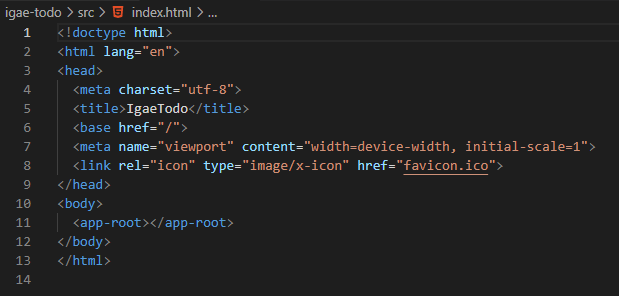
COMPONENTES

Son las piezas fundamentales de las aplicaciones en Angular, que nos trae diversos beneficios que mejoran sensiblemente la forma de organización de una aplicación, su mantenimiento, reutilización del código. Equivalen a etiquetas de HTML nuevas que podemos inventarnos para realizar las funciones que sean necesarias para nuestro negocio.

COMPONENTE INICIAL DE NUESTRA APLICACIÓN – APP.COMPONENT

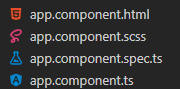
Si observamos el index.HTML, vemos la siguiente etiqueta:

(img1)



<app-root></app-root> no es una etiqueta estándar de html5. Este es el primer componente de nuestra aplicación, y lo ha generado Angular de forma automática al crear el proyecto con ng new archetype-mrr. Su estructura es la siguiente:

(img2)



Todo componente en Angular consta de 4 ficheros, con primer apellido: .component:

*1)app.component.html*: Equivale a lo que conocemos por "vista" en la arquitectura MVC.

*2)app.component.css*: Permite colocar estilos al contenido, siempre que éstos estén encapsulados en este componente.

*3)app.component.ts*: Es el archivo TypeScript, que se traducirá a Javascript antes de entregarse al navegador. Sería el equivalente al controlador en el MVC.

*4)app.component.spec.ts*: Un archivo TypeScript destinado a tareas de testing de componentes.

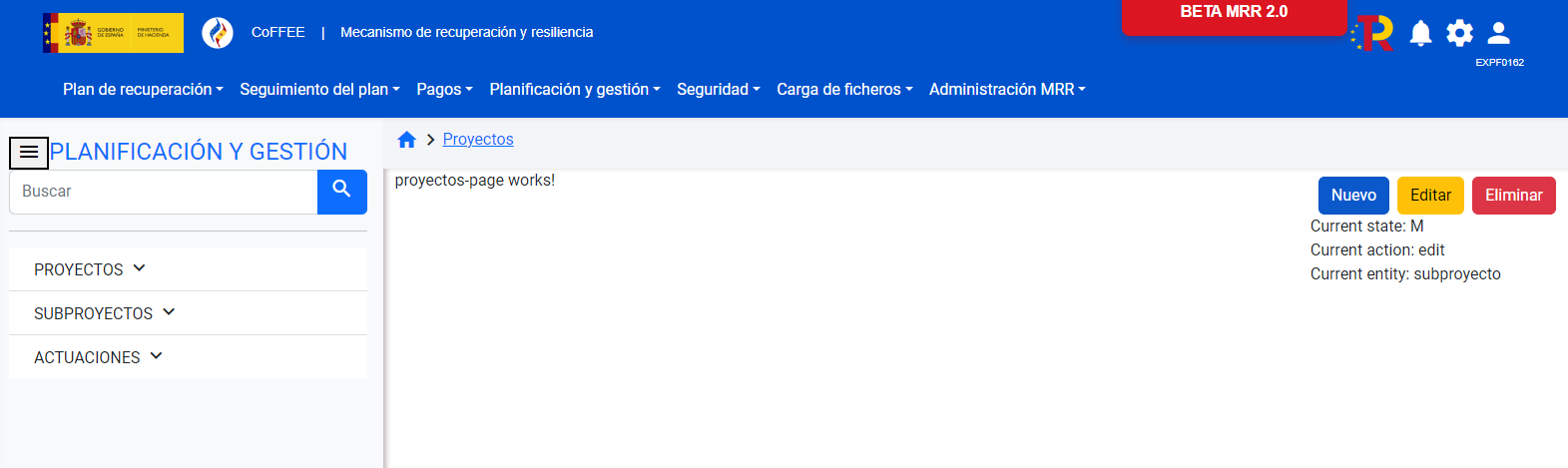
CONTENIDO DEL COMPONENTE:

Si abrimos el archivo app.component.html veremos el código de la vista generada por defecto al lanzar el comando ng new. Vamos a proceder a borrar la vista y dejarla vacía.

Nuestro proyecto será un prototipo de vista, parecido al que tenemos en COFFEE-MRR.

Al final de estas prácticas, el aspecto deberá ser parecido a esto:

(img3)



Iremos paso por paso hasta llegar a este modelo.

Para empezar, vamos a ver como crear y componetizar adecuadamente una aplicación. En angular, la creación de componentes se hace mediante el comando:

*ng g component nombreDeMiComponente (*g es el alias de GENERATE).

Al poner el nombre del componente en CamelCase. Angular creará una carpeta con el nombre seleccionado, separado por guiones: nombre\_de\_mi\_componente, en la cuál estarán los 4 archivos mencionados anteriormente.

Para nuestros primeros pasos, vamos a crear, en función del ejemplo de vista final que tenemos arriba, 3 componentes principales:

1)Un menú de navegación

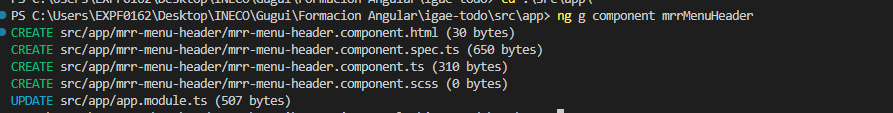
2)Un Sidebar, o menú lateral de opciones  
3)Un contenedor principal, donde mostraremos las distintas partes de la aplicación, en función de las opciones que hayamos puesto en los menús.

Además de estos 3 componentes, añadiremos un componente footer, que es necesario en toda aplicación empresarial.

Con estos componentes, tendremos la estructura general de la aplicación. Vamos a ello ☺:

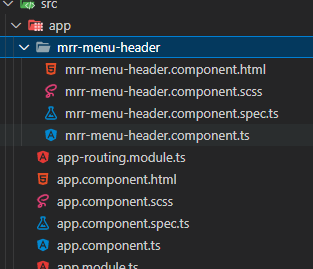
Para empezar, vamos a generar el primer componente de los 4, el menú de navegación. Vamos a llamarlo mrr-menu-header. Para crearlo, navegamos a la carpeta src/app/ y lanzamos en el terminal el siguiente comando: *ng g component mrrMenuHeader*

(img4)



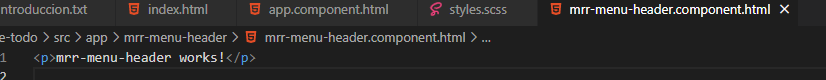
En el proyecto veremos que se ha creado una carpeta de este estilo:

(img5)



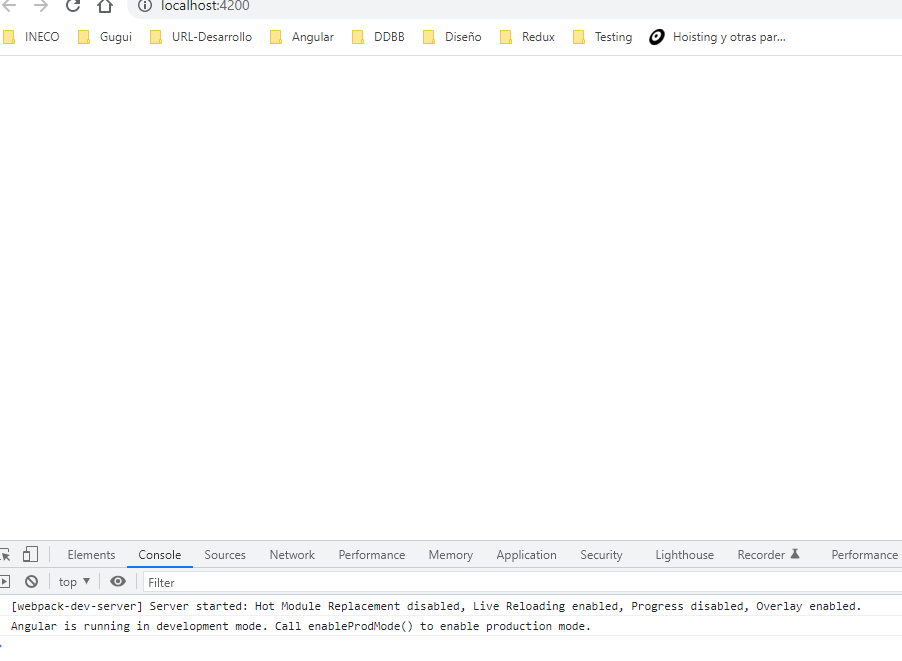
Con los 4 archivos comentados anteriormente. En el archivos .scss , el de estilos, no hay nada por defecto. En el .html, Angular por defecto siempre crea un párrafo con el nombre del componente, seguido de It Works!

(img6)



Esto es para que la vista del componente tenga algo, para poderla identificar en pantalla. Lo que ocurre es que, si volvemos a levantar el proyecto, nos encontramos la pantalla vacía:

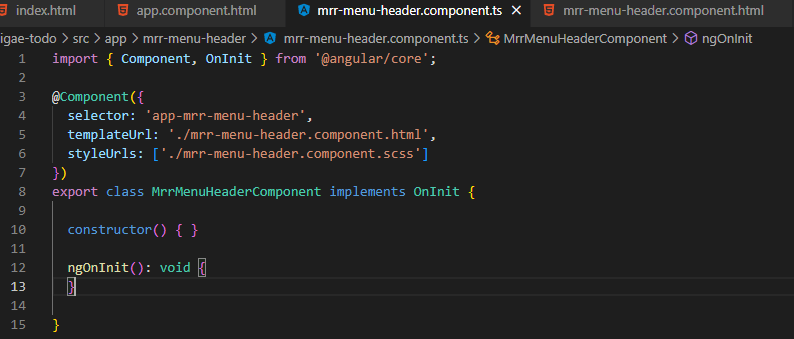
(img7)



Esto es porque nuestro app.component.html está vacío. Lo qué si podemos apreciar por consola, es que Angular está levantado y funcionando.

Para poder visualizar nuestro nuevo componente, debemos inyectarlo en alguna vista, para que la aplicación pueda servirlo. Para ello utilizamos el SELECTOR del componente. Si entramos en el archivo .ts del componente, nos encontramos el siguiente código, que Angular genera automáticamente:

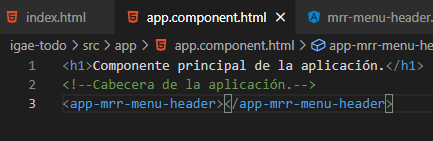
(img8)



* Imports de todo aquello que necesitemos. En principio de la librería @angular/core, pero luego veremos que aquí iremos colocando muchos otros imports a medida que vayamos necesitando código de más lugares.
* @Component - Decorador del componente, para su registro, contiene metadatos, como las anotaciones de Java.
* Clase que hace las veces del controlador, que tengo que exportar.

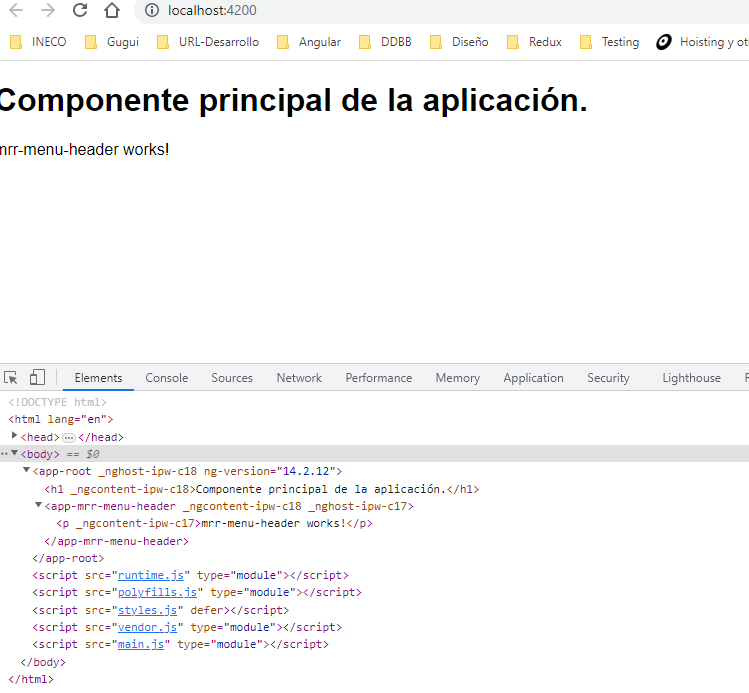
Ahora vamos a inyectar el selector del component en nuestro app.component.html, como una etiqueta html clásica, de esta forma:

(img9)



Hemos añadido una etiqueta H1, para poner diferenciar el código que solo pertenece al mrr-header-component, y el código que además presenta el appp.component. Ahora la apliación muestra lo siguiente:

(img10)



Vemos como se han pintado tanto la etiqueta h1 del app.component, como el párrafo del mrr-menu-header. Si nos fijamos en la pestaña elements de la consola del navegador, vemos como en el árbol DOM de la aplicación, se han creado el <app-root> y, en el interior de este, el <app-mrr-menu-header>, que son los selectores de mis 2 componentes.

Con esto ya hemos visto como crear e inyectar un componente en una aplicación Angular. Ahora vamos a crear los otros 3 componentes principales que van a conformar nuestra aplicación, y a inyectarlos igualmente.

Lanzaremos los comandos:

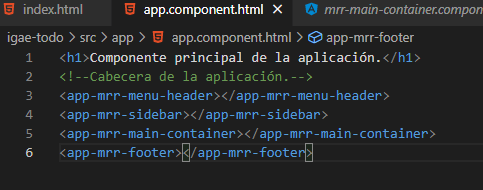
*ng g component mrrSidebar*

*ng g component mrrMainContainer*

*ng g component mrrFooter*

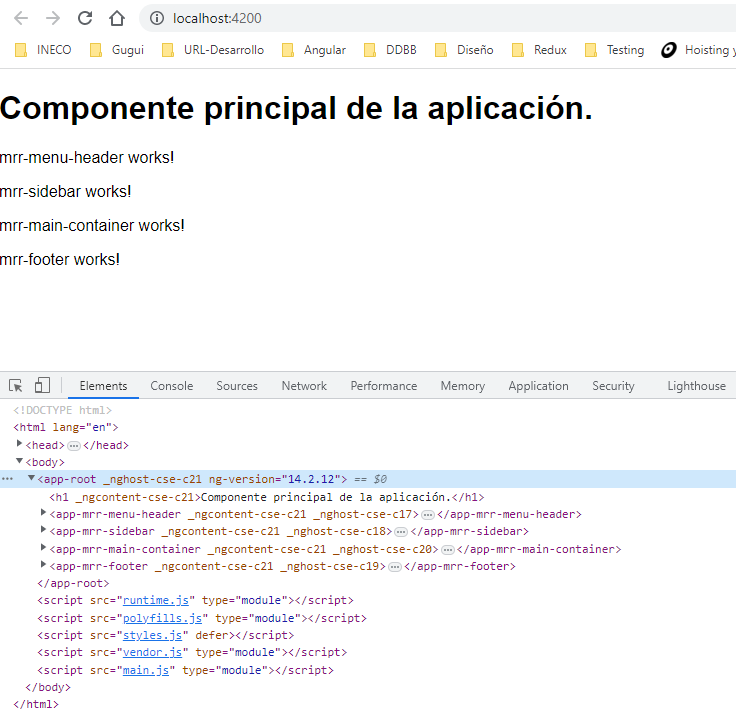
Una vez creados, procedemos a inyectarlos en nuestro componente principal, al igual que hicimos con el menú-header:

(img11)



Nuestra vista principal quedará de la siguiente manera:

(img12)



Comprobamos que los 4 componentes están inyectados correctamente, y están añadidos a la estructura del árbol DOM. Pero el aspecto de la página dista mucho de lo que tenemos en el ejemplo de la (img3), así que vamos a dar estilos a los distintos componentes, a través de sus archivos .scss y .html, para que el esqueleto de nuestra aplicación tenga la estructura visual que queremos.

1)Vamos a eliminar la etiqueta <h1> que pusimos en el app.component para explicar la diferencia con las etiquetas personalizadas. Añadiremos el siguiente código, para empezar a estructurar la vista principal de la página. Para ello nos vamos a basar en la librería bootstrap, que se utiliza en el MRR. Nuestro app.component.html quedaría de la siguiente manera:

<div class="container-fluid">

    <div class="row">

        <div class="col col-lg-12 col-md-12">

            <app-mrr-menu-header></app-mrr-menu-header>

        </div>

    </div>

    <main class="pagina-container">

        <div class="row">

            <div class="col col-md-3 col-lg-3 col-sm-3 sidebar-container">

                <app-mrr-sidebar></app-mrr-sidebar>

            </div>

            <div class="col col-md-9 col-lg-9 col-sm-9 info-container">

                <app-mrr-main-container></app-mrr-main-container>

            </div>

        </div>

        <div class="row footer-container">

            <app-mrr-footer></app-mrr-footer>

        </div>

    </main>

</div>

Y el app.component.scss:

.pagina-container {

    background-color: #f4f5f9;

    box-shadow: inset 5px 1px 5px -2px #c3b8b8;

    min-height: calc(100vh - 106px);

}

.sidebar-container {

    min-height: calc(100vh - 106px);

    padding-right: 0%;

    border-right: 1px solid black;

}

.info-container {

    min-height: calc(100vh - 106px);

    padding-left: 0;

}

.footer-container {

   border-top: 1px solid black;

}

2)En el mrr-menu-header.html cambiamos la etiqueta <p> por <header>, dado que nuestro componente va a ser una cabecera:

<header>mrr-menu-header works!</header>

Y en el mrr-menu-header.scss le daremos estilo y dimensiones:

header {

    background-color: #0052cc;

    min-height: 106px;

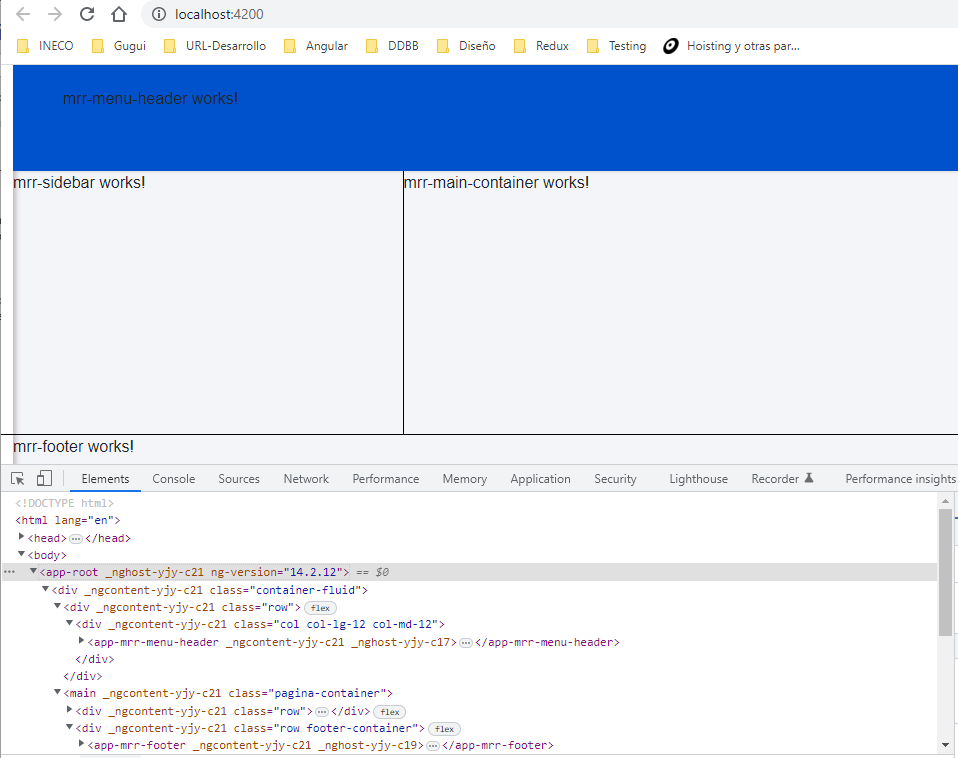
    padding-top: 22px;

    padding-left: 50px;

}

Con todo esto, tendríamos una primera disposición de la vista principal:

(img13)



En la siguiente práctica, aprenderemos a dar funcionalidad a nuestros componentes, y a que hablen entre ellos.