Bootstrap 5

Miguel Ortuño
Escuela de Ingeniería de Fuenlabrada
Universidad Rey Juan Carlos

Febrero de 2025

¿Qué es Bootstrap?

- Bootstrap es un framework libre para desarrollo web
- Desarrollado inicialmente en 2011 por ingenieros de Twitter
- La versión actual, Bootstrap 5, aparece en mayo de 2021. A diferencia de las anteriores, emplea *vanilla JavaScript*, no *jQuery*
- Incluye plantillas HTML y CSS con tipografías, formas, botones, cuadros, tablas, barras de navegación, carruseles de imágenes y muchas otras
- Aunque su preferencia es *mobile first*, permite crear diseños que se ven bien en múltiples dispositivos (*responsive design*)
- Orientado a programadores, no a diseñadores gráficos
- Es posiblemente la herramienta más popular para este fin, aunque hay alternativas como Foundation

Características de Bootstrap

Ventajas

- Resulta sencillo y rápido escribir páginas con muy buen aspecto
- Se adapta a distintos dispositivos (*responsive design*)
- Proporciona un diseño consistente
- Es compatible con los navegadores modernos
- Es software libre

Inconvenientes

- Al ser una herramienta muy popular, las páginas web que no estén personalizadas *quedan iguales que las de todo el mundo*
- No es especialmente fácil personalizar los estilos (Foundation puede ser más adecuado para esto)

Holamundo en Bootstrap

Para usar Bootstrap basta con

- Definir el *viewport*
- Incluir un elemento *link* apuntando al CSS de Bootstrap
- Incluir un elemento *script* apuntando al código JavaScript de Bootstrap

```html
<!doctype html>
<html lang="es-ES">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
   integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
   rel="stylesheet" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
   integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
   crossorigin="anonymous">
  </script>

  <title>Holamundo en Bootstrap 5</title>
</head>

<body>
  <div class="container">
    <h1>Holamundo en Bootstrap 5</h1>
  </div>
</body>

</html>
```

http://ortuno.es/hola_bootstrap5.html

Adaptación del contenido a la pantalla

Ya desde su diseño original, un requisito importante para el web es que las páginas se pueda representar en pantallas de cualquier tamaño. Con la aparición de los *smartphones*, esto es aún más necesario y más complicado. A lo largo de los años se han usado varias técnicas para conseguir esto, cada vez mejores

1. Técnica inicial
   Viewport. Barras de desplazamiento horizontal y vertical, recomposición de los elementos sobre el *viewport*
2. Primeros *smartphones*
   Viewport virtual
3. Teléfonos móviles actuales
   Diseño *responsive* basado en *grid*

Viewport

Para diseñar webs en dispositivos móviles, es importante tener claro qué es el *viewport* y cómo se comporta

- *Viewport* es la zona visible de una página web. En los navegadores tradicionales de escritorio, coincide con la ventana del navegador
- Supongamos una página web grande y compleja, como la portada de un periódico. La página no cabrá en la ventana del navegador, el usuario usará las barras de scroll para mover el *viewport* sobre el documento. Al redimensionar la ventana, cambiará el tamaño del *viewport*
- Cambiar el tamaño del viewport reposiciona el texto y todos los elementos: las líneas se truncan, las imáges se recolocan, etc

Viewport es un rectángulo donde se compone un fragmento (tal vez completo) de la página web para presentarla al usuario

Viewport virtual

Con la aparición de los navegadores en teléfonos móviles, los cambios del tamaño de la pantalla son mucho más drásticos. La técnicas tradicionales siguen funcionando, pero proporcionan una experiencia de uso muy poco satisfactoria

- El área visible de un móvil es demasiado pequeña, componer una página web tradicional en ese *viewport* queda mal. Observa lo que sucede en esta página antigua cuando la ventana es muy grande o muy pequeña
  `https://tinyurl.com/y7e77lvw`
- Además, en un navegador para móvil no hay barras de scroll, ocuparían un espacio demasiado valioso. Ni ventanas, serían demasiado pequeñas

Para solucionar este problema, aparece el concepto del *viewport virtual*, mayor que el *viewport físico*, esto es, mayor que lo que se puede representar en la pantalla real del dispositivo

- Lo introduce Apple para Safari en iOS, luego pasa a ser estándar
- El ancho del *viewport* virtual es razonablemente grande, por ejemplo 980 pixeles en el navegador safari para iPhone
- El navegador compone la página sobre este viewport virtual, ya no hacen falta barras desplazamiento horizontal
- El usuario arrastra el *viewport físico* (la pantalla real, más pequeña) sobre el viewport virtual, para que le muestre una zona u otra del documento. También se le puede permitir hacer zoom
  - Redimensionar este viewport físico ya no provoca la recomposición de la página

Páginas responsive

Una página web moderna con un minimo de calidad se entiende que tiene que ser *responsive*

- La página se adapta al tamaño de la pantalla (escritorio, tablet, móvil), sin usar la barra de desplazamiento horizontal, que es muy incómoda. La barra de desplazamiento vertical se sigue usando, no es molesta
- El diseño *responsive* tal y como lo conocemos en la actualidad se basa en el uso de un *grid*. En español se traduce por *cuadrícula*, *rejilla* o *casilla* [1]. Aquí veremos el *grid* de Bootstrap, hay otros pero siempre son similares
- En estas páginas ya no hace falta un *viewport* virtual, porque la página está diseñada para adaptarse al *viewport* ordinario (la pantalla pequeña)

[1] en ocasiones le llamaremos celda por analogía con las hojas de cálculo

Las mismas 12 casillas se presenta de forma distinta en un ordenador

XXXXXXXXXXXX

En un tablet

XXXXXX
XXXXXX

En un móvil

XXXX
XXXX
XXXX

Mobile first

- Con una propiedad de etiqueta meta, podemos indicar la escala inicial del *viewport*
- Como las páginas con bootstrap son *responsive*, especificamos que el *viewport* virtual coincida con el ancho de la pantalla, esto es, con el *viewport* ordinario. En otras palabras: que no haya un *viewport* virtual

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

- También se puede inhabilitar el zoom en dispositivos móviles con `user-scalable=no`
- Los usuarios sólo podrán hacer *scroll* y tendrá una apariencia nativa.

```
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no">
```

Contenedores

- Para que sean *responsivos*,todos los elementos de Bootstrap deben estar dentro de un elemento contenedor
- Los contenedores no se pueden anidar
- Asegúrate de cerrar correctamente cada contenedor. Si alguna fila queda fuera, sus columnas quedarán mal alineadas. Y este error no lo detecta el W3C validator.

Para un contenedor responsivo de tamaño fijo, se usa `.container`

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

http://ortuno.es/container.html

Si se desea un contenedor con el ancho total (del *viewport*), se ha de usar `.container-fluid`

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

http://ortuno.es/container_fluid.html

El sistema de cuadrículas de Bootstrap

- La pantalla se divide en filas y columnas
- Llamaremos *celda* a la intersección entre una fila y una columna [2]
- El contenido se coloca dentro de una celda, y siempre se mostrará dentro de esa misma celda
- El ancho de cada celda se mide en casillas
- En cada fila hay hasta 12 casillas, que el diseñador decide cómo repartir entre celdas
- Cuando la pantalla tiene la suficiente resolución, las celdas de la misma fila se ven unas al lado de otras (*disposición normal*)
- Cuando la resolución disminuye, las celdas que originalmente estaban en la misma fila, pasan a verse unas encima de otras (*disposición apilada*)

[2] en realidad Bootstrap no usa este concepto, habla solo de columna, pero entendemos que es una terminología confusa: un ladrillo individual no puede ser una columna, hacen falta varios ladrillos apilados

Una fila puede estar formada por cualquiera de estas combinaciones

- 2 celdas de 6 casillas
- 3 celdas de 4 casillas
- 4 celdas de 3 casillas
- 1 celda de 2 casillas, 1 celda de 8, 1 celda de 2
- 1 celda de 6, 2 celdas de 3
- etc

Lo habitual es consumir las 12 casillas, pero tampoco es obligatorio. P.e. podríamos tener 2 celdas de 4 casillas.

- Cada fila es un elemento *div* de HTML con la clase *row*. Observa que empleando notación de los selectores de CSS (donde el punto significa *clase*), podemos llamarle `.row`
- Dentro de la fila hay elementos a los que en esta asignatura llamamos *celdas*, que pueden ser de los tipos `.col-N`, `.col-sm-N`, `.col-md-N`, `.col-lg-N`, `.col-xl-N` o `.col-xxl-N`
- Estos 6 tipos de celdas dependen del ancho de *viewport* (pantalla) en el que queramos que las celdas se muestren en disposición normal, no apilada

Ejemplo:

```
<div class="row">
  <div class="col-md-4"> Bla, bla </div>
  <div class="col-md-8"> Bla, bla, blá </div>
</div>
```

- .col-N
  Pantallas muy pequeñas, de menos de 576px
- .col-sm-N
  Pantallas pequeñas de al menos 576px
- .col-md-N
  Pantallas medianas de al menos 768px
- .col-lg-N
  Pantallas grandes de al menos 992px
- .col-xl-N
  Pantallas muy grandes de al menos 1200px
- .col-xxl-N
  Pantallas extra grandes de al menos 1400px

Donde N es un número entre 1 y 12, que indica el ancho de cada columna. El total de las columnas de cada fila puede sumar un máximo de 12. La frontera entre cada uno de estos tamaños se denomina *breackpoint*

- Columnas *.col-xxl-N*
  Disposición normal en pantallas *extra grandes*.
  Se apilan en pantallas muy grandes, grandes, medianas, pequeñas o muy pequeñas
- Columnas *.col-xl-N*
  Disposición normal en pantallas *muy grandes* o *extra grandes*.
  Se apilan en pantallas grandes, medianas, pequeñas o muy pequeñas
- Columnas *.col-lg-N*
  Disposición normal en pantallas *grandes*, *muy grandes* o *extra grandes*.
  Se apilan en medianas, pequeñas o muy pequeñas

- Columnas *.col-md-N*
  Disposición normal en pantallas *medianas*, *grandes*, *muy grandes* o *extra grandes*.
  Se apilan en: pequeñas o muy pequeñas
- Columnas *.col-sm-N*
  Disposición normal en pantallas *pequeñas*, *medianas*, *grandes*, *muy grandes* o *extra grandes*.
  Se apilan en muy pequeñas
- Columnas *.col-N*
  Disposición normal en cualquier pantalla: *muy pequeñas*, *pequeñas*, *medianas*, *grandes*, *muy grandes* o *extra grandes*.
  Nunca se apilan

Dicho de otro modo

- Cada tipo de columna se muestra en su disposición normal, esto es, horizontalmente, si la pantalla es de su tipo o de un tipo mejor
- En otro caso, las casillas se apilan verticalmente

Esto parece un poco complicado, pero con el siguiente ejemplo verás que no:

1. Vete a `http://ortuno.es/rejilla_01.html`
2. Maximiza la ventana
3. Vete reduciéndo el ancho gradualmente. Esto es equivalente a tener una pantalla menor. Verás que según vayas reduciendo, las cuadrículas que originalmente están en disposición normal (horizontal) se van apilando (verticalmente)

Alineación horizontal de las cuadrículas

Las celdas (que formarán columnas cuando sean varias a la misma distancia del eje vertical) se pueden alinear de diversa forma en horizontal añadiendo a la fila (el *div* de clase *row*) las clases

```
justify-content-start
justify-content-center
justify-content-end
justify-content-around
justify-content-between
justify-content-evenly
```

En el código fuente de este ejemplo podrás ver

- El resultado de usar las distintas clases de alineamiento horizontal. En este caso con dos columnas de 3 casillas cada una
- El uso de la clase *border* con el color *border-primary*

`http://ortuno.es/rejilla_02.html`

Componentes de Bootstrap

Bootstrap viene con una serie de estilos (generalmente en formato de clase CSS) y componentes en JavaScript.

- `btn`
- `table`
- `card`
- `carousel`
- y otras utilidades responsivas

Colores contextuales

La gama concreta de colores se decidirá en el CSS. Aquí pondremos clases con valor semántico.

- Con alguna excepción como *light* o *white*, puesto que al elegir el color del fondo, puede ser necesario indicar también el color del texto (en este ejemplo, el texto blanco sobre fondo blanco no se ve)

```
<h2>Colores del texto</h2>
<p class="text-muted">Muted (silenciado, apagado).</p>
<p class="text-primary">Primary.</p>
<p class="text-success">Success (éxito).</p>
<p class="text-info">Info.</p>
<p class="text-warning">Warning.</p>
<p class="text-danger">Danger.</p>
<p class="text-secondary">Secondary.</p>
<p class="text-body">Body (típicamente negro).</p>
<p class="text-light">Light grey .</p>
<p class="text-white">White.</p>
```

```
<h2>Colores del fondo</h2>
<p class="bg-primary text-white">Primary.</p>
<p class="bg-success text-white">Sucess (éxito)</p>
<p class="bg-info text-white">Info.</p>
<p class="bg-warning text-white">Warning.</p>
<p class="bg-danger text-white">Danger.</p>
<p class="bg-secondary text-white">Secondary.</p>
<p class="bg-dark text-white">Dark (grey).</p>
<p class="bg-light text-dark">Light (grey).</p>
```

http://ortuno.es/colores.html

Botones

La clase `btn` de Bootstrap puede añadirse a los elementos HTML `<button>`, `<input>` y `<a>`

- Tienen efecto *hover*: destacan un botón cuando se posiciona el ratón encima

```
<button type="button" class="btn">Basic</button>
<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-info">info</button>
<button type="button" class="btn btn-warning">warning</button>
<button type="button" class="btn btn-danger">danger</button>
<button type="button" class="btn btn-dark">dark</button>
<button type="button" class="btn btn-light">light</button>
<button type="button" class="btn btn-link">link</button>
```

```
<button type="button" class="btn btn-outline-primary">btn-outline-primary</button>
<button type="button" class="btn btn-outline-secondary">btn-outline-secondary</button>
<button type="button" class="btn btn-outline-success">btn-outline-success</button>
<button type="button" class="btn btn-outline-info">btn-outline-info</button>
<button type="button" class="btn btn-outline-warning">btn-outline-warning</button>
<button type="button" class="btn btn-outline-danger">btn-outline-danger</button>
<button type="button" class="btn btn-outline-dark">btn-outline-dark</button>
<button type="button" class="btn btn-outline-light text-dark">btn-outline-light</button>
```

Con el atributo disabled (atributo, no clase), el botón queda inhabilitado

```
<button type="button" class="btn btn-primary" disabled>
  disabled Primary
</button>
```

http://ortuno.es/botones.html

Imágenes

Para modificar el aspecto de una imagen, Bootstrap, nos permite añadir clases al elemento `<img>`
Contorno:

- rounded
  Esquinas redondeadas
- rounded-circle
  Circular
- img-thumbnail
  Miniatura (reborde blanco)

Alineación:

- float-start
  Izquierda
- float-end
  Derecha
- mx-auto d-block
  centrada
- fluid
  El uso de esta clase es especialmente interesante: no fijamos un tamaño en pixeles, sino que la imagen se adapta en cada caso para ocupar todo el espacio disponible en la celda. Si la imagen no cabe, se reduce (no se trunca)

```
<div class="row">
  rounded
  <div class="col-xl-12">
    <img src="images/plaza_espana.jpg" alt="Plaza de España,
    ↪  Madrid"
     width="400" class="rounded">
  </div>
</div>
```

http://ortuno.es/imagenes.html

Tablas

Para dar formato a un elemento <table>, Bootstrap 5 nos ofrece las clases *.table*, *.table-bordered*, *.table-hover*, *.table-dark* y *.table-striped*

```
<table class="table table-striped">
  <thead>
    <tr>
      <th>Baraja española</th>
      <th>Baraja francesa</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Caballo</td>
      <td>Reina</td>
    </tr>
    <tr>
      <td>Rey</td>
      <td>Rey</td>
    </tr>
  </tbody>
</table>
```

http://ortuno.es/tablas.html

cards

Una *tarjeta* (*card*) es una caja redondeada dividida en cabecera, cuerpo y pie.

- Es útil para agrupar otros elementos como botones, formularios, imágenes, etc
- Sucesor de los antiguos *panels* en las versiones anteriores de Bootstrap
- Se les puede poner un color contextual de fondo añadiendo las clases que ya conocemos:
  .bg-primary, .bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light

```
<div class="card" style="width:400px">
  <div class="card-header">
    <h4 class="card-title">Gato Panchi</h4>
  </div>
  <div class="card-body">
    <img src="images/gato.jpg" alt="Fotografía de un gato
     llamado Panchi" width="300" class="mx-auto d-block">
  </div>
  <div class="card-footer">
    <a href="#" class="btn btn-primary float-end">Más
    ↪  información</a>
  </div>
</div>
```

http://ortuno.es/card.html

Formularios

Bootstrap incluye clases para mejorar el aspecto y usabilidad de los formularios

- El uso de `<label>` es necesario, no es válido escribir texto HTML para identificar los elementos del formulario
- Los distintos elementos de un formulario aparecen unos debajo de otros. Si los queremos unos al lado de otros, usaremos las filas y columnas de la rejilla

- A los `<label>` les añadimos `class="form-label"`
- A los elementos de entrada de texto, `<input>` y `<textarea>` les añadimos `class="form-control"`
- A los `<checkbox>` los metemos en un `<div>` al que añadimos `class="form-check"`
- A los `<input type="radio"> <input type="checkbox">` les añadimos `class="form-check-input"`

`http://ortuno.es/form_b5.html`

carousel

El componente `carousel` muestra fotografías que se desplazas horizontalmente, como un pase de diapositivas. Se les puede añadir título o cualquier otro texto

- El elemento de mayor nivel jerárquico del carrusel es un *div* con las clases *carousel* y *slide* Tiene un atributo *id* cuyo valor será referenciado por los botones que contenga

```
<div id="carrusel01" class="carousel slide"
 data-bs-ride="carousel">
```

El `.slide .carousel` contendrá tres *divs*

- `.carousel-indicators`
  Los puntos o pequeñas líneas que representan cada una de las fotos. Un *div* de clase *carousel-indicators*
- `.carousel-inner`
  Un *div* de clase *carousel-inner* con las imágenes.
  - Cada imagen es un *carousel-item*, que contiene la imagen y un *carousel-caption*. Importante: es muy recomendable que todas las imágenes tenga la misma relacion alto/ancho
- Los botones

`http://ortuno.es/carrusel.html`

Deshabilitar elementos

Como hemos visto, muchos elementos bootstrap admiten la clase *disabled* para indicar que tengan un aspecto gráfico distinto, deshabilitado

- Pero esto no los deshabilita realmente
- Para deshabilitar por completo un elemento, usamos el atributo *disabled*

```
<button type="button" class="btn btn-lg" disabled>Botón</button>

<input type="text" name="lname" disabled><br>
```

Depuración

Si la página no tiene el aspecto que buscas:

- Asegúrate de que todos los elementos están dentro de un *container*. Normalmente solo deberías usar uno para la página
- Usa el W3C validator. Detectará p.e. elementos sin cerrar (aunque no elementos cerrados en el sitio incorrecto)
- Comprueba que la estructura de los *div* está bien, que no has cerrado ninguno demasiado pronto o demasiado tarde. Un buen editor te ayudará con esto mostrando el código por niveles. P.e atom cuenta con los atajos Ctrl k Ctrl 1, Ctrl k Ctrl 2, Ctrl k Ctrl 3, etc
- Si usas Bootstrap, no añadas directamente reglas CSS. Excepto si estás seguro de lo que haces

Enlaces relacionados

- Documentación oficial

  `https://getbootstrap.com/docs/5.1/getting-started`

- Tutorial en w3schools

  `https://www.w3schools.com/bootstrap5/index.php`