ".articulo" que según el tamaño de pantalla reajuste toda la apariencia de un artículo con todos los elementos que contenga.

- Cualquier cosa que se os ocurra!

Componentes *responsive*

Además del sistema de rejilla Bootstrap incluye un completo conjunto de componentes para facilitarnos aún más el diseño de una web *responsive*. Estos componentes aplican estilos a los elementos HTML existentes para crear un diseño más moderno y además adaptable a todos los dispositivos.

Algunos de estos componentes son:

- Barras de navegación
- Botones
- Formularios
- Tablas
- Desplegables
- y muchos más...

A continuación se explica el funcionamiento de los componentes más utilizados.

Botones

Mediante la clase `.btn` podemos aplicar estilo a los elementos tipo `button`. Esta clase la podemos combinar con `.btn-primary`, `.btn-secondary`, `.btn-success`, `.btn-danger`, `.btn-warning`, `.btn-info`, `.btn-light`, `.btn-dark` o `.btn-link` para crear botones para diferentes estados o acciones en nuestros formularios:

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```

Con lo que ontendríamos el siguiente resultado:

Elementos tipo botón

Estas clases no son exclusivas para las etiquetas `button` sino que también funcionarán de la misma forma con `<a>` y `<input>`:

```
<a class="btn btn-primary" href="#" role="button">Link</a>
<button class="btn btn-primary" type="submit">Button</button>
<input class="btn btn-primary" type="button" value="Input">
<input class="btn btn-primary" type="submit" value="Submit">
<input class="btn btn-primary" type="reset" value="Reset">
```

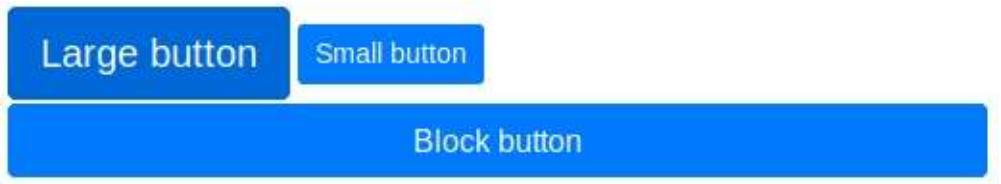
Obteniendo en todos los casos botones con la misma apariencia:

Tamaño de los botones

Podemos variar el tamaño de los botones simplemente añadiendo las clases `.btn-lg`, `.btn-sm` o `.btn-block`, para obtener botones con un tamaño más grande, más pequeño, o un botón que ocupe todo el ancho. Por ejemplo, con el siguiente código:

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-primary btn-sm">Small button</button>
<button type="button" class="btn btn-primary btn-block">Block button</button>
```

Obtendríamos el siguiente resultado:



Desplegables

Bootstrap nos facilita la creación de botones con listas de opciones desplegables mediante la clase `.dropdown`. Este elemento requiere que el *plugin* JavaScript de Bootstrap esté incluido en la plantilla. La estructura básica para crear un elemento de este tipo es la siguiente:

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButt
on" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropdown button
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</div>
```

Con lo que obtendríamos el siguiente resultado:

Sobre el botón principal podemos aplicar todos los colores de botones que hemos visto en la sección titulada "Botones", por ejemplo "`.btn-success`" o "`.btn-danger`". También podemos añadir los modificadores de tamaño "`.btn-lg`" y "`.btn-sm`" para aumentar o disminuir el tamaño del botón del desplegable.

Si nos fijamos en el código anterior, para el botón principal se ha usado la etiqueta "`button`" y para los elementos del desplegable la etiqueta "`a`", sin embargo podríamos haber usado solamente la etiqueta "`a`" o solamente la etiqueta "`button`", es decir, funcionan exactamente igual y su apariencia es la misma.

> Los atributos que empiezan con "aria" son para crear contenido accesible, para que los lectores de pantalla pueden encontrar las etiquetas correctas a la hora de interpretar el contenido. Para más información consultar la documentación sobre HTML 5 ARIA.

Para alinear un menú a la derecha se puede añadir la clase `.dropdown-menu-right` a la lista "`dropdown-menu`", por ejemplo:

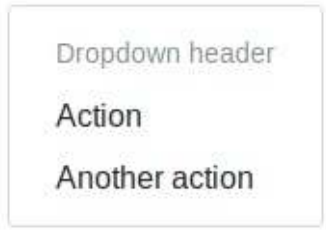
```
<div class="dropdown-menu dropdown-menu-right">
```

Encabezados en un desplegable

Para añadir un encabezado (o varios) y dividir en secciones un desplegable podemos utilizar la clase `.dropdown-header` de la siguiente forma:

```
<div class="dropdown-menu">
  <h6 class="dropdown-header">Dropdown header</h6>
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
</div>
```

Con lo que obtendremos:



Separadores en un desplegable

También podemos añadir separadores en un desplegable mediante la clase `.dropdown-divider` de la forma:

```
<div class="dropdown-menu">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">Separated link</a>
</div>
```

Con lo que obtendríamos el siguiente listado con separador:


Action
Another action
Something else here
Separated link

Grupos de botones

Podemos crear un grupo de botones en una línea agrupándolos dentro de un elemento contenedor con la etiqueta `.btn-group`.

```
<div class="btn-group" role="group" aria-label="Basic example">
  <button type="button" class="btn btn-secondary">Left</button>
  <button type="button" class="btn btn-secondary">Middle</button>
  <button type="button" class="btn btn-secondary">Right</button>
</div>
```

Con lo que obtendríamos el siguiente resultado:

> Los atributos "`role`" que utiliza Bootstrap provienen también de HTML 5 ARIA y sirven para indicar el rol o función de un elemento, en este caso se indica la agrupación de los botones.

Mediante la librería JavaScript de Bootstrap podemos añadir comportamientos tipo *checkbox* o *radio button* a un grupo de botones, para que al pulsarlos se quede marcados. Para más información consultad la documentación oficial.

Barra de botones

La barra de botones nos permite combinar grupos de botones para crear componentes más avanzados:

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">
  <div class="btn-group mr-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-secondary">1</button>
    <button type="button" class="btn btn-secondary">2</button>
    <button type="button" class="btn btn-secondary">3</button>
    <button type="button" class="btn btn-secondary">4</button>
  </div>
  <div class="btn-group mr-2" role="group" aria-label="Second group">
    <button type="button" class="btn btn-secondary">5</button>
    <button type="button" class="btn btn-secondary">6</button>
    <button type="button" class="btn btn-secondary">7</button>
  </div>
  <div class="btn-group" role="group" aria-label="Third group">
    <button type="button" class="btn btn-secondary">8</button>
  </div>
</div>
```

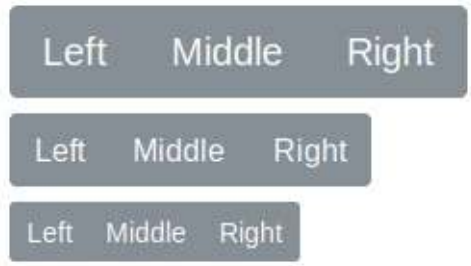
Con lo que obtendríamos el siguiente resultado:

Tamaños de los grupos de botones

Los grupos de botones también nos permiten indicar el tamaño de los botones que contienen mediante las etiquetas `.btn-group-*` , donde "*" puede ser " `sm` " o " `lg` ", por ejemplo:

```
<div class="btn-group btn-group-lg" role="group" aria-label="...">...</div>
<div class="btn-group" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-sm" role="group" aria-label="...">...</div>
```

Con lo que podríamos obtener grupos de botones de diferentes tamaños:



Grupo de botones con desplegables

También es posible añadir desplegables a los grupos de botones. Para esto el desplegable tendrá que estar contenido a su vez dentro de otro grupo de botones, de la forma:

```
<div class="btn-group" role="group" aria-label="Button group with nested dropdown">
  <button type="button" class="btn btn-secondary">1</button>
  <button type="button" class="btn btn-secondary">2</button>

  <div class="btn-group" role="group">
    <button id="btnGroupDrop1" type="button" class="btn btn-secondary dropdown-toggle"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      Dropdown
    </button>
    <div class="dropdown-menu" aria-labelledby="btnGroupDrop1">
      <a class="dropdown-item" href="#">Dropdown link</a>
      <a class="dropdown-item" href="#">Dropdown link</a>
    </div>
  </div>
</div>
```

El resultado visual obtenido sería el siguiente:

Como se puede observar en el código de ejemplo anterior, la única diferencia con un desplegable normal es que la etiqueta contenedora en vez de ser de tipo `.dropdown` es un `.btn-group`.

Formularios

Bootstrap aplica estilos a los elementos de tipo formulario para convertirlos en elementos responsive, mejorar su apariencia y permitirnos crear diferentes alineaciones. La estructura básica de un formulario es la siguiente:

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
           aria-describedby="emailHelp" placeholder="Enter email">
  </div>
</form>
```

Con lo que obtendríamos un formulario **en vertical** como el de la siguiente figura, es decir, los elementos del formulario se dispondrán en vertical, unos debajos de otros.

Para permitir que Bootstrap ajuste correctamente el espaciado, cada bloque o grupo de un formulario (normalmente formado por una etiqueta `label` y algún elemento de entrada de datos como un *input*, *textarea*, etc.) tendrá que estar agrupado por una caja contenedora con la clase `.form-group` . Además a cada *input* se le tiene que aplicar la clase `.form-control` .

Bootstrap sobrecarga y aplica estilos a los principales elementos de formulario definidos en HTML 5, como son: *text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel* y *color*.

Formulario *inline*

Mediante la utilización de la clase `.form-inline` sobre la etiqueta `<form>` podemos crear formularios que se dispondrán en una sola línea. A continuación se incluye un ejemplo de este tipo de formularios:

```
<form class="form-inline">
   <div class="form-group mx-sm-3">
       <label for="inputUser" class="sr-only">User</label>
       <input type="password" class="form-control" id="inputUser" placeholder="User">
   </div>
   <div class="form-group mx-sm-3">
       <label for="inputPass" class="sr-only">Password</label>
       <input type="password" class="form-control" id="inputPass" placeholder="Pass">
   </div>
   <button type="submit" class="btn btn-primary">Confirm</button>
</form>
```

Obteniendo el siguiente resultado:

> Aunque los campos del formulario no contengan etiquetas (*labels*) es necesario incluirlas por cuestiones de accesibilidad, para dar soporte a los lectores de pantalla. Por este motivo se han incluido en el ejemplo anterior con la clase `.sr-only` (*screen readers only*).
>
> Esta alineación tipo "inline" solo será visible para pantallas grandes. En pantallas pequeñas los elementos cambiarán a alineación vertical.
>
> En el ejemplo se ha añadido la etiqueta ".mx-sm-3" para crear un pequeño margen en los laterales de cada elemento del formulario. Para más información sobre este tipo de etiquetas consultad la documentación oficial.

Formulario horizontal

Mediante el uso del sistema de rejilla de Bootstrap podemos crear formularios con disposición en horizontal. Para esto tendremos que crear una fila (`.row`) por cada grupo, y situar la etiqueta y el input cada uno en una columna. A continuación se incluye un ejemplo:

```
<form>
  <div class="form-group row">
    <label for="inputEmail3" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3" placeholder="Email">
    </div>
  </div>
  <div class="form-group row">
    <label for="inputPassword3" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3" placeholder="Pas
sword">
    </div>
  </div>
  <div class="form-group row">
    <div class="col-sm-2">Checkbox</div>
    <div class="col-sm-10">
      <div class="form-check">
    <label class="form-check-label">
      <input class="form-check-input" type="checkbox"> Check me out
    </label>
      </div>
    </div>
  </div>
  <div class="form-group row">
    <div class="col-sm-10">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</form>
```

Con lo que obtendríamos:

Es importante que nos fijemos que la etiqueta `.row` se añade al `div` de cada grupo, manteniendo también la etiqueta " `.form-group` ". Además, podemos aplicar la clase de las columnas para las etiquetas `label` directamente sobre dicho elemento, sin necesidad de crear una caja contenedora.

Estados de validación de un formulario

Bootstrap también incluye clases para aplicar diferentes estados de validación a un formulario. Para utilizarlo simplemente tenemos que añadir las clases: `.is-valid` o `.is-invalid` sobre el propio input. De esta forma, el color de los elementos del formulario cambiará. A continuación podemos ver un ejemplo:

```
<form>
  <div class="form-group">
      <label for="validation01">First name</label>
      <input type="text" class="form-control is-valid" id="validation01"
             placeholder="First name" value="Mark" required>
  </div>
  <div class="form-group">
      <label for="validation02">City</label>
      <input type="text" class="form-control is-invalid" id="validation02" placeholder=
"City" required>
  </div>
</form>
```

Que se mostraría de la forma:

Agrupar *inputs* con otros elementos

Podemos añadir texto o botones al principio, final o a ambos lados de campo tipo `<input>`. Para esto tenemos que agrupar dicho *input* dentro de un `.input-group` y añadir dentro del grupo el elemento que queremos agrupar con la etiqueta `.input-group-addon`. A continuación se incluye un ejemplo:

```
<div class="input-group">
  <span class="input-group-addon">@</span>
  <input type="text" class="form-control" placeholder="Username">
</div>

<div class="input-group">
  <input type="text" class="form-control">
  <span class="input-group-addon">.00</span>
</div>

<div class="input-group">
  <span class="input-group-addon">$</span>
  <input type="text" class="form-control">
  <span class="input-group-addon">.00</span>
</div>
```

Con lo que obtendríamos el siguiente resultado:

Navegación

Los elementos de navegación de Bootstrap comparten la etiqueta `.nav` para su marcado en la clase contenedora. Estos elementos necesitan la librería JavaScript para su correcto funcionamiento. Algunos de los elementos de navegación que podemos utilizar son las fichas o pestañas y las "píldoras".

Fichas o pestañas

Mediante la clase `.nav-tabs` podemos crear un grupo de pestañas o fichas, para ello tenemos que seguir la siguiente estructura:

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Es importante que nos fijemos en cómo se usan las clases CSS `.nav`, `.nav-tabs`, `.nav-item` y `.nav-link`. Cada elemento del menú será un `.nav-item`, los cuales contienen un enlace tipo `.nav-link` a la sección a mostrar. Para marcar el elemento del menú que está activo o seleccionado se utiliza la clase `.active`. Además disponemos de la clase `.disabled` para deshabilitar elementos del menú.

Si visualizamos el código de ejemplo anterior obtendríamos un menú en forma de pestañas como el siguiente:

Píldoras

La clase `.nav-pills` se define de igual forma que la `.nav-tab` pero sus elementos adoptarán una apariencia más similar a botones o "píldoras":

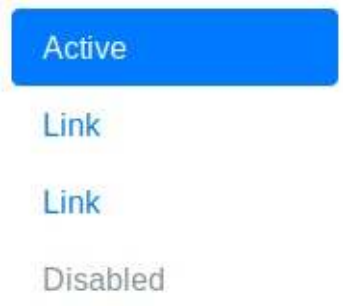
```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

En este caso el aspecto del menú sería el siguiente:

También podemos crear un menú vertical o apilado añadiendo la clase `.flex-column` a la etiqueta contenedora:

```
<ul class="nav nav-pills flex-column">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Obteniendo:



Ancho justificado

También podemos indicar que el ancho de las pestañas o de las píldoras se distribuya equitativamente según el ancho disponible. Para esto simplemente tenemos que aplicar la clase `.nav-fill` a la etiqueta contenedora, de la forma:

```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Con lo que obtendríamos:

> Este estilo no funcionará para pantallas con un ancho menor a 768px, que son las pantallas definidas como extra pequeñas o de *smartphone*. Para estos tamaños cada elemento del menú ocupará el ancho justo que necesite.

Elementos de navegación con desplegables

También podemos añadir elementos desplegables a nuestros menús de navegación, tanto al de tipo tabs como al de píldoras. Para esto simplemente añadiremos el dropdown como un elemento del menú más, usando la notación que vimos en la sección "Desplegables", pero llevan cuidado de que para la etiqueta incicial (que en el dropdown normal era " `<div`

`class="dropdown">` ") se utilice el propio elemento " `.nav-item` " del menú, añadiendo la clase " `.dropdown` " de la forma: " `<li class="nav-item dropdown">` ". A continuación se incluye un ejemplo completo:

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button"
       aria-haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Con lo que obtendríamos un resultado como el de la siguiente figura:

Barra de navegación

Bootstrap nos facilita la creación de la barra principal de navegación de nuestra web mediante la clase `.navbar`. Esta barra se adaptará al tamaño de pantalla, mostrando los elementos colapsados en un botón en pantallas pequeñas y de forma normal para pantallas más grandes.

Para añadir esta barra a nuestro sitio web utilizaremos la etiqueta "`<nav>`", que es la etiqueta de HTML 5 que identifica un elemento de navegación. En caso de no usar esta etiqueta también podemos crear la barra de navegación usando un "`<div>`", pero en este caso tendremos que añadir el atributo `role="navigation"` por cuestiones de accesibilidad. Además, en esta etiqueta también añadiremos dos etiquetas para indicar el estilo y los colores a aplicar con "`.navbar-light .bg-light`" (más adelante veremos qué otros colores podemos usar), y la etiqueta `.navbar-expand-lg` para indicar el tamaño a partir del cual la barra se mostrará de forma expandida. La etiqueta `.navbar-expand-lg` indica que la barra se mostrará en su tamaño completo a partir del tamaño de pantalla grande (lg), colapsándose para tamaños más pequeños. Este sería el comportamiento por defecto, pero si queremos lo podemos modificar cambiando el tamaño "lg" por otro de los posibles tamaños definidos por Bootstrap: "`sm`", "`md`", "`lg`" o "`xl`".

Dentro de la etiqueta "`<nav>`" el contenido de la barra estará dividido en tres secciones:

- Nombre o logotipo de la web, marcado con la etiqueta "`.navbar-brand`".
- Botón toggler marcado con "`.navbar-toggler`", que se mostrará únicamente cuando el menú se colapse y se ocultará cuando el menú aparezca expandido. Cuando sea visible podremos pulsar sobre él para mostrar u ocultar el menú.
- Las opciones de menú, las cuales las añadiremos dentro de una lista tipo "`<lu>`" con la clase "`.navbar-nav`". Además, esta lista la tendremos que meter dentro de una caja "`<div>`" con las clases "`.collapse .navbar-collapse`", que definirá la zona que se colapsará (u ocultará) para pantallas pequeñas.
    - Cada elemento de la lista de menú `<lu>` se definirá mediante una etiqueta "`<li>`" sobre la que aplicaremos la clase "`.nav-item`". Además, como ya veremos más adelante, podremos añadir otros elementos dentro de las opciones de menú, como por ejemplo un formulario.

A continuación se incluye un ejemplo completo de una barra de navegación:

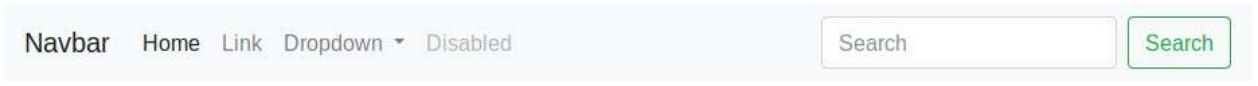
```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
          data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

          aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"

           data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label
="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button
>
    </form>
  </div>
</nav>
```

Si añadimos este código a nuestro sitio Web y lo visualizamos con un navegador, obtendremos el siguiente resultado cuando lo visualicemos en pantallas medianas y grandes:



En las pantallas pequeñas los elementos de navegación se colapsarían en un botón (*toggler*), de la forma:

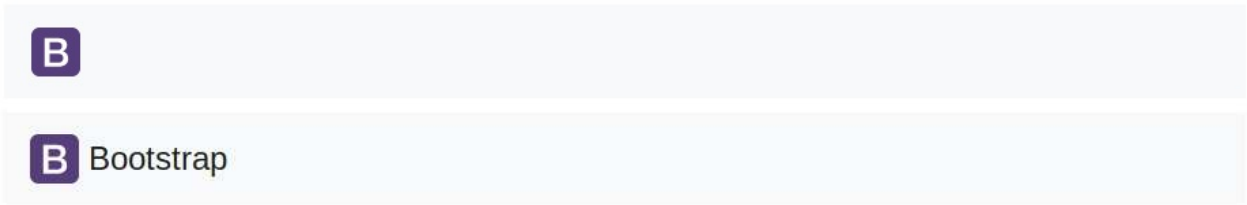
Imagen en la barra de navegación

Para incluir el logotipo de nuestra web en la barra de navegación tenemos que modificar la sección `navbar-brand` del ejemplo anterior para incluir la etiqueta `<img>`, de la forma:

```
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    <img src="/assets/brand/bootstrap-solid.svg" width="30" height="30" alt="">
  </a>
  ...
</nav>

<!-- O si queremos incluir un logotipo y texto... -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    <img src="/assets/brand/bootstrap-solid.svg" width="30" height="30" class="d-inlin
e-block align-top" alt="">
    Bootstrap
  </a>
</nav>
```

Con lo que obtendríamos los siguiente resultados, en el primer caso se mostraría el logotipo



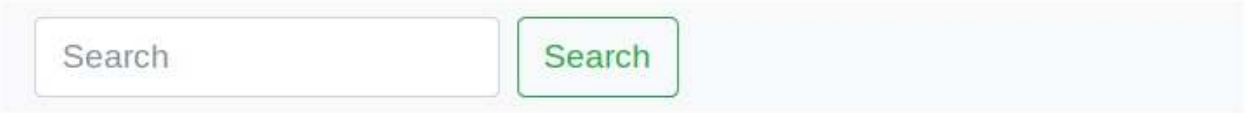
> Es posible que sea necesario añadir o modificar los estilos para disponer correctamente la imagen en la barra de navegación.

Barra de navegación con formulario

Podemos añadir formularios a nuestra barra de navegación utilizando el tipo de formulario inline, definido con " `.form-inline` " como vimos en la sección "Formularios", por ejemplo:

```
<nav class="navbar navbar-light bg-light">
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label=
"Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Obteniendo:

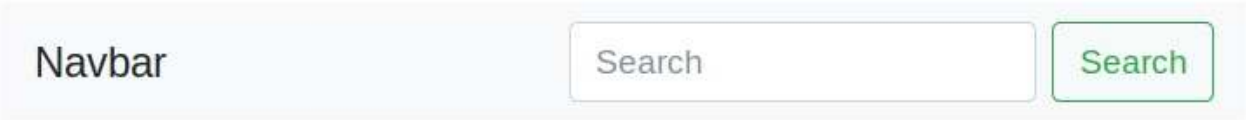


Alineación

Para modificar la alineación del formulario podemos utilizar las clases que vimos en la sección "Alineación horizontal" dentro del "Sistema de rejilla", como por ejemplo " `.justify-content-between` ":

```
<nav class="navbar navbar-light bg-light justify-content-between">
  <a class="navbar-brand">Navbar</a>
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label=
"Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Que produciría que el formulario se alenee a la derecha:



Anclajes de la barra de navegación

Bootstrap nos permite anclar o fijar la posición de la barra de tres formas distintas: fijarla a la parte superior añadiendo la clase `.fixed-top` a la etiqueta nav, fijarla a la parte inferior con `.fixed-bottom` , o usar el modo sticky (o pegajoso) con la etiqueta `.sticky-top` , el cual anclará la barra a la parte superior mientras se realiza scroll y cuando se alcanza el tope permanecerá fija. A continuación se incluye un ejemplo de cada uno de estos modos:

```
<!-- Fixed top -->
<nav class="navbar fixed-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed top</a>
</nav>

<!-- Fixed bottom -->
<nav class="navbar fixed-bottom navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed bottom</a>
</nav>

<!-- Sticky top -->
<nav class="navbar sticky-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Sticky top</a>
</nav>
```

En los modos " `.fixed-top` " y " `.fixed-bottom` ", dado que la barra se colocará de forma "flotante" sobre el contenido, es posible que oculte una parte del mismo. Para solucionar esto es necesario añadir un pequeño espaciado superior o inferior a la etiqueta `<body>`. El alto de la barra es de 50px, por lo que se suele recomendar un espaciado de 70px, de la forma:

```
body { padding-top: 70px; }  /* En el caso de .fixed-top */
body { padding-bottom: 70px; }  /* En el caso de .fixed-bottom */
```

Contenedores

Aunque no es completamente necesario, sí que se recomienda añadir un elemento contenedor a la barra. El cual podrá añadirse de dos formas: un contenedor externo que incluya toda la barra, o un contenedor interno que incluya solo los elementos de la barra. A continuación se incluyen ambos ejemplos:

```
<!-- Contenedor externo -->
<div class="container">
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
  </nav>
</div>

<!-- Contenedor interno -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>
```

El resultado obtenido solo difiere para los tamaños de pantalla grandes, en los cuales, en el primer caso la barra aparecerá centrada (y el color de la barra solo se aplicará en el espacio central), y en el segundo caso la barra ocupará todo el ancho posible (por lo que el color de aplicará en todo el ancho) pero los elementos de la barra aparecerán centrados. A continuación se incluye un ejemplo de los dos casos:

Colores de la barra de navegación

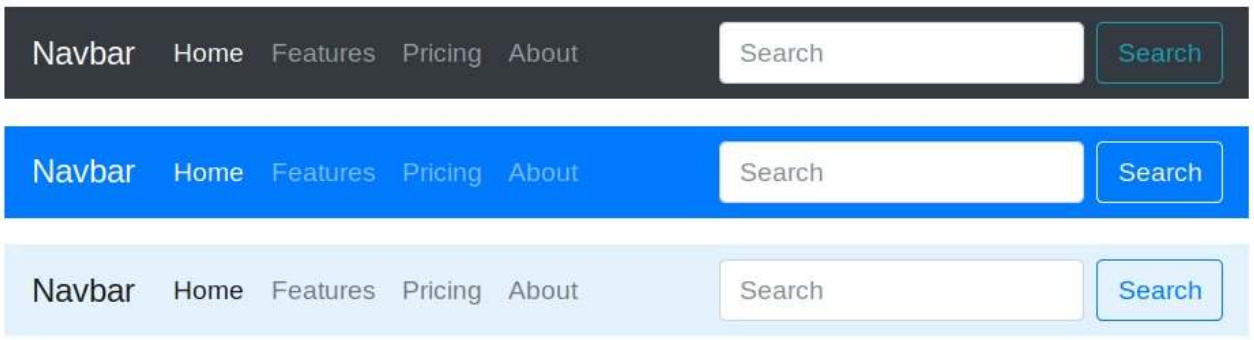
Podemos personalizar el color de la barra y los elementos que la componen de una forma muy sencilla. En primer lugar tendremos que elegir entre el tema claro (`.navbar-light`) o el tema oscuro (`.navbar-dark`), y además asignar un color de fondo con las clases `.bg-*` para personalizar el color (a continuación se incluye la lista de colores posibles). Por ejemplo podríamos modificar la barra de navegación de las siguientes formas:

```
<nav class="navbar navbar-dark bg-dark">
  <!-- ... -->
</nav>

<nav class="navbar navbar-dark bg-primary">
  <!-- ... -->
</nav>

<nav class="navbar navbar-light" style="background-color: #e3f2fd;">
  <!-- ... -->
</nav>
```

Con lo que obtendríamos los siguientes resultados:



Los posibles colores que podemos elegir como fondo para la barra de navegación son los siguientes: `.bg-primary` , `.bg-secondary` , `.bg-success` , `.bg-danger` , `.bg-warning` , `.bg-info` , `.bg-light` , `.bg-dark` y `.bg-white` . Además de poder aplicarlo sobre la barra de

navegación también se pueden utilizar para definir el color de fondo de cualquier otro elemento. A continuación se incluye una imagen de estos colores:

Tablas

Bootstrap también define una serie de clases para aplicar estilos sobre las tablas de HTML. La más básica es la clase `.table` :

```
<table class="table">
  ...
</table>
```

La cual configura los estilos de las tablas básicas de HTML para que adopten el siguiente aspecto:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

En la tabla anterior las celdas de la primera fila estarían marcadas con "th" y el resto de celdas con "td".

Tablas pequeñas

Si queremos compactar el tamaño de la tabla para que deje un padding (o espaciado interior) inferior, podemos aplicar la clase `.table-sm` de la forma:

```
<table class="table table-sm">
    ...
</table>
```

Obteniendo:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

Colores alternos

Si además aplicamos la clase `.table-striped` a nuestra tabla conseguiremos que las filas presenten colores alternos:

```
<table class="table table-striped">
  ...
</table>
```

Con lo que obtendríamos una tabla con el siguiente aspecto:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

Tablas con bordes

También podemos dibujar un borde al rededor de la tabla añadiendo la clase `.table-bordered`, de la forma:

```
<table class="table table-bordered">
  ...
</table>
```

Obteniendo el siguiente resultado:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Mark	Otto	@TwBootstrap
3	Jacob	Thornton	@fat
4	Larry the Bird		@twitter

Tablas *Responsive*

Bootstrap proporciona una forma de crear tablas *responsive* que se basa en crear un *scroll* horizontal para que se vean correctamente. Para que esto funcione simplemente tenemos que añadir la etiqueta `.table-responsive` a la propia tabla:

```
<table class="table table-responsive">
  ...
</table>
```

Obteniendo:

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading	Tabl hea
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell	Tabl cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell	Tabl cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell	Tabl cell

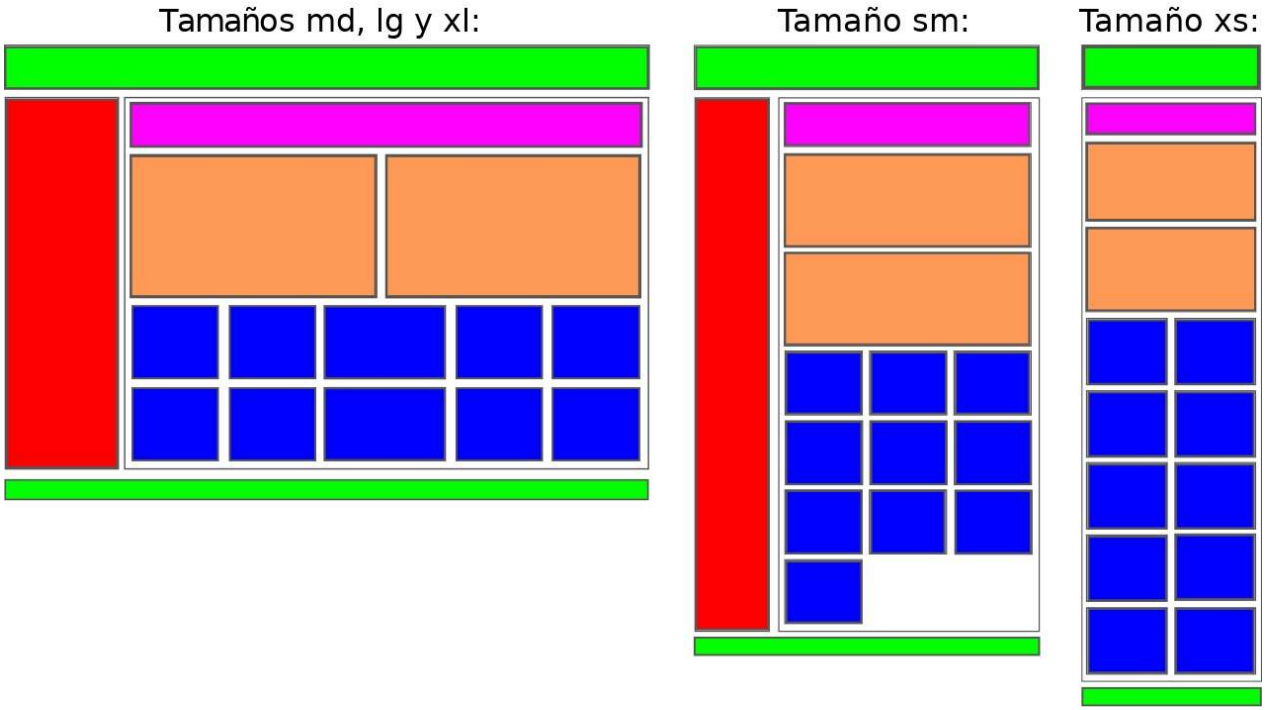
Este efecto se aplicará únicamente sobre dispositivos pequeños (`<576px`) mientras que en el resto de dispositivos no se notará la diferencia. Si queremos que el punto de ruptura a partir del cual se aplique el responsive sobre la tabla sea un tamaño mayor podemos indicar

un sufijo de tamaño sobre esta etiqueta, de la forma `.table-responsive-*`, donde "`*`" podrá ser `sm`, `md`, `lg` o `xl`.

Ejercicios 1

Ejercicio 1 - Diseño *responsive* (1 punto)

En este ejercicio vamos a practicar con la librería Bootstrap y su sistema de rejilla. Partiremos de la plantilla para una página web básica facilitada en la teoría, le añadiremos un contenedor de tipo `container` e iremos añadiendo filas y columnas intentando imitar el diseño (y colores) del esquema de la siguiente figura:



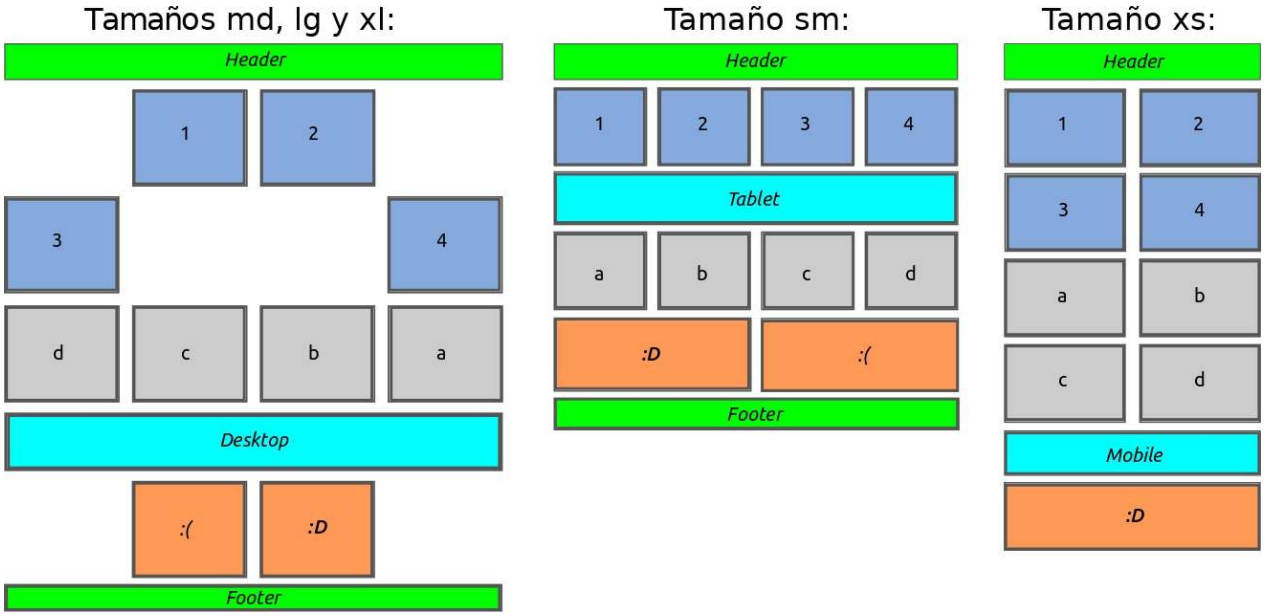
En el esquema de la figura se pueden ver tres disposiciones de la misma web, la de la izquierda se refiere a los tamaños grandes (xl y lg) y medianos (md), la disposición central al tamaño pequeño o de *tablets* (sm) y la de la derecha la correspondiente a móviles (xs).

Tenéis que aplicar las clases de Bootstrap necesarias para que al cambiar el tamaño de la pantalla se cambie la disposición de los bloques como se muestra en el esquema. Tened en cuenta que la columna roja tendrá que desaparecer cuando el tamaño sea extra pequeño (xs).

Ejercicio 2 - *Offset* y ordenación (1 punto)

En este ejercicio vamos a practicar con algunas características más de Bootstrap: la posibilidad de añadir un *offset* (o espacio inicial a las columnas), el cambio de orden de los elementos de una fila y la visibilidad de las columnas según el tamaño del dispositivo.

Para ello nos crearemos una nueva página web partiendo de la plantilla básica, le añadiremos un contendor de tipo `container` e iremos añadiendo filas y columnas intentando imitar el diseño, colores y contenidos del esquema de la siguiente figura:



Tened en cuenta que:

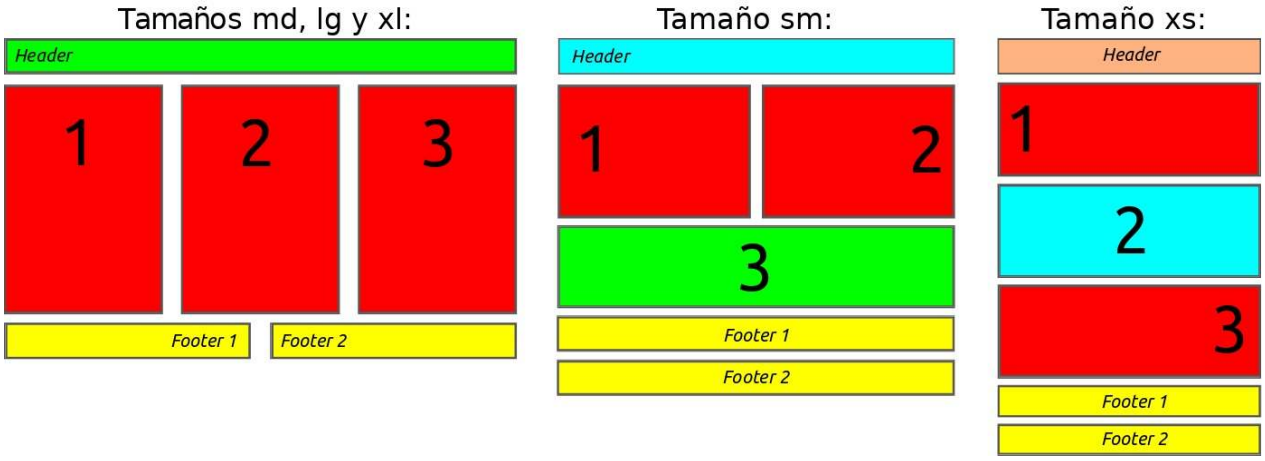
- La segunda fila (que contiene 4 columnas con los números 1, 2, 3 y 4) es solamente una fila a la que se le han añadido *offsets*. Para forzar el cambio de fila se puede añadir un elemento entre la 2ª y la 3ª columna que solo sea visible cuando la pantalla sea mediana o grande (md, lg o xl) y que aplique la clase `.w-100` de Bootstrap.
- El orden de la tercera fila (con las letras a, b, c, d) se ha alterado para las disposiciones de pantalla grandes (md, lg y xl) usando las clases de bootstrap `order-*`.
- En la 5ª fila naranja se ha aplicado un cambio de orden y un offset para las pantallas grandes y medianas (md, lg y xl). Además, cuando la pantalla sea de tipo *xs* se deberá de ocultar una de sus columnas.
- La fila azul claro en la que pone *Desktop* (para pantallas md, lg y xl), *Tablet* (para sm) y *Mobile* (cuando la pantalla es xs) en realidad son 3 filas distintas con clases para que solo se muestren en dichos tamaños de pantalla.
- La última fila (*Footer*) se deberá de ocultar solamente cuando la pantalla sea del tipo *xs*.

Ejercicio 3 - Personalizando mediante *media query* (1 punto)

En este ejercicio se pide que creéis una nueva página web usando la librería Bootstrap. El contenido aparecerá centrado en la pantalla y constará de tres filas con el siguiente contenido y disposición, cuando la pantalla sea de tamaño medio (md) y grande (lg y xl):

- Una fila en la parte superior con una única columna con fondo verde que ocupará todo el ancho, en dicha columna aparecerá el texto "*Header*" alineado a la izquierda y en grande.
- Una segunda fila con tres columnas en color rojo con el mismo ancho y con los números 1, 2 y 3 (respectivamente) centrados y en letras grandes.
- La tercera y última fila contendrá dos columnas de igual ancho y en color amarillo, la primera columna tendrá el texto "*footer 1*" alineado a la derecha y la segunda el texto "*footer 2*" alineado a la izquierda (ambos usando un tamaño de fuente grande).

En la siguiente imagen se puede ver un esquema de la web a realizar:



Como se puede ver en el esquema de la imagen, la disposición de las columnas y la alineación de los textos variará dependiendo del tamaño de la pantalla. Tenéis que reproducir este comportamiento para que la apariencia de la web sea similar al esquema (número de columnas, alineaciones de los textos y colores) cuando el tamaño de la pantalla sea la de un *tablet* (sm) o la de un teléfono (xs).

Tened en cuenta que:

- Siempre que sea posible se utilizarán las clases que provee Bootstrap.
- Cuando no sea posible (por ejemplo para controlar la alineación de los textos y el cambio de color del fondo) tendréis que definir una *media query* que lo haga.

Ejercicios 2

Ejercicio 1 - Crear una Web responsive (3 puntos)

Para poner en práctica los conceptos teóricos sobre diseño *responsive,* se propone como ejercicio la creación de un pequeño sitio Web estático que use los estilos y componentes de Bootstrap.

La temática, contenidos y estilos del sitio son libres, pero deberá tener al menos las siguientes características:

- El sitio estará formado por al menos 3 páginas enlazadas entre sí (con contenidos estáticos).
- Ser completamente *responsive*, de forma que se adapte tanto a pantallas extra pequeñas de *smartphone* como a *tablets* y pantallas más grandes de portátiles y de escritorio.
- Tener una barra de navegación principal que se contraiga cuando la pantalla sea pequeña. Esta barra tendrá al menos:
  - Dos enlaces.
  - Una imagen como logotipo.
  - Un buscador (aunque no sea funcional).
- Contener los siguientes elementos (un ejemplo de cada uno en alguna de las páginas del sitio web):
  - Botones.
  - Un desplegable.
  - Una sección con fichas o pestañas.
  - Un formulario horizontal.
  - Una tabla responsive con bordes y de tipo *striped.*
- El estilo base a utilizar será el que define Bootstrap, si se definen estilos CSS personalizados tendrán que estar en un fichero separado, llamado "custom.css", y que será común para todas las páginas del sitio.

Un posible ejemplo de una web que podéis realizar sería, por ejemplo, una web de recetas. Esta podría tener una página principal con la información más importante, una página con una receta de ejemplo (aquí se podrían utilizar las fichas o pestañas para cambiar entre

elaboración e ingredientes, los cuales podrían estar en una tabla) y otra página para el envío de recetas (con un formulario horizontal, botones para envíar y cancelar, y un desplegable para elegir la categoría).

De forma similar se podría crear la web sobre coches u otro tipo de vehículos, mascotas, bicicletas, etc.

> Al ser una web estática tendréis que repetir partes del código en todas las páginas, por ejemplo la barra de menú principal tendrá que ser igual en todas las páginas. Por este motivo se recomienda realizar primero estas partes, y una vez probadas, copiar y pegar el código en el resto de páginas.

Bibliografía

- http://getbootstrap.com/

  Página oficial de Bootstrap desde donde descargar la librería y consultar toda la documentación.

- http://blog.getbootstrap.com/

  El blog oficial de Bootstrap donde se publican las últimas novedades.

- Temas y plantillas gratuitas para Bootstrap:
  - http://startbootstrap.com/
  - http://bootstrapzero.com/
  - http://bootswatch.com/
  - http://www.bootbundle.com/
- http://bootsnipp.com

  Ejemplos y trozos de código útiles para Bootstrap. Aquí podrás encontrar cientos de ejemplos, desde como hacer un formulario de login hasta todo tipo de elementos con animaciones o estilos avanzados.

- http://expo.getbootstrap.com/

  Ejemplos *inspiradores* de uso de Bootstrap.

- http://startbootstrap.com/bootstrap-resources/

  Listado completísimo con todo tipo de recursos disponibles para Bootstrap.