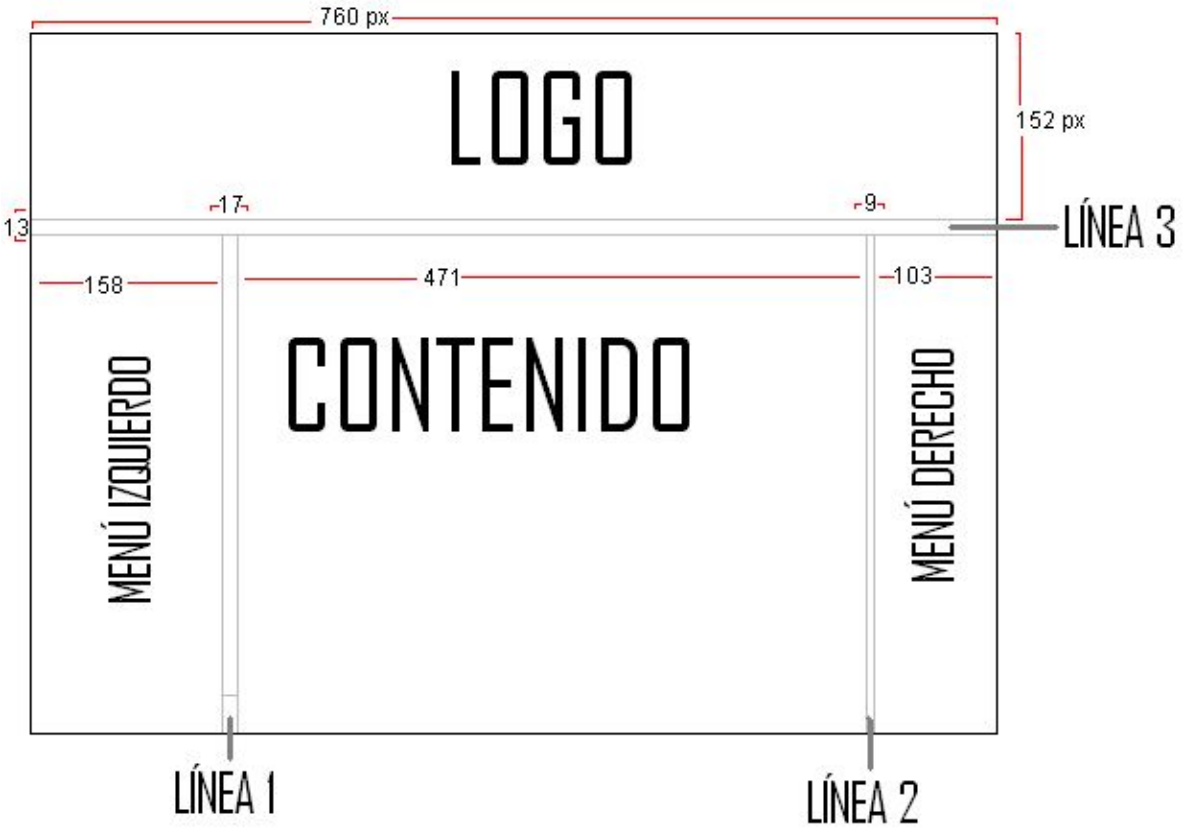
Taller de Maquetado Web

por Fernando Dodino, Emmanuel Schonfeld

Versión 1.0

Junio 2016



Indice

1 Antes de arrancar...

Deben descargar el proyecto que está en el repositorio: https://github.com/uqbar-project/eg-amigos-web/ (branch *fresh* del repo)

2 Objetivo

El presente taller tiene como objetivo introducir al lector a las tecnologías de maquetado web, en particular conoceremos

- html como lenguaje de descripción de contenidos
- css como un lenguaje para definir estilos o temas
- frameworks de front end web que facilitan las dos tareas anteriores

¿Es un apunte de "diseño web"? Ciertamente no, pero para poder desarrollar una aplicación web necesitamos comprender la tecnología de presentación que manejan los navegadores y este entendimiento nos permitirá tomar mejores decisiones de diseño posteriormente.

En el taller obtendremos nociones de maquetado web: hay diseñadores, especialistas de front-end cuyo oficio les lleva años perfeccionar (somos conscientes de ello). Pero de la misma manera en que creemos que no se puede diseñar sin conocer la tecnología de implementación, no podemos construir aplicaciones web si no conocemos las reglas básicas con la que se define el layout de una pantalla.

3 Primer caso: una lista de amigos

Tenemos una lista de amigos, del cual conocemos

- su nombre
- su apodo
- una foto
- la fecha en la que nos conocimos
    - y por lo tanto, la cantidad de años que llevamos como amigos

Queremos mostrar esa lista de amigos en un navegador web.

4 HTML

La vista en la que vamos a trabajar será HTML, que **no es un lenguaje de programación**, sino que está pensado **para mostrar contenido**, justamente lo

que vamos a hacer. Para más información te recomendamos que leas la página específica de HTML de la wiki de Uqbar.

4.1 Diseñando la vista

Aclaremos que vamos a definir una vista estática, no va a tener lógica (al menos por ahora). ¿Cuál es el layout?

- tendremos un container con los campos de búsqueda
    - podríamos tener un rango desde/hasta cantidad de años de amistad
    - y un campo nombre "comienza con"
    - debe aparecer un botón que a futuro dispare la búsqueda
    - y podría haber un botón Crear amigo
- luego puede haber una tabla o grilla con los resultados, que tenga las columnas
    - nombre, un link que permita referenciar a la vista de detalle/edición
    - apodo
    - foto (una imagen)
    - la fecha en la que nos conocimos
    - la cantidad de años que llevamos como amigos

En este paso el contenido se presenta con una **jerarquía** que debe expresarse en la estructura HTML. Para esto se usan las etiquetas específicas para este fin: h1, h2, h3, pero también strong, em (enfatizado), mark (texto resaltado), que reemplazan a los anteriores tags b (bold), i (italic) que estaban atados a la presentación, los nuevos tags expresan conceptos dentro del trabajo de edición de contenido.

Vamos a escribir los tags html que describan esto[1] en un archivo **list.html**:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">[2]
  <title>Lista de amigos</title>
</head>
<body>
  <header>
```

[1] Para más ayuda puede verse http://www.w3schools.com/html/
[2] Los encoding permiten asegurarnos de que las tildes, eñes y demás caracteres "especiales" se verán bien. Para más información recomendamos leer este artículo

```
    <h2>Lista de amigos</h2>
  </header>
  <section>
    <h3>Criterio de búsqueda</h3>
    <div>
      Cantidad de años de amistad<br>
      <input type="number" id="aniosDesde" placeholder="desde"
size="5" max="5">
      <input type="number" id="aniosHasta" placeholder="hasta"
size="5" max="5">
    </div>
    <div>
      Nombre comienza con<br>
      <input type="text" id="nombreComienzaCon" placeholder="nombre"
max="25">
    </div>
    <div>
      <input type="button" id="crearAmigo" value="Crear amigo">
      <input type="button" id="buscarAmigos" value="Buscar"
autofocus="true">
    </div>
  </section>
  <section>[3]
    <table>
      <tr>
        <th>Nombre</th>
        <th>Apodo</th>
        <th>Foto</th>
        <th>Fecha amistad</th>
        <th>Años</th>
      </tr>
      <tr>
        <td>Juan Contardo</td>
        <td>Juancho</td>
        <td><img src="images/juancho.jpg" height="50px;"
width="50px;"></td>
        <td>07/04/2011</td>
        <td>5</td>
      </tr>
```

[3] HTML 5 te pide que toda section tenga un título, pero acá no aclaramos qué contenido estamos mostrando (por ejemplo la lista de amigos... de todas maneras es útil cuando leemos una página sin utilizar una herramienta visual (por ejemplo para comunicársela a una persona no-vidente)

```
      <tr>
        <td>Clara Allende</td>
        <td>Clari</td>
        <td><img src="images/clari.jpg" height="50px;"
width="50px;"></td>
        <td>12/06/2010</td>
        <td>6</td>
      </tr>
      <tr>
        <td>Leo Cesario</td>
        <td>HH</td>
        <td><img src="images/leo.jpg" height="50px;"
width="50px;"></td>
        <td>05/11/2006</td>
        <td>10</td>
      </tr>
    </table>
    </section>
  </body>
</html>
```

Respecto del contenido:

- si se fijan en la estructura raíz del documento, tenemos dos grandes secciones: la que permite seleccionar el criterio de búsqueda y el que muestra los resultados devueltos. Esto se refleja exactamente así en HTML: son sections
- Con los divs internos del criterio de búsqueda ya no es tan necesario crear subsecciones porque semánticamente no se justifica: son separaciones visuales pero no de contenido.

4.2 Prueba en un navegador

Abrimos el archivo list.html en un navegador, y vemos cómo se interpretan los tags html en el navegador:

Nada sorprendente, pero podemos ver

- que el tag header permite definir un title que se ve en la solapa del navegador:

- el tag h2 muestra un título con los encabezados (headings) por defecto
- html permite definir widgets o controles para ingresar valores. En particular a partir de HTML5 se incorporó el input type number, que agrega dos spinners o botones para ingresar el rango de amistad en años:

- los inputs tienen placeholders, que permiten orientar al usuario el dato que está cargando 

. Más allá de eso, tenemos etiquetas o labels que es simplemente contenido dentro de un tag div para aclarar qué información se pide agregar.
- tenemos botones, en particular
    - input de tipo *submit* como acción principal de esa página (asociada a presionar el botón Enter, entonces ojo con asociar un submit al botón “Eliminar todos los amigos”)
    - button si vamos a realizar una acción que ocurrirá dentro de la página
    - link si vamos a llevar al usuario a otra página

- la tabla HTML permite definir headers (th) y datos (td) en columnas, que se agrupan en filas (tr).

Si jugamos a cambiar el tamaño del navegador, la vista no se acomoda: la tabla queda del mismo tamaño y en todo caso el browser nos muestra cuadros de desplazamiento para hacer scroll down/up.

Para mejorar la vista podríamos cambiar ciertos atributos en los tags html, pero vamos a trabajarlo en un archivo aparte, mediante un css.

5 CSS

La hoja de estilos "en cascada" (cascade style sheet) permite definir un nuevo lenguaje para asignar fuentes, formatos, espacios, alineamientos, etc. sin acoplarlo a un html específico.

Vamos a escribir el siguiente css[4] dentro del directorio *css*, en el archivo *styles.css*:

```
body {
    background-color: #d0e4fe;
    font-size: 14px[5];
}

h2 {
    color: orange;
    text-align: center;
}

p {
    font-family: "Times New Roman";
}
```

Tenemos que referenciarlo en el html dentro del header, de la siguiente manera:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>Lista de amigos</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
```

[4] Para más información ver la página específica de CSS de la materia
[5] Pueden jugar a cambiar el font-size y van a ver entonces por qué CSS es Cascading, los estilos se aplican en cascada a los elementos que forman parte de la jerarquía del documento

Recargamos la página y ...

Bueno, sigue siendo un poco rústico pero reconocemos que cambió el fondo y el título “Lista de amigos” ahora se visualiza en naranja. ¿Cómo se dio esto?

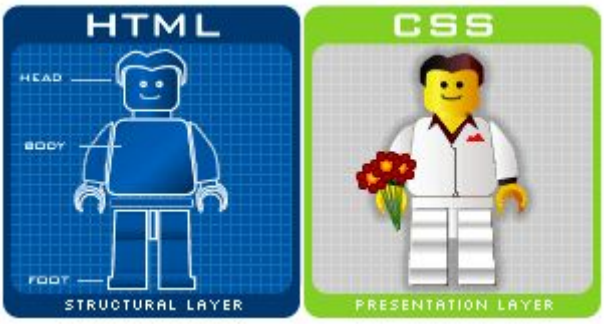
```
body {
    background-color: #d0e4fe;
}
```

Esto setea el color de fondo celeste.

```
h2 {
    color: orange;
    text-align: center;
}
```

Esto implica que cualquier tag html H2 toma los siguientes valores: color de texto naranja y alineación del texto centrado respecto a su contenedor.

En resumen, la estructura HTML se termina de completar con información de presentación que le agrega el CSS, que puede reutilizarse en varios HTML.



5.1 Mejorando el look & feel

“Look & feel” o su nuevo derivado “User experience” se refiere a la forma que tiene el usuario de navegar por una aplicación de una manera intuitiva, cómoda y que tenga una agradable interfaz.

Haremos algunos ajustes al css:

- volvemos al fondo blanco
- el font será un poco más grande,
    - no lo determinaremos por píxeles fijos. En general no es una buena técnica pensar en valores fijos, sino trabajar en medidas relativas, como el *em* (tamaño del font del contenedor donde se encuentra el elemento), *rem* (tamaño del font del contenedor principal) o %. Pueden aprender más sobre medidas relativas en css workshop o en sitepoint. Por otra parte, este blog cuenta algunas anécdotas sobre el uso de tipografías relativas y la experiencia mobile.
    - también trabajaremos con otro tipo de letra (las sans-serif)
    - Todo esto lo haremos para el contenido que está dentro del body (h2, div, etc.) entonces estará encerrado dentro de un tag body.
    - el título tendrá un color más azul
- agregaremos espacios entre divs (padding-top, padding-bottom, etc.)
- y por último cambiaremos la vista de la tabla
    - que tomará todo el ancho del contenedor (100%)
    - los th y td tendrán un padding de 10px y el texto se alineará a la izquierda por defecto
    - para distinguir cada fila pondremos colores diferentes a las filas pares e impares
    - y le agregaremos un color diferente al encabezado de la tabla

Esto se escribe:

```css
h2 {
    color: #0080FF;
    text-align: center;
}

body {
    font-family: sans-serif;
    font-size: 1.2em;
}

input {
  font-family: sans-serif;
  font-size: 1em;
}

div {
    width: 95%;
    padding: 0.5em;
}

table {
    width: 100%;
}

tr:nth-child(even) {background-color: #f2f2f2}

th, td {
    padding: 15px;
    text-align: left;
}

th {
    background-color: #2E9AFE;
    color: white;
}
```

Podemos ver que se pueden compartir varios valores para los mismos tags:
`th, td { ...`
(vale para los tags th, o para los td)

Y también hay algunos atisbos de cosas que parecen tener lógica:
`tr:nth-child(even) {`

(las filas pares definen un background color diferente).

Como resultado, cambiando solo el css y sin hacer ningún ajuste en el html tenemos algo que va queriendo más:

Bastante mejor, ¿no?

Como además definimos el ancho de la tabla al 100% y el título centrado respecto a su contenedor, podemos jugar a cambiar el tamaño del navegador, y vemos ahora que la tabla se acomoda en forma proporcional (adquiere signos de *responsiveness*).

5.2 ¿Por qué cascading style sheet?

Esta explicación está extraída de [este artículo de la wiki de uqbar](). Para profundizar recomendamos su lectura.

Las reglas se aplican en cascada, esto significa dos cosas:

1. En primer lugar cada componente hereda determinados estilos de sus contenedores, por ejemplo un td (celda de una tabla) hereda los del tr (fila) y del table correspondientes. Cuando un componente define un

estilo pisa la definición que tenía del contenedor, mientras que aquellos que no estén especificados se heredan. No todas las indicaciones de estilo son "heredables" (inheritable en inglés), es importante entender el comportamiento de cada una de las diferentes indicaciones de estilo.

2. En segundo lugar se puede aplicar más de un estilo a un componente en base al tag, class y id respectivamente. Esos diferentes estilos se van a combinar permitiendo que el estilo más específico sobreescriba los estilos más generales, pero manteniendo las indicaciones correspondientes al estilo más general que no sean redefinidas.

Vamos a agregar al css un class “centrada” que alinea el texto al centro de su contenedor:

```
.centrada {
    text-align: center;
}
```

Y lo podemos aplicar para el th y td de la foto:

```
    <table>
            <tr>
              <th>Nombre</th>
              <th>Apodo</th>
              <th class="centrada">Foto</th>
              <th>Fecha amistad</th>
              <th>A&ntilde;os</th>
            </tr>
            <tr>
              <td>Juan Contardo</td>
              <td>Juancho</td>
              <td class="centrada"><img src="images/juancho.jpg"
    height="50px;" width="50px;"></td>
              <td>07/04/2011</td>
              <td>5</td>
            </tr>
```

Entonces al aplicar el css al html la columna de la foto queda centrada (al menos para Juan):

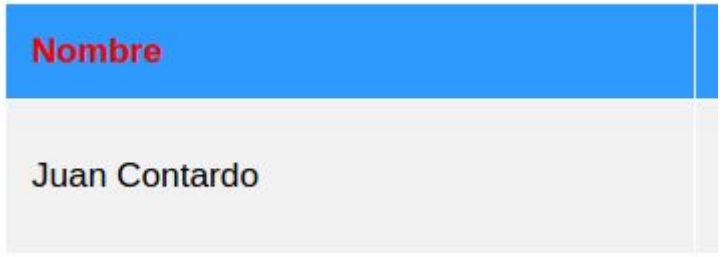
Nombre	Apodo	Foto	Fecha amistad	Años
Juan Contardo	Juancho		07/04/2011	5

El uso de tags por id consiste en definir una configuración para aquellos elementos que tengan como id el valor ingresado en el css. Por ejemplo, en el css escribimos:

```
#enRojo {
   color:red;
}
```

Y luego en el html lo referenciamos por id:

```
<table>
   <tr>
     <th id="enRojo">Nombre</th>
```



El uso de ids en el css se desaconseja como práctica, ya que dificulta reutilizar la definición en más de un lugar.

6 Utilizando frameworks UI

Desarrollar un css requiere mucho tiempo y esfuerzo, por eso utilizar frameworks nos dan un conjunto de herramientas para abordar los desafíos y complejidades que la industria requiere.

Qué brinda un framework css?

- una propuesta de diseño consistente (por ejemplo, definiendo qué es *mobile first* o *responsive*)
- mejoras en la experiencia de usuario (ej. animaciones cuando estamos cargando un input como resaltarlo cuando tiene el foco, un botón que muestra un spinner mientras está ejecutando, etc.)
- una biblioteca de componentes que habilita a usar patrones de diseño de interfaces de usuario, como el formulario modal, el concepto de WYSIWYG, el password strength meter para medir qué tan buena es una contraseña, etc.

Elegir un framework de css también puede traernos alguna desventaja: ej. si nos proponen una página dividida en 12 columnas, es una estructura difícil de trabajar y donde es fácil errar el camino.

Algunos ejemplos son

- **Material Design** http://www.google.com/design/spec/material-design/introduction.html, desarrollado por Google
- **Bootstrap Twitter** http://getbootstrap.com/, desarrollado por Twitter
- **Bulma** https://bulma.io/
- Para otras opciones ver https://en.wikipedia.org/wiki/CSS_frameworks

En esta oportunidad vamos a jugar un poco con Material Design for Bootstrap, una mezcla de los dos primeros frameworks.
http://mdbootstrap.com/

Lo bueno... tenemos un framework que resolverá muchas cosas que de otra manera implicaría un esfuerzo de css (y generalmente también de javascript) por parte nuestra. El costo... tenemos que aprender del framework y adaptarnos a su metáfora y reglas de juego.

Por suerte ya tenemos en nuestro proyecto base los archivos requeridos para empezar a trabajar. Primero que nada, debemos incorporar los css en el header de nuestro list.html:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>Lista de amigos</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <!-- Font Awesome -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.0/css/font-aw
esome.min.css">
    <!-- Bootstrap core CSS -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- Material Design Bootstrap -->
    <link href="css/mdb.min.css" rel="stylesheet">
  </head>
```

También al final hay que agregar archivos javascript:

```
      </div>
      <!-- SCRIPTS -->
          <!-- JQuery -->
          <script type="text/javascript"
src="js/jquery-2.2.3.min.js"></script>
          <!-- Bootstrap tooltips -->
          <script type="text/javascript"
src="js/tether.min.js"></script>
          <!-- Bootstrap core JavaScript -->
          <script type="text/javascript"
src="js/bootstrap.min.js"></script>
          <!-- MDB core JavaScript -->
          <script type="text/javascript"
src="js/mdb.min.js"></script>
    </body>
  </html>
```

6.1 Primeros cambios

Vamos con los dos primeros ajustes:

- animaremos el título “Lista de amigos” haciéndolo rebotar al comienzo
- y también utilizaremos los botones de material design

Lo primero se logra utilizando la clase animated bounce:

```
 <body>
    <h2 class="animated bounce">Lista de amigos</h2>
    <div>
```

Lo segundo mediante el tag button específico:

```
       <input type="text" id="nombreComienzaCon"
placeholder="nombre" max="25">
      </div>
      <div>
        <button id="crearAmigo" class="btn btn-primary">Crear
amigo</button>
        <button id="buscarAmigos" class="btn btn-info"
autofocus="true">Buscar</button>
      </div>
```

Vemos la animación del título y los botones nuevos que tienen sombra

Ah, además vemos que hay otros cambios:

- la fuente Roboto por defecto
- los input (que por el momento toman todo el width)

ambos tienen una gran reminiscencia a Android.

6.2 Efectos sobre las fotos

Ahora agregaremos un efecto de zoom sobre la foto cuando el usuario se posicione sobre la misma:

```
      <tr>
           <td>Juan Contardo</td>
           <td>Juancho</td>
           <td class="centrada">
             <div class="view hm-zoom">
               <img src="images/juancho.jpg" height="60px;"
width="60px;">
             </div>
           </td>
           <td>07/04/2011</td>
           <td>5</td>
         </tr>
         <tr>
           <td>Clara Allende</td>
           <td>Clari</td>
           <td class="centrada">
             <div class="view hm-zoom">
               <img src="images/clari.jpg" height="60px;"
width="60px;">
             </div>
```

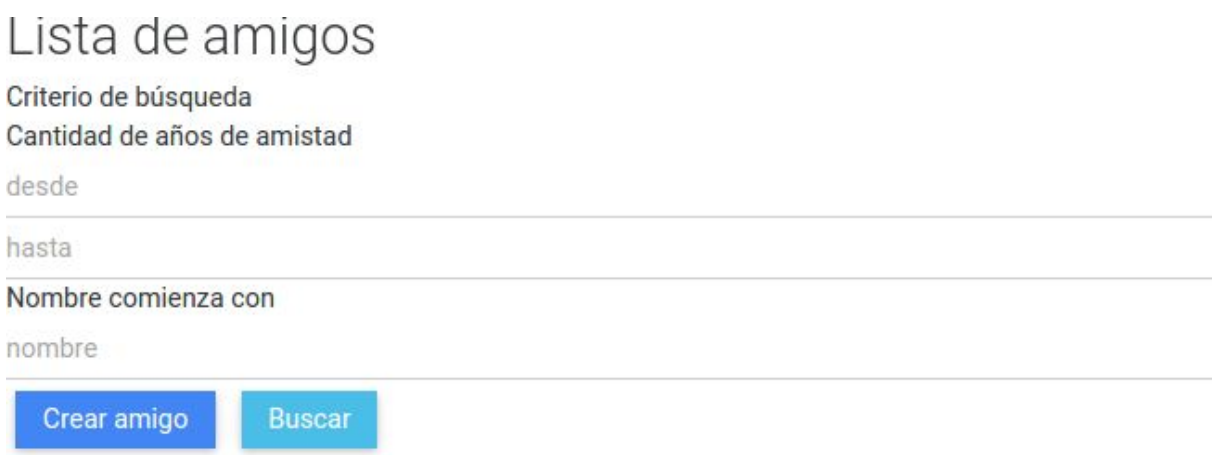
```
          </td>
          <td>12/06/2010</td>
          <td>6</td>
        </tr>
        <tr>
          <td>Leo Cesario</td>
          <td>HH</td>
          <td class="centrada">
            <div class="view hm-zoom">
              <img src="images/leo.jpg" height="60px;"
width="60px;">
            </div>
          </td>
          <td>05/11/2006</td>
          <td>10</td>
        </tr>
```

Es un poco repetitivo, sí, pero es el precio de estar maquetando algo estático. Cuando agreguemos elementos dinámicos podremos unificar todo en un solo lugar.

Lo pueden probar, y ahora cuando posicionamos el cursor sobre la foto se agranda ligeramente.

6.3 Mejoras sobre el criterio de búsqueda

El criterio de búsqueda está algo desprolijo:



Vamos a hacerle algunas mejoras:

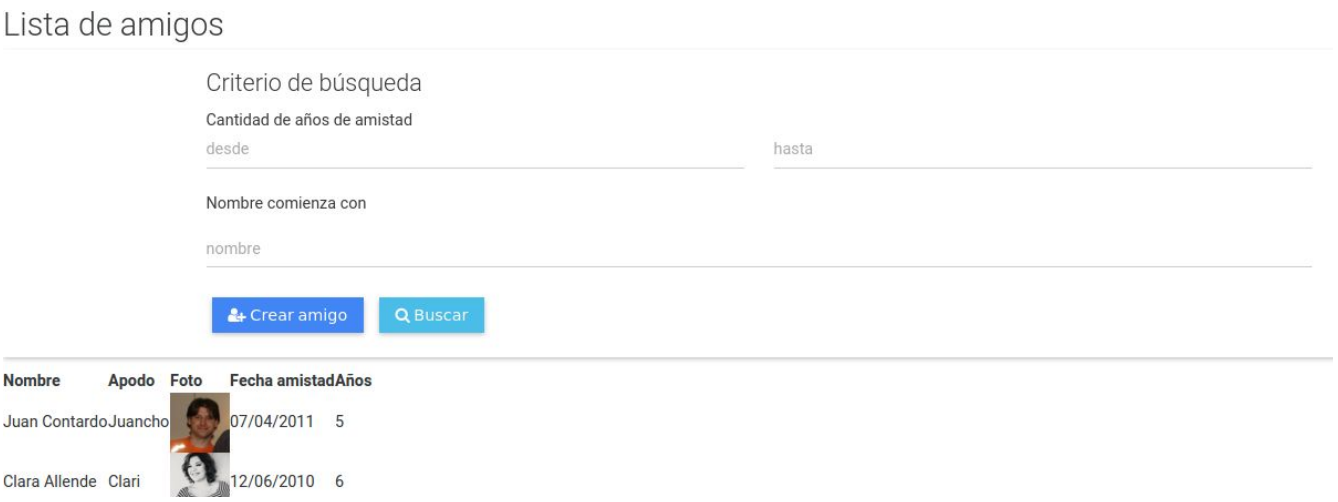
- Podemos agrupar el div que contiene al criterio de búsqueda generando una *card*, un container con características especiales
    - eso incluye mostrar el título “Criterio de búsqueda” que resalte respecto a los labels de cada campo
- Podemos tener en una sola línea los rangos desde/hasta cantidad de años. Esto requiere que pasemos de un modelo en donde los controles se acomodan “uno al lado del otro” a un layout que se compone de una grilla con 12 columnas, nosotros manejamos hasta 12 columnas a la vez. Cuando queremos separar en dos columnas, cada columna está tomando 6 de esas 12 que tiene el ancho total. Si quisiéramos tener 3 columnas, tenemos que pensar en columnas que toman 4 columnas del ancho total. Por eso el número que debemos utilizar depende de la cantidad de columnas deseada:
  `número = 12 / cantidadColumnasDeseada`
  Y también debemos especificar para qué tipo de dispositivo se va a ver mejor, por eso los tamaños xs, sm, md, lg.
- A los botones les podemos agregar un ícono representativo, utilizando el conjunto de íconos Font Awesome que viene con Bootstrap

El nuevo container que tiene el criterio nos queda

```
<body>
    <h2 class="animated bounce">Lista de amigos</h2>
    <div class="card">
      <div class="card-block">
        <div class="container">
          <h4 class="card-title">Criterio de
b&uacute;squeda</h4>
        </div>
        <div class="container">
          <div class="row">
            <div class="col-sm-12">
              <p class="card-text">Cantidad de
a&ntilde;os de amistad</p>
            </div>
            <div class="col-sm-6">
              <input type="number" id="aniosDesde"
placeholder="desde" size="5" max="5">
            </div>
            <div class="col-sm-6">
              <input type="number" id="aniosHasta"
placeholder="hasta" size="5" max="5">
            </div>
```

```
            </div>
          </div>
          <br>
          <div class="container">
            <div class="row">
              <div class="col-sm-12">
                <p class="card-text">Nombre comienza
con</p>
                <input type="text" id="nombreComienzaCon"
placeholder="nombre" max="25">
              </div>
            </div>
          </div>
          <br>
          <div class="container">
            <div class="row">
              <div class="col-sm-12">
                <button id="crearAmigo" class="btn
btn-primary"><i class="fa fa-user-plus"
aria-hidden="true"></i>  Crear amigo</button>
                <button id="buscarAmigos" class="btn
btn-info" autofocus="true"><i class="fa fa-search"
aria-hidden="true"></i>  Buscar</button>
              </div>
            </div>
          </div>
        </div>
      </div>
```

¡Terrible el anidamiento de divs!
Pero esto nos permite visualizar un criterio de búsqueda con más estilo:



6.4 Mejorando la tabla

Ahora nos concentraremos en mejorar la tabla, para eso

- el ancho debe volver a tomar el 100%
- hay que ponerle algún efecto de hover a la tabla cuando nos posicionemos en algún elemento
- y sería bueno que tenga un efecto *striped* (o *zebra*) en el que se diferencien las filas pares e impares.

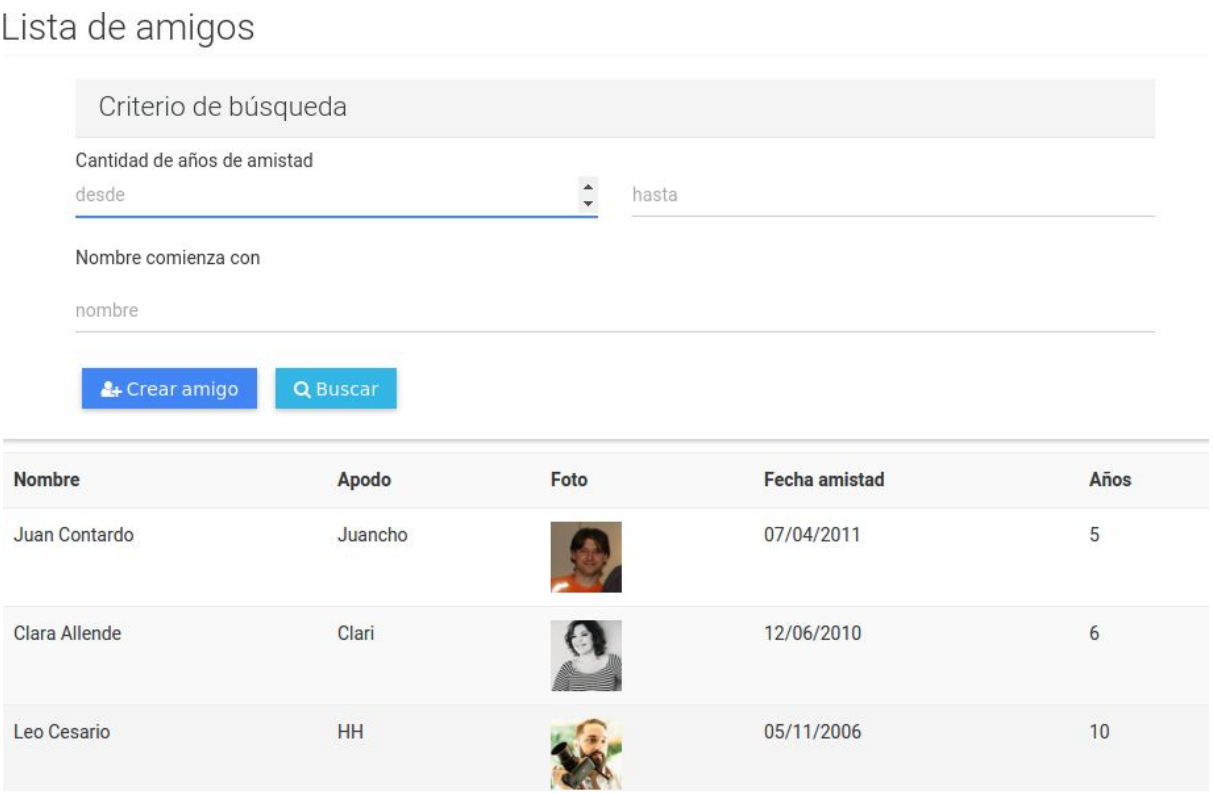
Es poco lo que hay que hacer en este caso, simplemente en la definición de la table escribimos:

```
<table class="table table-striped table-hover">
```

Y agregamos un cambio más, “Criterio de búsqueda” como card-header en lugar de card-title:

```
<h4 class="card-header">Criterio de b&uacute;squeda</h4>
```

Con eso solo ya nos alcanza para tener la página casi lista:



Podemos cambiar el tamaño de la pantalla y vemos cómo el diseño de la página es responsive, ya que ajusta el tamaño de los controles en base al contenedor principal.

La versión final del proyecto la pueden encontrar en el branch master de este repo: https://github.com/uqbar-project/eg-amigos-web

7 Resumen

La tecnología web implica trabajar una serie de lenguajes de diversa índole. En particular, en el front end el navegador web maneja

- html como lenguaje para manipular contenido
- css como lenguaje para presentar ese contenido
- ninguno de los anteriores son lenguajes de programación como sí lo es javascript que tiene una amplia variedad de usos (que dejaremos para otra oportunidad)

En el maquetado web partimos de un contenido estático, sin funcionalidad real, para ir encontrando lo que será la cara de la aplicación final. No es una tarea de programación, pero en muchos trabajos termina siendo una tarea interdisciplinaria entre el diseñador gráfico y un especialista en contenidos y usabilidad, que define el diseño de la interfaz a construir.