Bootstrap y AngularJS: Tutorial práctico

Procesos del Software

José Antonio Martínez López - joseantonio.martinez7@alu.uclm.es

Pedro Manuel Gómez-Portillo López - pedromanuel.gomezportillo@alu.uclm.es

Índice

Introducción

Antes de introducirnos en los detalles técnicos de los lenguajes de desarrollo web demos un breve repaso a cómo funcionan los sitios web. Éstos están formados por un servidor que provee páginas web a sus clientes en forma de archivos HTML, que no son otra cosa que archivos de texto plano que el navegador renderiza como páginas web.

Pues bien, hay muchas maneras para desarrollar ese HTML, y dos de ellas(las cuales son compatibles entre sí) son a través de los frameworks Bootstrap y AngularJS, mediante los cuales podremos desarrollar pa ǵ inas web atractivas y dinámicas, respectivamente.

Este trabajo tiene como objeto servir de tutorial de iniciación a todo aquel que quiera aprender estas dos tecnologías.

Bootstrap

¿Qué es?

Twitter Bootstrap es un *framework* o conjunto de herramientas para el diseño de páginas y aplicaciones web. Bootstrap contiene plantillas de diseño de formularios, botones, menús de navegación y otros elementos prediseñados basados en HTML, CSS y JavaScript.

¿Cómo lo uso?

Para usarlo, lo primero que tendrá que tendremos que hacer será añadir Bootstrap a nuestra página web, ya sea descargando sus archivos e importándolos en el *<head>* de nuestra página web o simplemente importarlos ya compilados desde internet. Si optamos por la segunda opción tendremos que importar la librería jQuery, el JavaScript y el CSS de Bootstrap desde un CDN. En el ejemplo de abajo son importados desde BootstrapCDN, aunque hay varios proveedores.

```
<head>
<script src = "https://code.jquery.com/jquery.js"></script>
<script src =
"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
<link rel = "stylesheet" href =
"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs">
```

```
</head>
```

Una vez añadido ya podremos incluir el código HTML de los elementos que queramos usar en nuestra página web.

Características

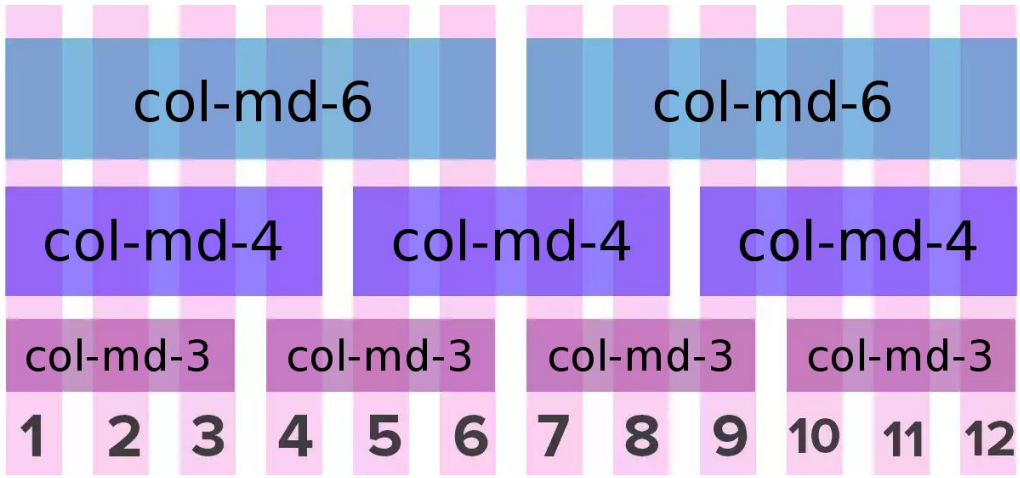
- Diseño en columnas

  Bootstrap trabaja con una disposición fija de 12 columnas, de tal manera que el desarrollador decide qué elementos sitúa en cada columna y cuántas ocupa cada uno. Las columnas se pueden combinar (como se ve en la imagen) para personalizar el diseño de la página.

  Para usarlo tendremos que crear un *div* (una sección) de la clase "col-md-n", siendo *n* el número de columnas que quiere que ocupe éste.

  *e.g. *

  La sección empezará a partir de la última columna utilizada y sólo se queda en la misma fila si hay espacio suficiente, de lo contrario pasa a la siguiente fila. Por ejemplo, si ya hemos ocupado 10 columnas y añadimos un elemento que ocupe 4 éste pasará a fila de abajo.



- Soporta diseños responsive:

  Esto significa que el diseño y la disposición de los elementos de la página se ajusta de forma dinámica teniendo en cuenta las características del dispositivo desde el que se accede a la página web, ya sea una pantalla 4K, una tablet o un móvil de 5 pulgadas.

- Componentes prediseñados

  Una de las principales ventajas de Bootstrap es que contiene una gran cantidad de elementos listos para su uso como botones, listas desplegables, etiquetas, imágenes tipográficas (*glyphicons*), sistema de pestañas, formatos para mensajes de alerta o barras de progreso, entre muchos otros.

  Para usarlos, simplemente tendremos que copiar el código HTML de los elementos que queramos y pegarlo en nuestra página web o implementar los nuestros usando las clases de Bootstrap.

- Es de código abierto y está disponible en GitHub

  Los desarrolladores están motivados a participar en el proyecto y a hacer sus propias contribuciones a la plataforma, de tal manera que esta está en continua evolución y los errores que se encuentran pueden ser rápidamente solucionados.

AngularJS

¿Qué es?

AngularJS es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para desarrollar la parte funcional de una página web. Su objetivo es aumentar las aplicaciones basadas en navegador implementando el patrón Modelo-Vista-Controlador (MVC), es decir, separar la interfaz de la parte funcional.

¿Cómo lo uso?

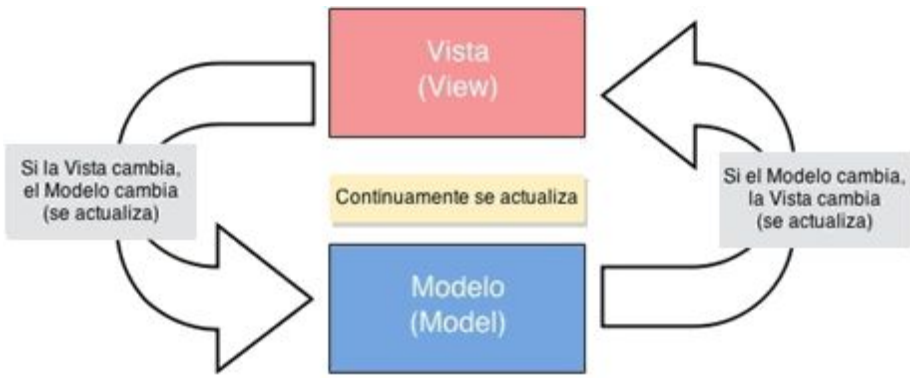
Para usarlo en nuestro sitio web, y del mismo modo que Bootstrap, tendremos o bien que descargarlo e importarlo o bien usar un CDN que nos permita añadirlo sin necesidad de descargarlo. En este caso el propio Google se ofrece como proveedor.

```
<head>
<script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.5.7/angular.min.js">
</script>
</head>
```

Características

- Actualización en tiempo real

  Una de las características más importantes, hace referencia a que AngularJS está continuamente observando los cambios que suceden tanto en la vista como en el modelo de datos y sincroniza datos entre estos. Si la vista cambia, modifica el modelo y viceversa.



- Centrado en expresiones

  Cuando queramos escribir resultados de funciones en la vista colocaremos estas expresiones entre llaves dobles. Vemos unos ejemplos:

  - `{{1 + 2}}` mostrará 3.
  - `{{mensaje}}` mostrará el valor de la variable `mensaje`.
  - `{{user.name}}` escribirá el valor que tiene el atributo `name` en el objeto `user`.

- Orientado a objetos

  AngularJS implementa la orientación a objetos, beneficiándose así de las ventajas de este paradigma de programación.

  La manera de crear estos objetos es ir añadiendoles los diferentes atributos conforme se necesitan, por ejemplo:

```
<div ng-app>
      <input type="text" ng-model="empleado.nombre">
      <input type="text" ng-model="empleado.apellido">
      <input type="text" ng-model="empleado.direccion">
      <p>{{empleado}}</p>
</div>
```

  De esta manera el objeto `empleado` adquirirá los atributos `nombre`, `apellido` y `direccion`, y al imprimirlo obtendremos algo como:

```
{“nombre”: “Juan”, “apellido”: “Lopez”, “direccion”: “Calle Mata”}
```

Elementos

- Módulos

  Los módulos son contenedores de diferentes partes de nuestra aplicación. Podemos definir la cantidad de módulos que necesitemos para desacoplar el código, sea por características, por funcionalidad, por componente reusable, etc. Cuanto más desacoplado tengamos nuestro código mucho más fácil será mantenerlo y escalarlo.

Un módulo se declara de la siguiente manera:

```
var modulo = angular.module('Nombre_del_modulo', []);
```

- Controladores

Son objetos que contienen la lógica de las aplicaciones. Enlazan los módulos con la parte funcional que queramos asignares. Para crear un controlador tendremos que haber definido previamente el módulo al que se lo queremos asignar y a continuación pasarle el nombre del controlador y una función con la parte funcional.

```
var modulo = angular.module('Nombre_del_modulo', []);
modulo.controller('Nombre_del_controlador', function($scope){
      //contenido funcional del módulo
});
```

- Directivas

Las directivas son la forma en la que AngularJS extiende nuestro HTML, permitiéndonos crear nuevas características y añadir comportamiento.

Cuando comenzamos a ver AngularJS, nos encontramos con muchas de las directivas que forman este *framework*. Algunas de ellas son ng-app, ng-controller, ng-model, ng-repeat, ng-click o ng-show, entre otras. Todas ellas cumplen una función en particular, pero a continuación vamos a ver las más comunes.

❖ ngApp (ng-app):

Es la directiva que indica dónde empieza nuestra aplicación y se debe colocar como atributo en la etiqueta que será la raíz de la misma. A partir de aquí ya podremos empezar a trabjar con AnguarJS.

En el siguiente ejemplo definimos una sección con una aplicación en la que sumaremos dos números.

```
<div ng-app>
      <p>El resultado de sumar 1 y 2 es {{1+2}}</p>
</div>
```

- ngModel (ng-model)

Da un nombre a un elemento del HTML y permite obtener la información ingresada por el usuario si es un elemento de entrada o modificarlo desde otra parte. En el ejemplo siguiente crearemos un cuadro de entrada de texto, obtendremos lo que el usuario escribe y lo mostraremos en la fila de abajo, detrás de la palabra “Hola”.

```
<div ng-app>
      <label>Ingrese su nombre</label>
     <input type="text" ng-model="nombre">
      <span>Hola {{nombre}}</span>
<div/>
```

- ngController (ng-controller)

Esta directiva enlaza el HTML con una función el AngularJS. En el siguiente ejemplo rellenaremos automáticamente una entrada de texto con la palabra “Juan”:

```
<div ng-app = "miApp" ng-controller = "miCtrl">
    <label>Nombre</label>
    <input type="text" ng-model="nombre">
  </div>

  <script>
     var app = angular.module("miApp", []);
     app.controller("miCtrl", function($scope) {
          $scope.nombre = "Juan";
     });
  </script>
```

Como vemos, definimos en el *<body>* el modelo “nombre” dentro del alcance de la aplicación y el controlador y luego, dentro de la etiqueta *<script>,* definimos una función para esa aplicación y controlador, en la que asignamos inicialmente el valor “Juan” al modelo “nombre”.

---

[1]El argumento *$scope* describe el alcance del módulo. Una forma de entenderlo es como el *this* en Java o el *self* en Python.

- ngClick (ng-click)

  Esta directiva trabaja con un evento de clickado sobre un elemento HTML, normalmente un botón. Permite ejecutar una función en AngujarJS cuando este evento sea llamado. En el ejemplo práctico que hemos preparado se puede ver esta directiva en funcionamiento.

- ngChange (ng-change)

  Esta directiva detecta cualquier cambio que se produzca dentro de una etiqueta de entrada, ya sea de texto, una checkbox, etc. y permite llamar a una función de AngujarJS.

```
<div ng-app = "miApp" ng-controller = "miCtrl">
     <input type="checkbox" ng-change="miFuncion()"
                                        ng-model="tick">
     <p>{{mensaje}}</p>
</div>

<script>
var app = angular.module("miApp", []);
app.controller("miCtrl", function($scope) {
     $scope.mensaje = "Aún no has pulsado la checkbox";
     $scope.miFuncion = function() {
          $scope.mensaje = "Activada!";
     };
});
</script>
```

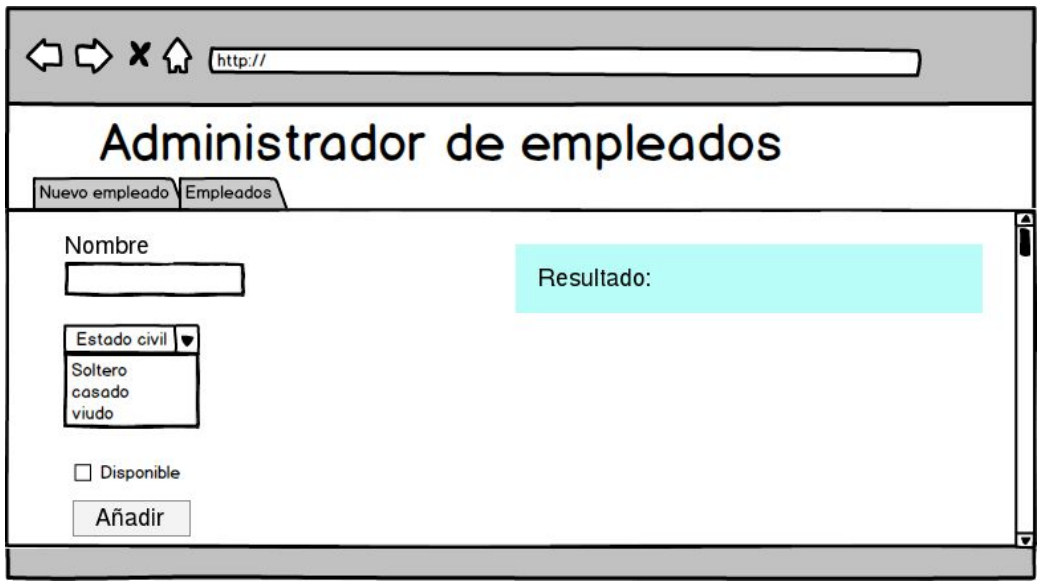
  En este ejemplo mostramos una checkbox y un mensaje, que inicialmente dirá `"Aún no has pulsado la checkbox"` y que cambiará cuando el usuario la active.

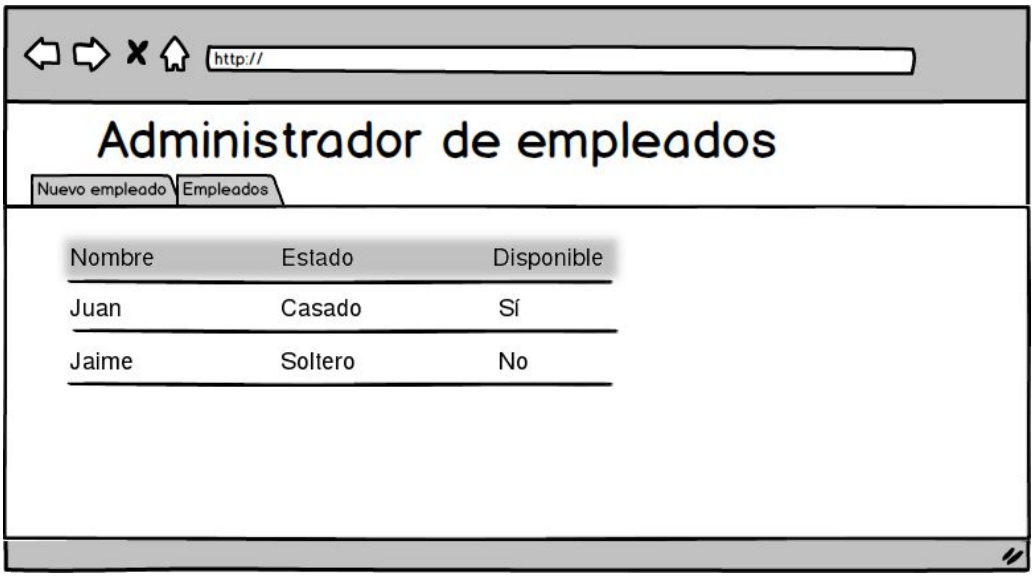
Ejemplo práctico

El código de la página web puede verse en este repositorio.

En este ejemplo vamos a ver un uso básico de ambos *frameworks*. Conviene tener claros los conceptos básicos de HTML, como etiquetas y clases, y saber algo de CSS. En la sección Enlaces interesantes hemos dejado algunos tutoriales de utilidad, en especial el nº1, ya que ofrece una introducción clara y rápida a estas nociones.

Desarrollaremos un ejemplo básico de un SI de administración de personas sin persistencia. Lo primero que vamos a hacer es diseñar el boceto de la página web que vamos a crear.



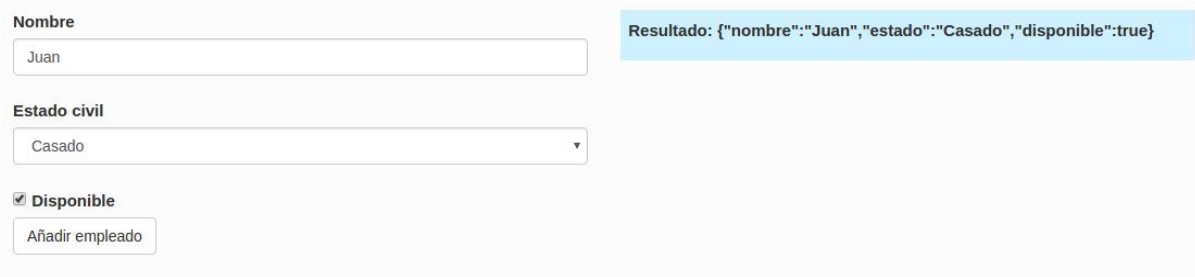


La página consistirá en un encabezado y dos pestañas, una con un formulario para añadir los datos de los empleados (nombre, estado civil y disponibilidad) y la otra con una tabla con ellos.

Pensando en el uso de Bootstrap, hemos añadido un header con un glyphicon, el título de la aplicación y nuestros nombres.

```
40  <div class="page-header">
41    <h1>
42      <span class="glyphicon glyphicon-user"></span>
43      Administrador de empleados <small>Pedro Manuel Gómez-Portillo y José Antonio Martínez</small>
44    </h1>
45  </div>
```

Además, en la primera pestaña hemos creado dos columnas, ambas de 6 unidades de ancho (* *). En la de la izquierda mostraremos el formulario con la entrada de datos y en la derecha el estado del objeto *empleado* de AngularJS.



Por otro lado, hemos usado algunas clases de Bootstrap para los inputs, como *form-control* o *btn*.

```
62  <input type="text" class="form-control" ng-model="empleado.nombre">
```

```
78  <button type="button" class="btn btn-default" ng-click="addEmpleado()">
```

La tabla tiene dos clases, *table* y *table-hover*. La segunda nos permite destacar una fila al pasar el ratón sobre ella.

```
94  <table class="table table-hover" id="tabla-empleados">
```

Y ahora pasemos a AngularJS. Nuestra aplicación empezará en la sección del formulario y lo abarcará entero, ya que necesitamos las funcionalidades que nos proporciona este framework en todo él. Le asignaremos un módulo llamado *miApp* y un controlador al que llamaremos *miCtrl*.

```
58  <div class="formulario" ng-app = "miApp" ng-controller = "miCtrl">
```

Más adelante, dentro de una etiqueta *<script>*, crearemos el módulo y le añadiremos un controlador.

```
109  var mainApp = angular.module("miApp", []);
110  mainApp.controller('miCtrl', function($scope) {
```

Ahora ya podemos usarlo, así que crearemos un modelo (*ng-model*) para cada elemento de entrada, todos de la clase *empleado*:

```
62  <input type="text" class="form-control" ng-model="empleado.nombre">
66   <select class="form-control" ng-model="empleado.estado">
74  <input type="checkbox" ng-model="empleado.disponible">
```

De este modo, cuando imprimamos el objeto *empleado* se mostrarán todos sus atributos.

```
85   <p>Resultado: {{empleado}}</p>
```

Resultado: {"nombre":"Juan","estado":"Casado","disponible":true}

Por otro lado, para implementar la funcionalidad al botón le añadiremos la directiva ng-click, de tal forma que al pulsarlo se llame a la función *addEmpleado()*.

```
78  <button type="button" class="btn btn-default" ng-click="addEmpleado()">
```

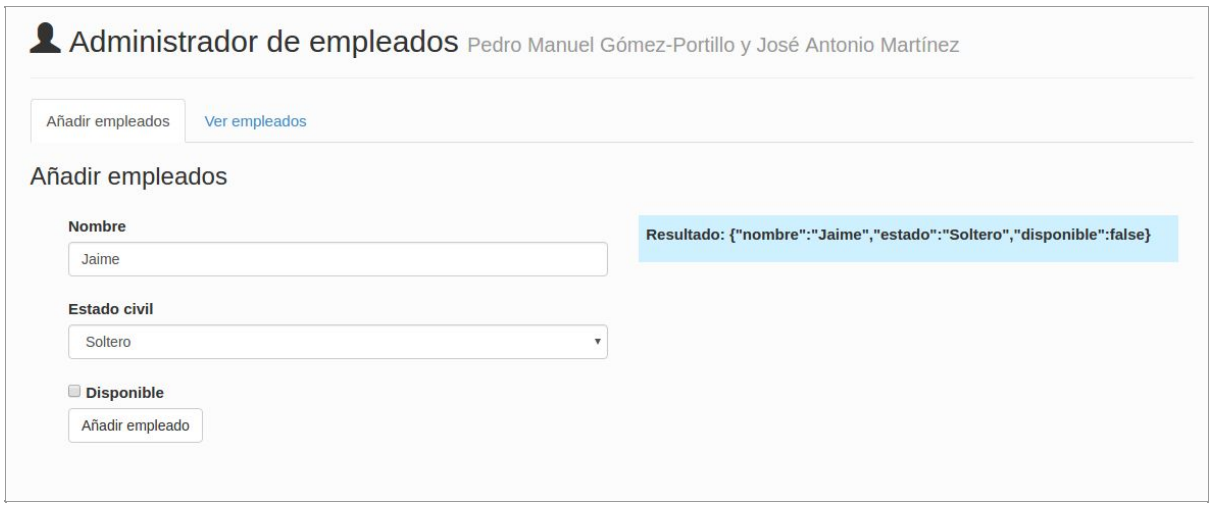
Volviendo al modelo y el controlador, pasemos a crear dicha función. Siguiendo por la línea 111, diremos que el objeto llamado *addEmpleado* será de tipo función, y en la 112 empezamos a implementarla.

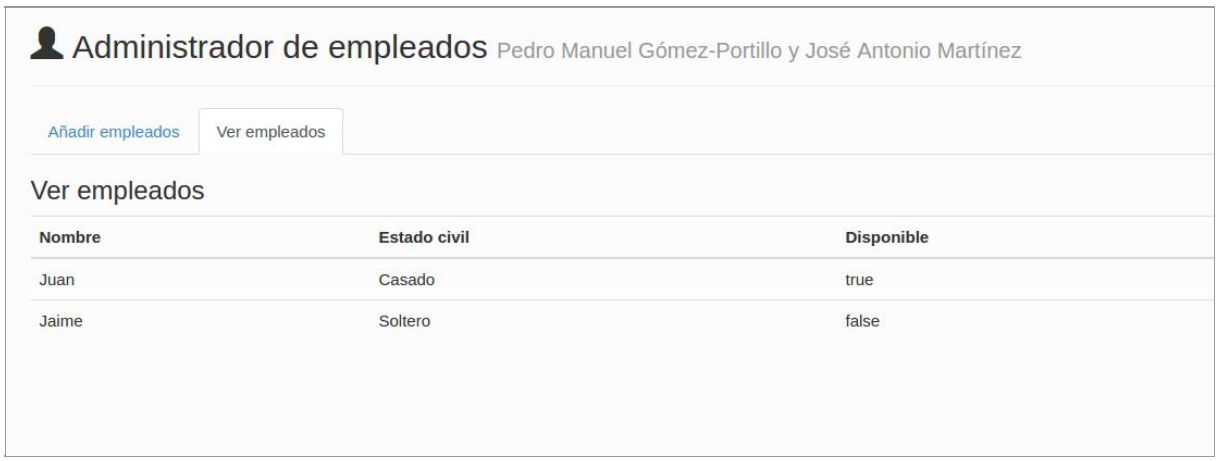
```
108  <script>
109  var mainApp = angular.module("miApp", []);
110  mainApp.controller('miCtrl', function($scope) {
111    $scope.addEmpleado = function() {
112      var nueva_fila= '<tr><td>' + $scope.empleado.nombre +
113                      '</td><td>'+ $scope.empleado.estado +
114                      '</td><td>'+ $scope.empleado.disponible +
115                      '</td></tr>'
116      $('#tabla-empleados > tbody:last-child').append(nueva_fila);
117    }
118  });
119    </script>
```

Las siguientes líneas no forman parte de AnguarJS, aún así las explicaremos brevemente. Creamos una variable en la que guardaremos el código HTML de una línea de tabla con los atributos del objeto *empleado*.

La línea 116 es una instrucción jQuery (aprovechando que tenemos que importar esta librería para Bootstrap) en la que buscamos un elemento con el id=”tabla-empleados”, seleccionamos la primera etiqueta *tbody* que tenga dentro y luego el último elemento de esa etiqueta, que será por definición la última fila que forme esa tabla. Una vez lo tengamos, añadimos después la fila que acabamos de crear.

Y con esto acabamos el diseño y la implementación de este ejemplo, que queda así:





Enlaces interesantes

1. http://marksheet.io/ - Tutorial básico de HTML y CSS
2. http://es.html.net/tutorials/css/ - Tutorial de CSS
3. http://getbootstrap.com/css/ - Ejemplos de Bootstrap
4. http://www.w3schools.com/angular/angular_intro.asp - Tutorial de AngularJS
5. https://frontendlabs.io/2152--hablemos-de-angularjs -Introducción AngularJS
6. https://frontendlabs.io/2287--aprendiendo-directivas-en-angularjs - directivas