Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Documentación Proyecto Final Desarrollo Web

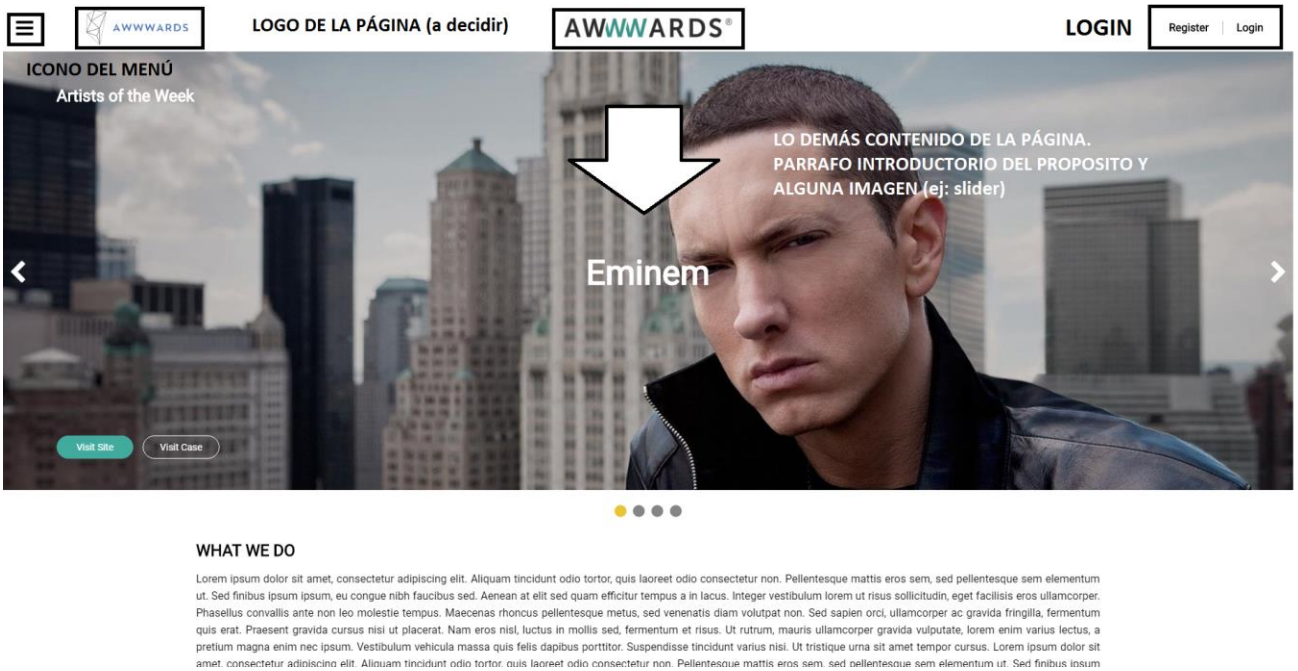
Primera Entrega

Referencia y comienzo

En primer lugar tomé como referencia una página web para sacar ideas como estructura y diseño general, colores y alguna funcionalidad. La misma es: https://www.awwwards.com/ .

En esta empecé a investigar cómo realizaba estructuras y funcionalidades, inspeccionando el código mediante la herramienta de desarrolladores de Google Chrome.

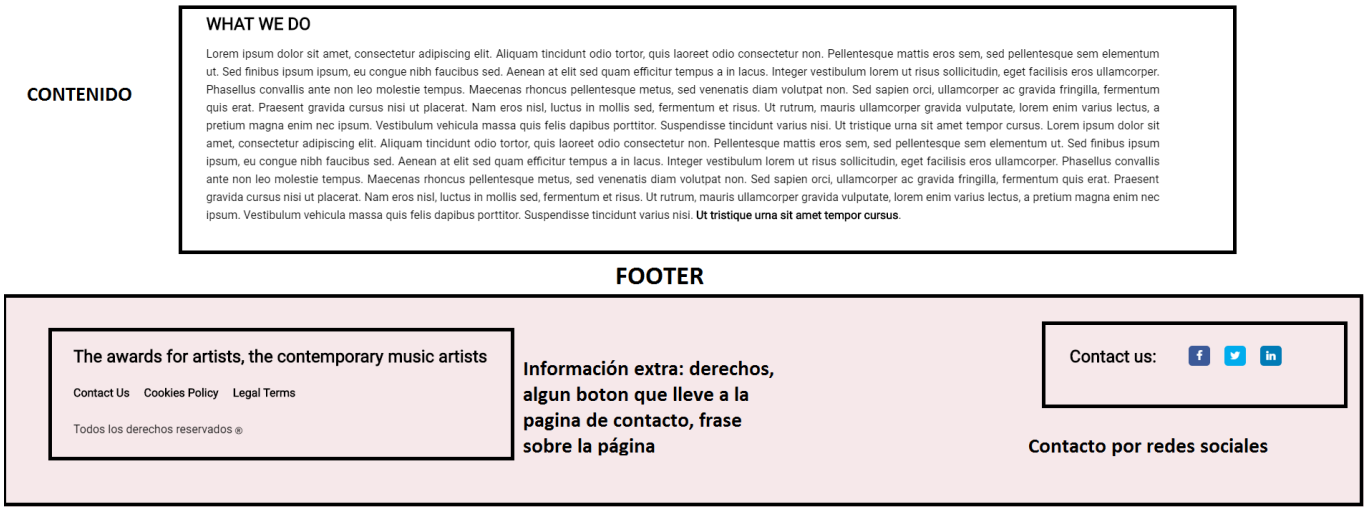
Fue ahí cuando empecé con una estructura básica de HTML, teniendo un header (encabezado), nav (navegación), section (sección) y un footer (pie de página). Al considerarme bastante ambicioso, lo primero que hice luego de tener la estructura general fue buscar cómo hacer un slider menú. Para hacer el mismo encontré un video tutorial de la plataforma Youtube en el que explicaba todos los pasos para realizarlo. En este primer caso la estructura se trataba de 2 funciones de JavaScript en las que se cambiaba la propiedad “left” de CSS en el evento “onclick” del elemento donde se encontraba el icono de menú. En esta pasaba de estar en -240px a 0 el menú al ser abierto y de 0 a -240px cuando se cerraba. El menú estaba estructurado con una lista no ordenada de links los cuales contenían las distintas páginas del proyecto.



Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Primeros detalles estéticos

También agregué a la misma 3 botones de redes sociales, los cuales en el momento de ser “tocados” por el mouse (hover selector) se elevan dejando una sombra en la pantalla. Esto se realiza con la propiedad transform y el ejemplo en este caso sería: transform: translateY (-5px); box-shadow: 0 6px 6px rgba (0, 0, 0, .5). Con estas dos propiedades movemos al elemento 5 pixeles en el eje de las Y y creamos una sombra al mismo con ese color rgba negro de 0,5 de opacidad.



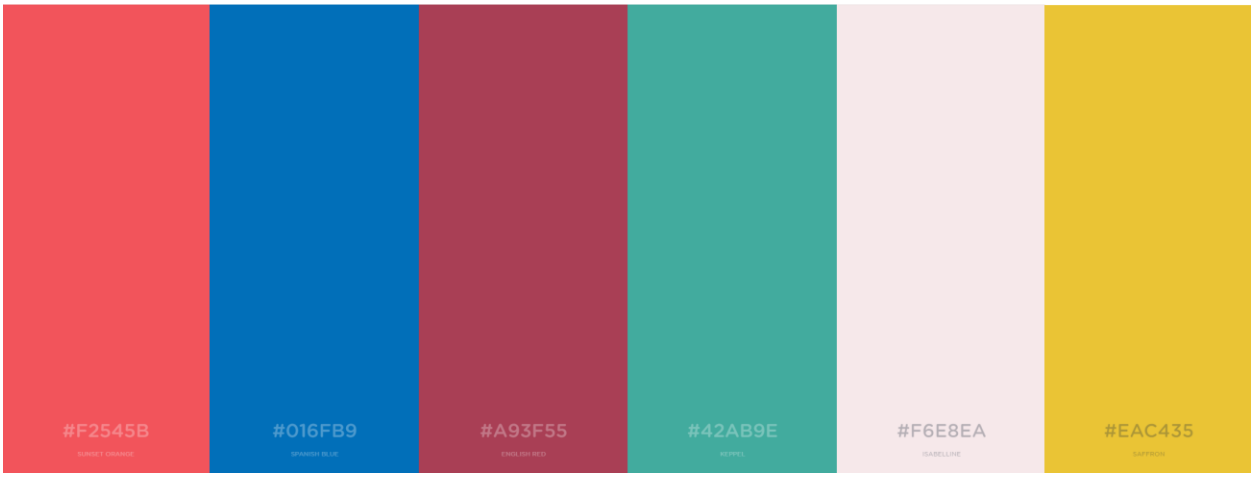
Esas eran las dos funcionalidades principales que tenía la primera versión de la página.

Luego, el sitio a elegir es de compra de música ya que uno de mis hobbies favoritos es escuchar música.

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

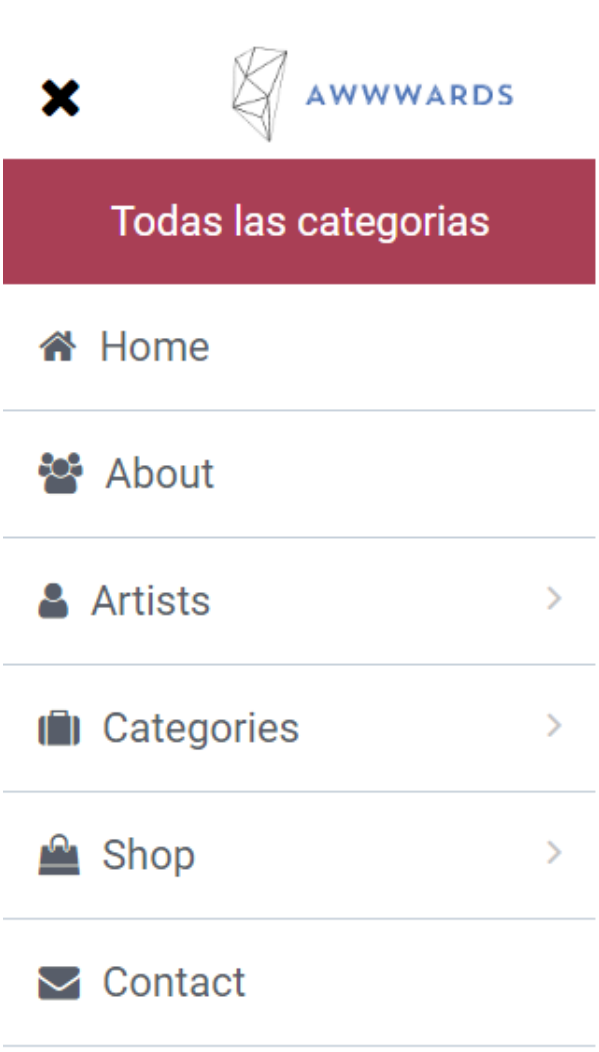
Segunda versión

En la segunda versión agregué un logo de prueba colocado en el header de la página y empecé a jugar con la paleta de colores de la misma. Para realizarla tomé el color verde utilizado www.awwwards.com y lo coloqué en el generador de paletas de colores www.coolors.co. Luego comencé a generar una paleta aleatoria dado ese color y me fui quedando con los colores que más me gustaban. Por último utilicé la hoja de estilos en cascada de Font-Awesome (http://fontawesome.io/icons/) para emplear íconos pre-hechos por esta página.



Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Tercera versión



En la tercera y última versión hasta el momento, utilicé todos los colores de mi paleta para crear una página más atractiva a la vista del ojo humano. Además, para tener un menú más lindo visualmente cambié las funciones hechas en JavaScript por una sola de jQuery. Para realizarla miré tutoriales completos de esta librería. Esta función se ejecuta sólo si el documento está listo y totalmente cargado, dentro de la misma se emplean 3 estructuras. La primera se ejecuta si se hace click sobre el botón de menú; luego si este botón tiene la clase de “fa fa-bars” (ícono por defecto del menú) se muestra el fondo transparente, se remueve esa clase y se agrega la clase “fa fa-close” (ícono de cierre) y por último a la barra de menú se le aplica la propiedad “left” de CSS 0px para mostrar el menú. Si esto no sucede (o sea el menú tiene la clase fa fa-close); se oculta el fondo transparente, se remueve la clase de cierre y agrega la de menú, se ocultan los submenús y se oculta el menú aplicando la propiedad “left” de CSS -320px. La segunda estructura es para mostrar el submenú, en esta se hace un selector de los elementos a que se encuentran dentro de la barra de menú y se le aplican una función onclick. Para hacer esto primero buscamos el valor del atributo “menu” y lo guardamos en una variable (este atributo dice que submenú es el que se quiere abrir). Después se hace un selector del submenú deseado, dependiendo de su atributo “menu” previamente guardado en una variable y a este selector se le aplica la propiedad “left” de CSS 0px. Por último para ocultar el submenú, se hace un selector del botón “Atrás”, que posee la clase go-back, que tiene como padre la lista no ordenada de clase submenu; y a este se le aplica una función onclick. La misma le aplica al padre del selector elegido la propiedad “left” de CSS -320px.

Con estas funcionalidades y con algunas propiedades de CSS aplicadas a los elementos de la navegación se llega al resultado mostrado.

Lo último agregado fue un slider de imágenes colocado en el index.html. Para realizarlo tomé una “base de código” previamente creado en un tutorial de Youtube. Luego de estudiarlo y agregarle nuevas funcionalidades quedó como muestra la imagen.

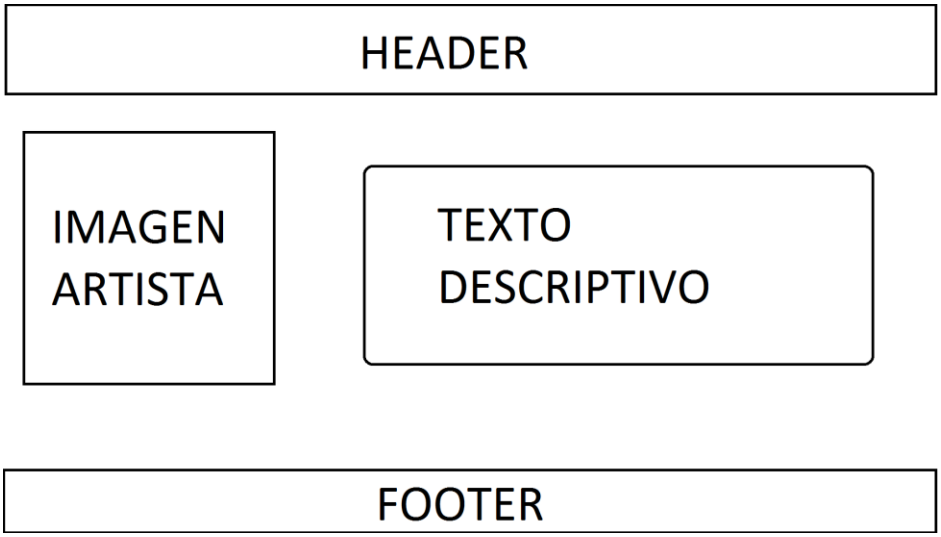
About

Por otro lado el sitio about.html fue creado con el objetivo de informar al lector sobre lo más primordial, en este caso un párrafo con información relevante sobre el contenido de la página y la imagen/imágenes de los creadores de la misma con sus respectivos párrafos introductorios.

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

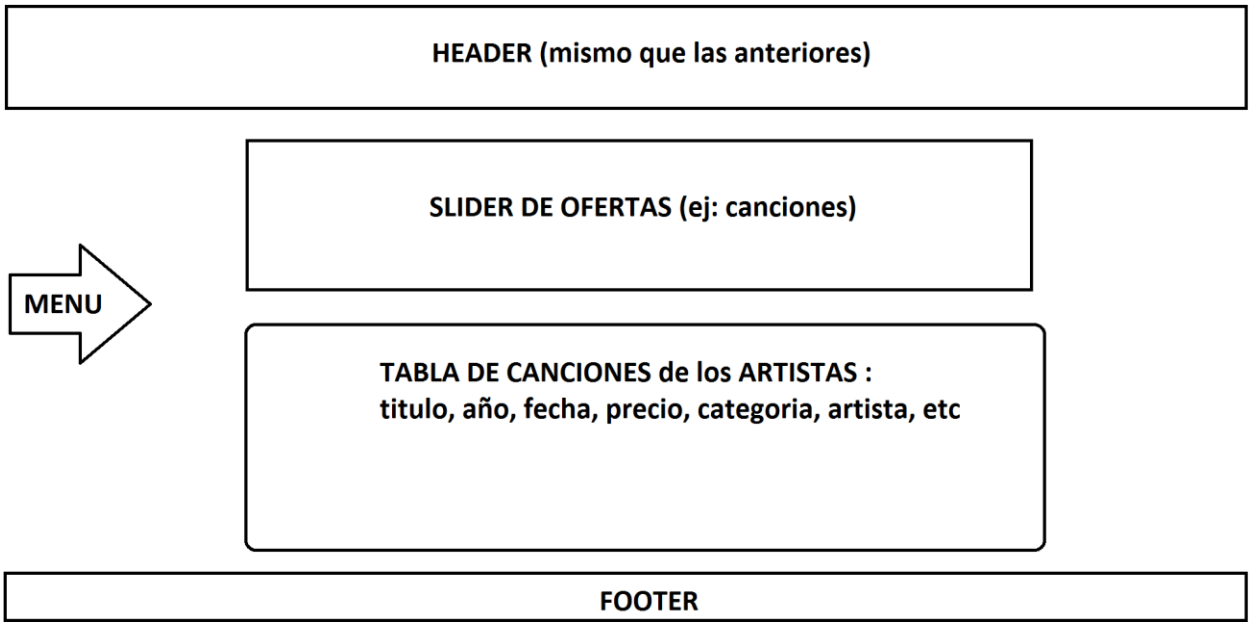
Artist

En el sitio artist.html (ejemplo: eminem.html) se tratará de dar una información general sobre ese artista e imágenes para acompañar al mismo.



Shop

Para finalizar en la página shop.html podrás encontrar en primer lugar las ofertas del día con las canciones más votadas y debajo una tabla de canciones de los artistas conteniendo un título, año, fecha, precio, categoría, artista, etc.



Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Primera Entrega Parte 2

Responsive

Para esta entrega al ya tener un diseño de página definido, me enfoqué en realizar las versiones para desktops, tablets y celulares de la misma. Esto me llevó por un largo proceso de ensayo y error ya que mi diseño fue realizado sin Bootstrap originalmente. De esta manera intenté incorporarlo a mi código pero nunca dio buenos resultados, debido a que mi barra de navegación de menú es una “sidebar menu”, la cual se despliega en pantalla por todo el alto de la misma; siempre resultando en algún error. Después de mucho tiempo de trabajo en vano, decidí no utilizar Bootstrap para hacer mi página responsive y optar por las media queries.

Por último esta funcionalidad hasta el momento solo está aplicada a la página de HOME.

Media queries

Celular 1: @media (min-width: 468px)

Celular 2: @media (min-width: 700px)

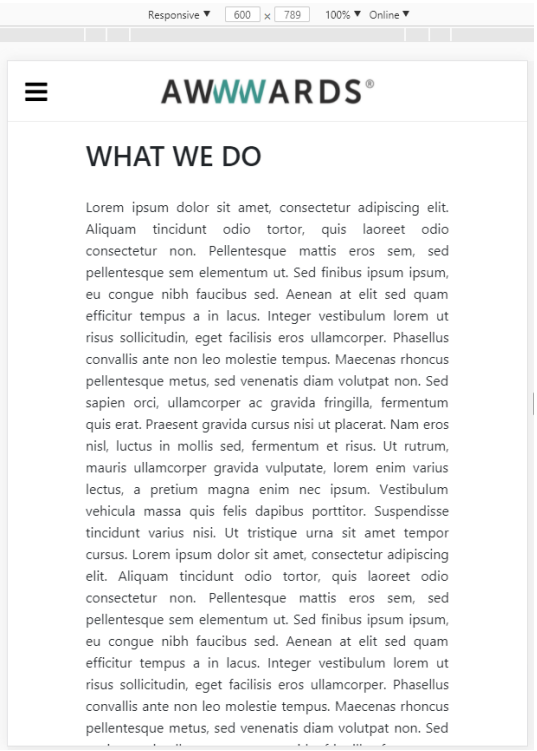
Tablet: @media (min-width: 768px)

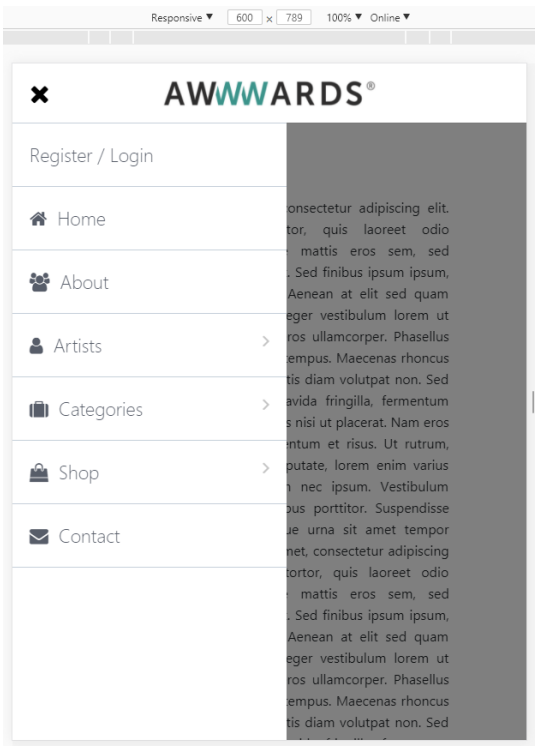
Desktop: @media (min-width: 992px)

Utilicé estos breakpoints para lograr una visualización lo más responsive posible hasta ahora. Luego agregaré más breakpoints dependiendo de cómo surge el flujo del contenido de la página.

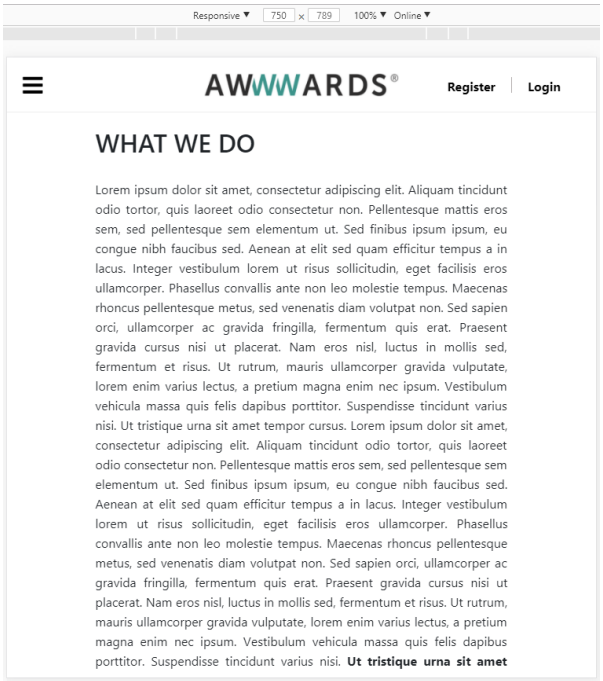
Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

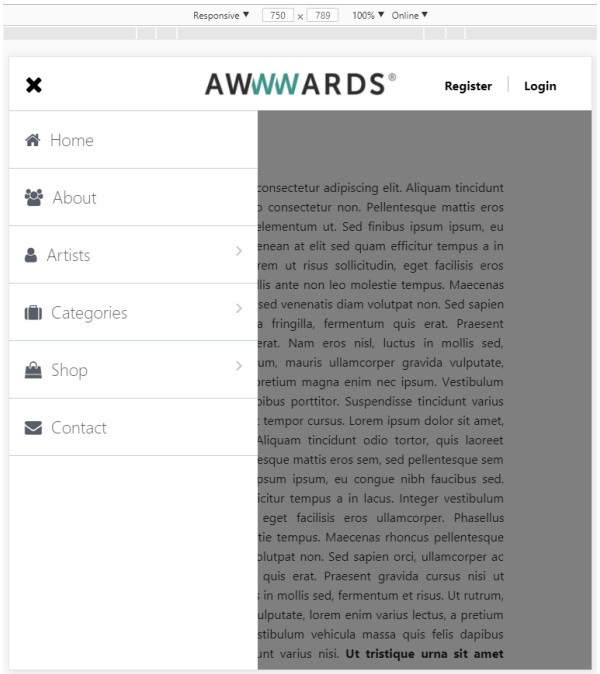
Celular 1



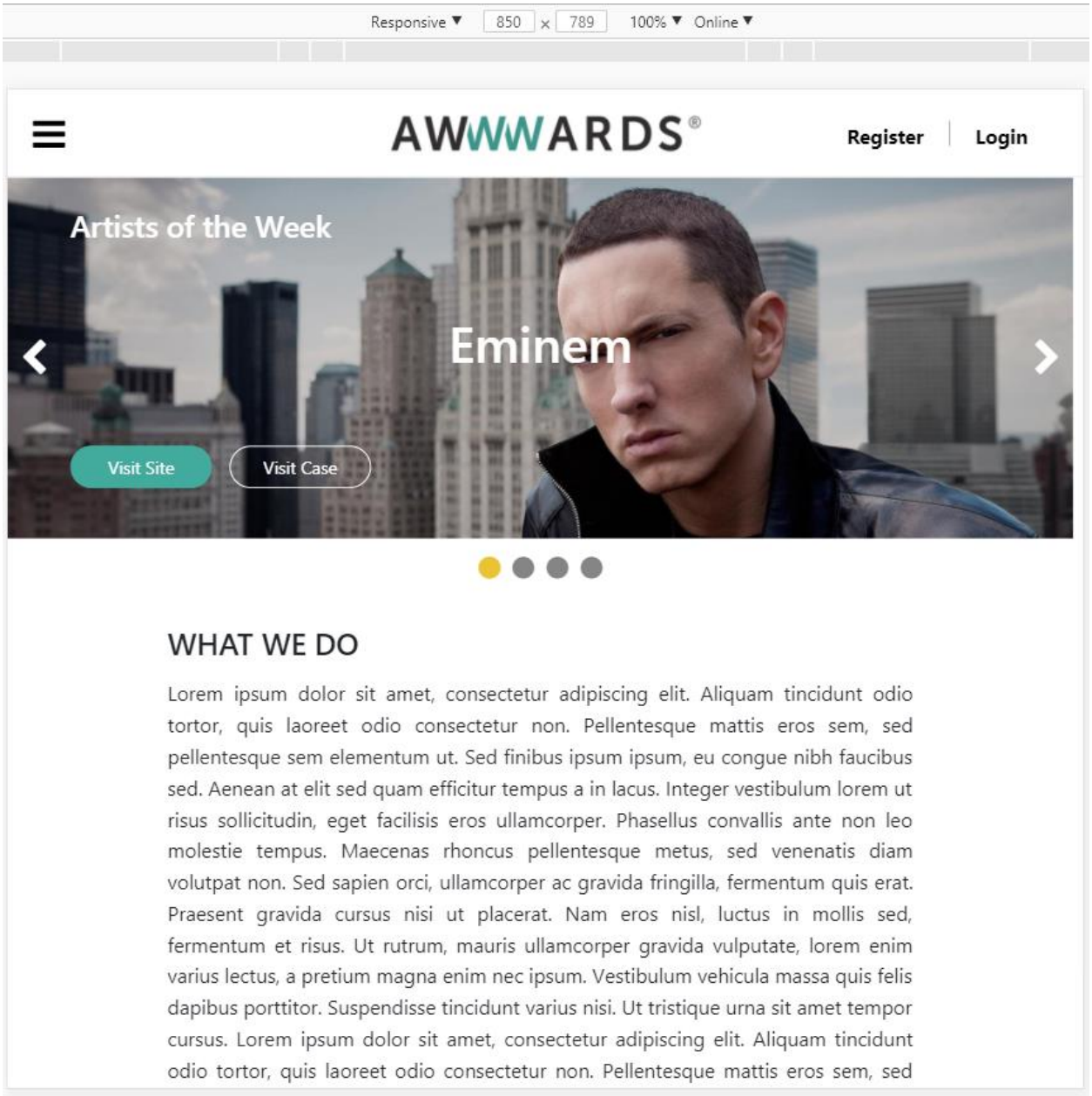


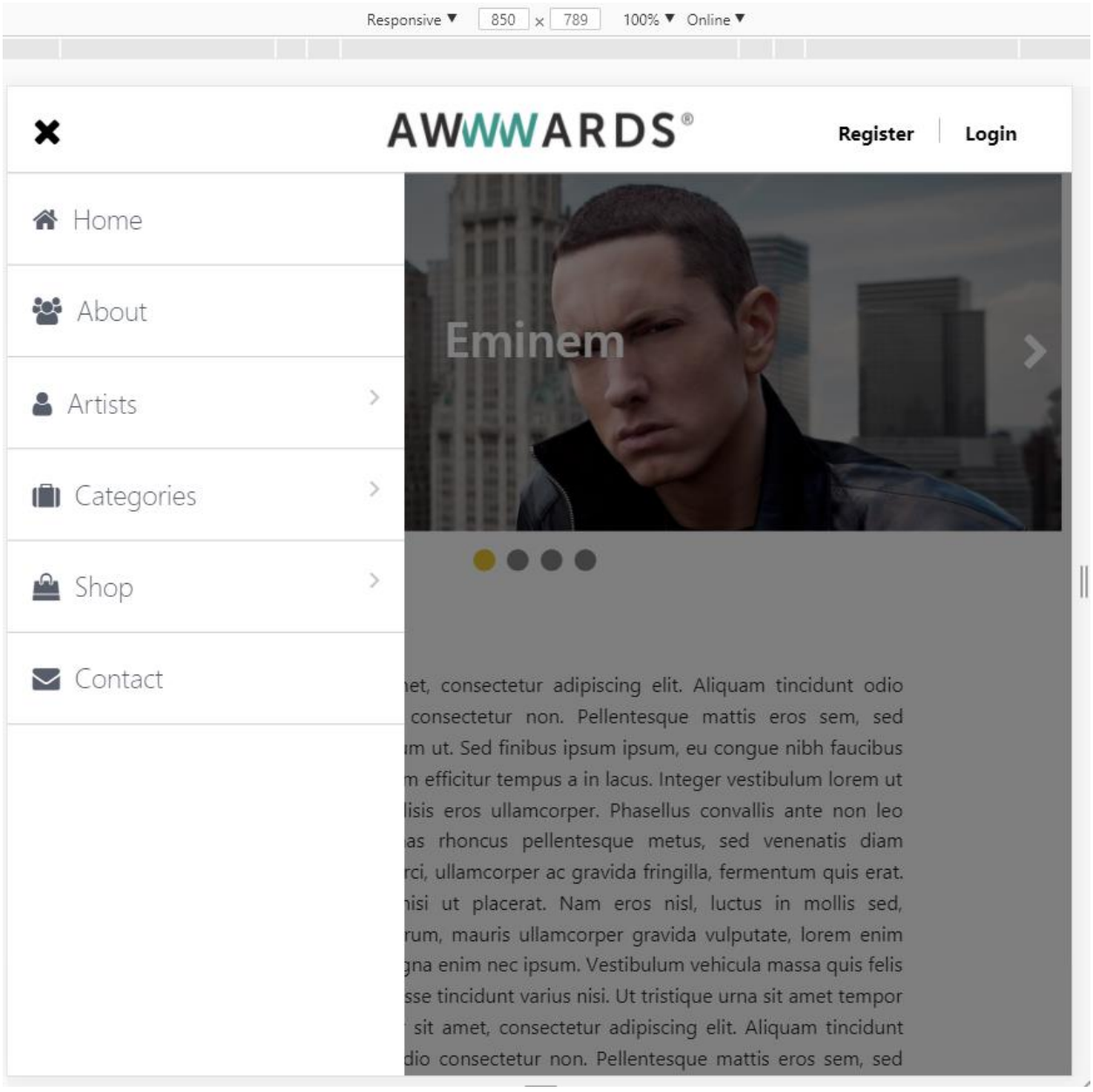
Celular 2





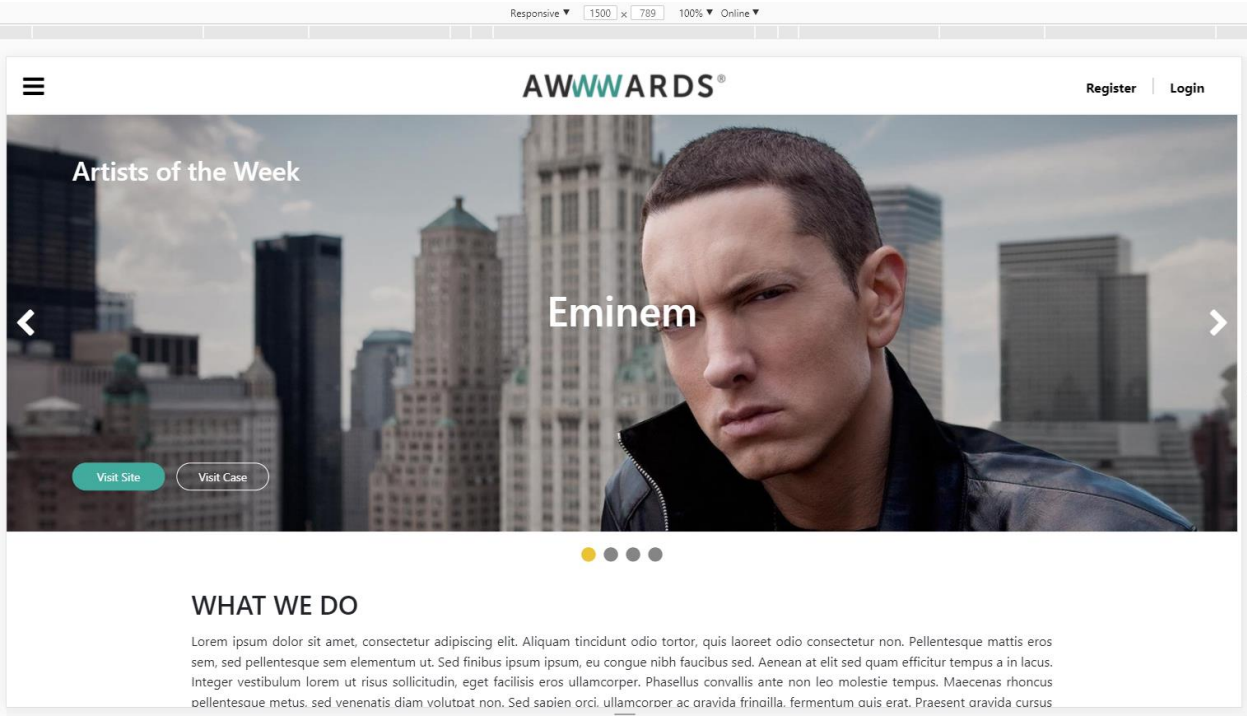
Tablet

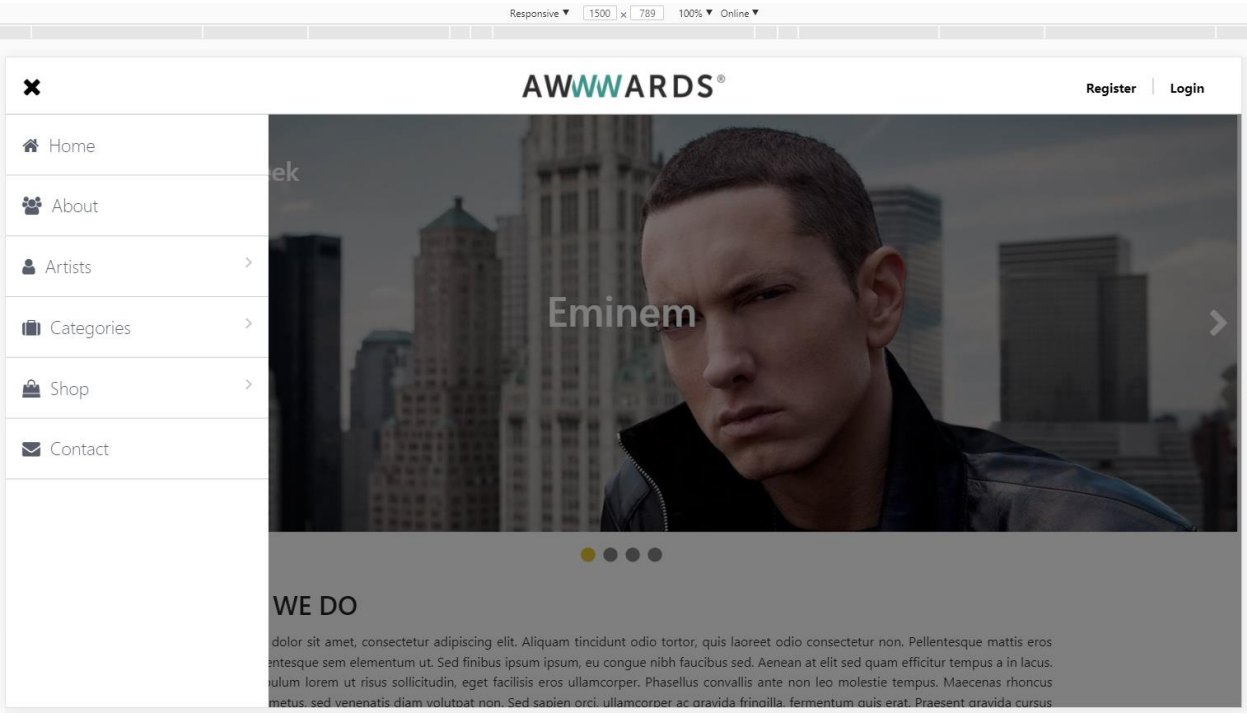

Responsive
850
789
100%
Online
AWWWARDS
Register
Login
Artists of the Week
Eminem
Visit Site
Visit Case
WHAT WE DO
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tincidunt odio tortor, quis laoreet odio consectetur non. Pellentesque mattis eros sem, sed pellentesque sem elementum ut. Sed finibus ipsum ipsum, eu congue nibh faucibus sed. Aenean at elit sed quam efficitur tempus a in lacus. Integer vestibulum lorem ut risus sollicitudin, eget facilisis eros ullamcorper. Phasellus convallis ante non leo molestie tempus. Maecenas rhoncus pellentesque metus, sed venenatis diam volutpat non. Sed sapien orci, ullamcorper ac gravida fringilla, fermentum quis erat. Praesent gravida cursus nisi ut placerat. Nam eros nisl, luctus in mollis sed, fermentum et risus. Ut rutrum, mauris ullamcorper gravida vulputate, lorem enim varius lectus, a pretium magna enim nec ipsum. Vestibulum vehicula massa quis felis dapibus porttitor. Suspendisse tincidunt varius nisi. Ut tristique urna sit amet tempor cursus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tincidunt odio tortor, quis laoreet odio consectetur non. Pellentesque mattis eros sem, sed


Responsive
850
789
100%
Online
AWWARDS
Register
Login
Home
About
Artists
Categories
Shop
Contact
Eminem
et, consectetur adipiscing elit. Aliquam tincidunt odio
consectetur non. Pellentesque mattis eros sem, sed
m ut. Sed finibus ipsum ipsum, eu congue nibh faucibus
m efficitur tempus a in lacus. Integer vestibulum lorem ut
isis eros ullamcorper. Phasellus convallis ante non leo
as rhoncus pellentesque metus, sed venenatis diam
rci, ullamcorper ac gravida fringilla, fermentum quis erat.
isi ut placerat. Nam eros nisl, luctus in mollis sed,
rum, mauris ullamcorper gravida vulputate, lorem enim
gna enim nec ipsum. Vestibulum vehicula massa quis felis
sse tincidunt varius nisi. Ut tristique urna sit amet tempor
sit amet, consectetur adipiscing elit. Aliquam tincidunt
dio consectetur non. Pellentesque mattis eros sem, sed

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Desktop



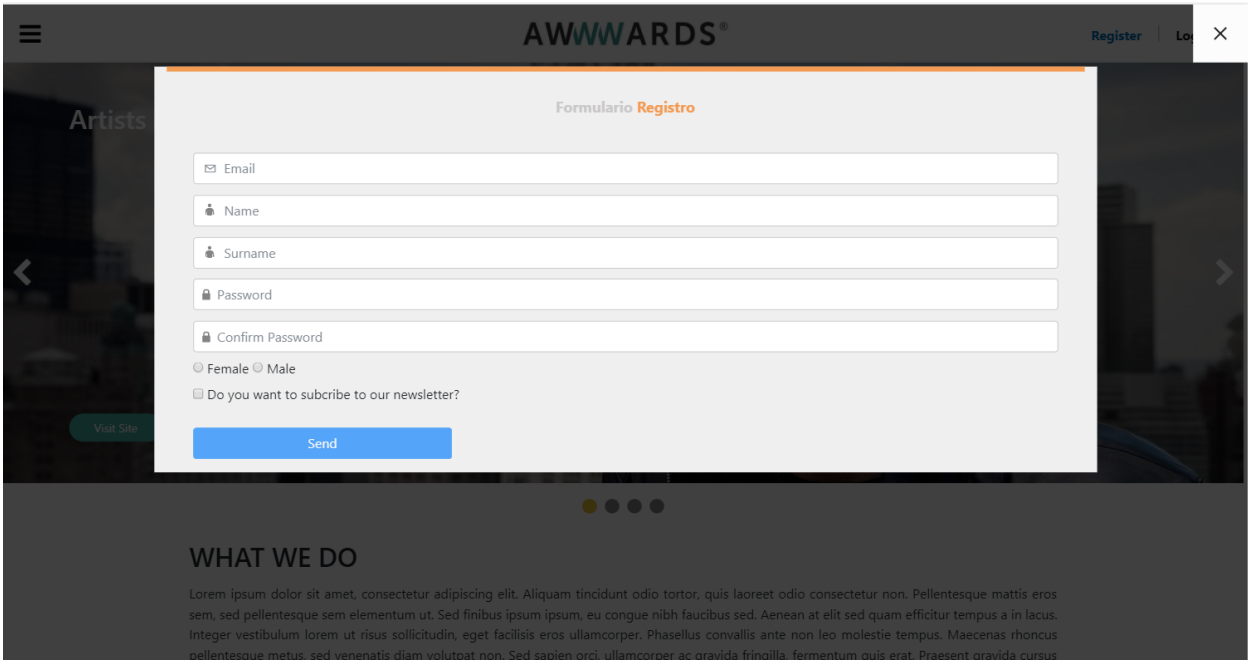


Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Formulario Registro

Angular

Hace bastante tiempo que ya vengo trabajando en el formulario de registro. Primero empecé a informarme sobre el tema de validación de los mismos y encontré que con la librería AngularJS de JavaScript podía realizar lo necesario para validar un formulario. De esta manera leí documentación sobre la validación de formularios con Angular y también vi video tutoriales sobre el tema. Sentí que con esta librería tenía todo lo necesario para realizar lo que quería, esa es la razón por la cual la utilicé.



Validaciones

El formulario sólo tiene las validaciones de ingreso pero no realiza ninguna función a la hora de tocar el botón Send. Estas validaciones son verificadas en el instante que el usuario empieza a ingresar datos en los campos de input y el botón Submit está deshabilitado hasta que todos los campos requeridos sean ingresados de forma satisfactoria (cumpliendo las restricciones ya especificadas en la letra de la segunda entrega).

La única validación que fue realizada sin AngularJS fue la de confirmación de contraseña ya que hasta el momento no tengo los conocimientos necesarios para realizarlo con este framework. Debido a esto, utilicé jQuery. Para la siguiente entrega una meta es poder hacer esta validación con Angular.

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Angular ng-messages

Las validaciones son realizadas con la directiva ng-messages de Angular, la cual está diseñada para mostrar y ocultar mensajes basados en el estado de una llave o valor de un objeto que lo está “escuchando”. La misma complementa mensajes de error reportándolos con el “ngModel $error object”, el cual guarda un estado de valor de las validaciones.

Ng-messages administra el estado de mensajes internos dentro de su elemento contenedor. Estos mensajes usan la directiva ng-messages y serán insertados o retirados de la página, dependiendo de si están presentes en el objeto clave/valor. Por defecto, sólo se mostrará un mensaje en un momento y esto depende de la prioridad de los mensajes dentro de la plantilla. (Esto puede cambiarse mediante el ng-mensajes-múltiples o varios atributos de la directiva del contenedor).

Ejemplo:

```
<form name="myForm">
  <label>
    Enter your name:
    <input type="text"
           name="myName"
           ng-model="name"
           ng-minlength="5"
           ng-maxlength="20"
           required />
  </label>
  <pre>myForm.myName.$error = {{ myForm.myName.$error | json }}</pre>

  <div ng-messages="myForm.myName.$error" style="color:maroon" role="alert">
    <div ng-message="required">You did not enter a field</div>
    <div ng-message="minlength">Your field is too short</div>
    <div ng-message="maxlength">Your field is too long</div>
  </div>
</form>
```

Esta es una estructura básica de un formulario, el cual valida si el valor ingresado en el campo de input del formulario tiene como mínimo 5 caracteres y como máximo 20, además de ser requerido. Para esto requerimos de ciertos atributos para algunos elementos del formulario. En primer lugar el formulario debe tener el atributo “name” con el nombre deseado para el mismo, luego en el input a llenar se debe poner el mismo atributo, el ng-model y por último los atributos para validar ese campo. En este ejemplo se utilizan los atributos ng-minlength=”5”, ng-maxlength=”20” y required. En segundo lugar, debajo del input a validar debemos crear un div con el atributo ng-messages, de forma tal que cumpla esta sintaxis:

“ng-messages=nombreDelFormulario.nombreDelInput.$error”. Dentro de este div irán todos los mensajes de error que refieran al input referido anteriormente. Estos div deberán tener un formato como el siguiente: “ Mensaje a poner en pantalla ”. Este increíble mecanismo funciona de la siguiente forma: En cada momento que el usuario cambia el valor del input deseado, el div que contiene el atributo de “ng-messages” verifica si este campo contiene algún error. Por una parte si el campo no contiene ningún error no se muestran mensajes. Por otra parte si el input contiene al menos un error, ng-messages mostrará el mensaje de error del div que corresponda con el error ingresado por el usuario, que esté en la jerarquía más alta.

Ejemplo de jerarquía de atributos ng-message:

```
<input class="form-control" id="email" type="email" placeholder="&#9993; Email"
name="email" ng-model="email" ng-required="true"/>


<div ng-messages="registerForm.email.$error" ng-if="registerForm.email.$touched">
        <small ng-message="required">You can't leave this field blank</small>
        <small ng-message="email">Enter a valid email</small>
</div>
```

En este ejemplo se realiza una validación de email con un campo de input y la verificación por parte de ng-messages. El input es de tipo email y es requerido, esas son las únicas dos validaciones para realizar. Con la condicional de ng-if=”registerForm.email.$touched” podemos saber si el usuario interactuó con el input, de esta manera los mensajes de error serán mostrados solo si el usuario “tocó” ese input.

Para explicar la jerarquía de los atributos ng-message debemos imaginarnos la situación en la que el usuario “toca” el input y “sale” del mismo sin haber ingresado ningún dato. En este caso se cumplen los dos errores de tipo ng-message, ya que el campo es requerido y el usuario no ingresó nada en el mismo, además de no cumplir el formato de “email” ya que valga la redundancia no ingresó nada. En este caso particular, dada la jerarquía superior del mensaje ng-message=”required” (esto se debe a que el mismo está primero que el otro), el mensaje de error mostrado en pantalla será el correspondiente con esta etiqueta. Luego, si el usuario ingresa cualquier carácter en el input sin cumplir el patrón de tipo email, el mensaje desplegado en pantalla será el referido al email.

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Última Entrega

Validación de Email

Para realizar la validación de email, en la que un usuario no puede registrarse con un email que ya existe en la base de datos, utilicé una estructura de código con un pedido Ajax de tipo GET. En la función on blur del input de email llamé a una función CheckMail, en la cual realizo una llamada Ajax en la que tomo como url la ruta de usuarios y como data el valor del campo de email. En el success de este pedido, llamo a la función procesar_usuarios pasando por parámetro el objeto datos. Dentro de esta función, en una estructura de tipo if, si el objeto datos devolvió algo muestro los errores necesarios en pantalla y declaro la variable userfound = true (esta servirá más adelante para realizar el POST del formulario). Por otra parte, si no devolvió nada se ocultan los errores de la pantalla, declaro la variable userfound = false y habilito el botón Submit.

Registro de Usuario

Para realizar el pedido Ajax de tipo POST el formulario tiene que haber aprobado todas las validaciones necesarias ya explicadas en la letra del proyecto. Mientras que el formulario no cumpla al menos una de estas validaciones, el botón submit del mismo estará deshabilitado. Por otro lado, si el mismo cumple con todo lo necesario estará habilitado. En la función on click del botón submit se llama a otra función sendForm. En primer lugar, solo si la variable userfound == false se ejecuta el pedido POST, sino se deshabilita el botón submit para que el formulario no sea enviado. En el caso de que la variable userfound sea igual a false, se crea el objeto user el cual contiene todos los datos ingresados por el usuario en el formulario de registro. Luego se realiza el pedido Ajax de tipo POST pasando por la url la ruta de usuarios y por data el objeto user ya declarado anteriormente. En la función success del mismo se cierra el formulario y se resetean los valores del mismo.

Login de Usuario

En el botón submit del formulario de Login se realiza una función para verificar si el email y contraseña ingresados coinciden con algún par de estos en la base de datos. En esta se realiza una función Ajax de tipo GET pasando por la url la ruta de usuarios y por data los valores de los campos de email y contraseña ingresados por el usuario. En la función de success se realiza una estructura for que sirve para recorrer los usuarios inscriptos y verifica si coincide el email y

contraseña de alguno de estos con lo que ingresó el usuario. En el caso de no encontrar ningún usuario que cumpla esos requisitos se muestran los errores necesarios en pantalla. Por otro lado si estos coinciden se ocultan estos errores, se guardan en el sessionStorage algunos datos del usuario y se llama a la función user_login. En esta, se cierra y resetea el formulario, se muestra el carrito de compras y se crea un div al lado del carrito que muestra un logo y el nombre del usuario logeado (obtenido por el sessionStorage).

Cargado de discos

Para el cargado automático de discos se realiza un pedido Ajax de tipo GET de todo el contenido de la ruta http://localhost:3000/discos. En la función success de este pedido se utiliza la estructura forEach para recorrer todo el contenido del objeto datos, el cual contiene todos los discos de la ruta indicada anteriormente; por cada disco encontrado se llamará a la función crear_articulo la cual crea la estructura para cada canción. En esta función crear_articulo, por medio de jQuery se crea una etiqueta article con la clase card, la imagen, nombre y el botón comprar para cada disco encontrado. Por último en la función comprar, se muestra por consola el id del disco comprado.

Errores Obtenidos / Funcionalidades sin terminar

Sin el poder utilizar el último día de entrega y la falta de tiempo por mi parte, me fue imposible realizar la funcionalidad del carrito de compras. De todas maneras, esta funcionalidad la sé realizar. Por otro lado, tuve bastantes problemas con las hojas de estilo en cascada de la página principal (index) ya que no utilicé Bootstrap desde un principio y en el momento de querer cambiar todos mis estilos con esta librería se creaban más problemas de los ya existentes en la página. Además, en esta última versión al agregar los formularios de registro y Login, ciertos errores o divs pueden aparecer en demás versiones que no sean la de desktop cuando no deberían mostrarse. Recomiendo realizar las funciones de los mismos teniendo la página en la máxima resolución disponible. Para poder utilizar la página necesitarás de conexión a Internet ya que todas las librerías utilizadas son integradas a la página por links web.

Por último me gustaría poder presentar y/o entregar otra versión de la página intentando agregar y arreglar los errores obtenidos anteriormente.

Matías Salazar
Grupo 26 – Desarrollo Web 11/10/2017

Repositorio de GitHub

- https://github.com/matiassalazar/JaP-Grupo-26---Proyecto-Final