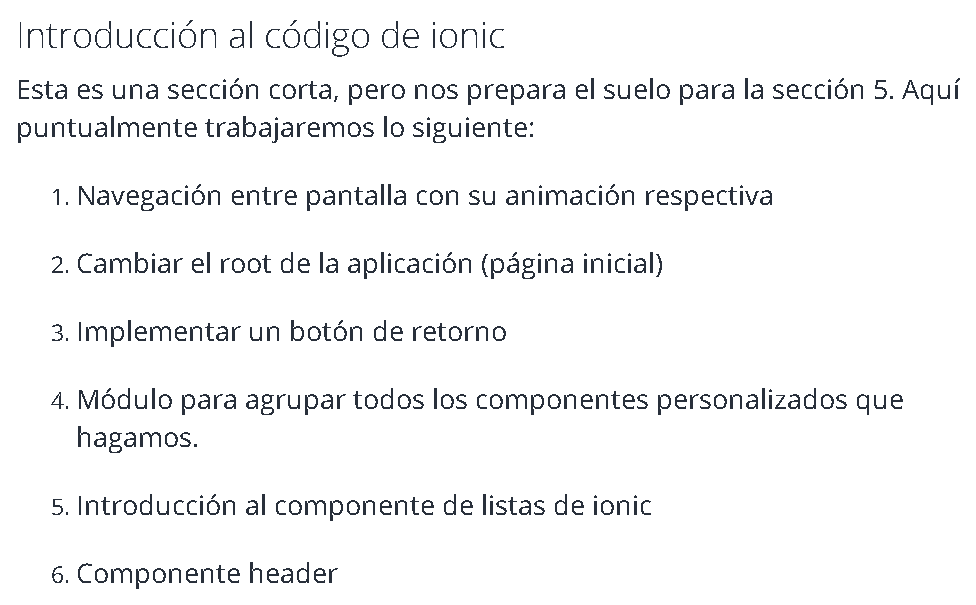
**Bases Ionic**

**Capitulo 1**



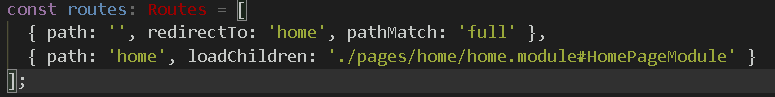
Para crear una aplicación



En el proyecto borramos la carpeta home que se crea por defecto, para crear nuestra propia pagina de inicio.



Crea la pagina en la ruta y actualiza automáticamente el app.routing.module con la ruta hacia este, con lazy-load.

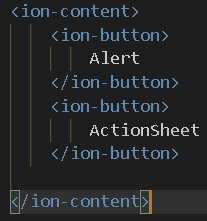


**Navegación entre paginas**

Creando paginas:

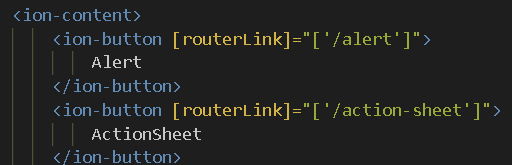
Se crean dentro de pages : alert y action sheet  


Creamos dentro del home 2 botones para poder navegar entre los nuevos componentes creados



Para añadirle la navegación

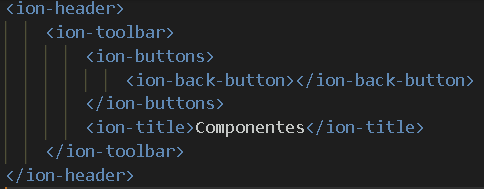
Se puede manejar de igual forma como en angular con el routerlink y el nombre ya agregado a la lista del modulo de routing.



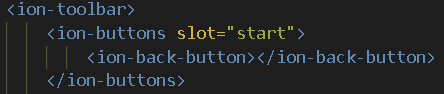
**BackButton**

Con esto se puede navegar, pero necesitamos agregar un botón de back para poder regresar al componente anterior.

Dentro del header -> toolbar de cada pagina agregamos un ion buttons y dentro un ion-back button, el cual viene programado para regresar a la pagina anterior.



Para posicionar los botones agregados en el toolbar se le agrega la propiedad slot y la posición



Y para que cargue automáticamente esta propiedad





Podemos personalizar la palabra que tenga este botón con la propiedad text



**Header**

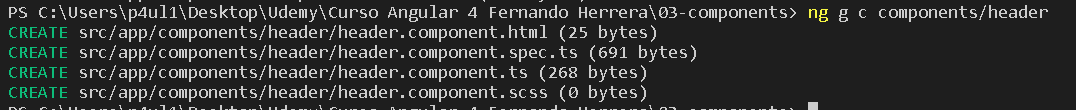
Dentro de app se crea una carpeta de components y dentro un modulo que va a manejar todos los componentes de la aplicación. Esto para no duplicar código por ejemplo con el header y solo hacer el llamado a este con su tag y personalizando un titulo.

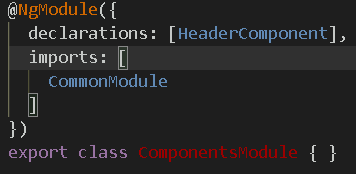


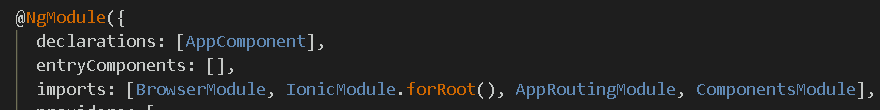
Para luego crear el componente del header



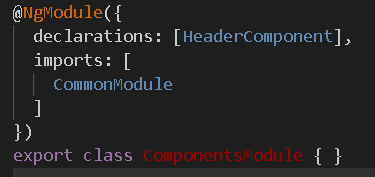
Verificar si no se modificó el module de componentes, ya que este componente debe estar declarado ahí para poder ser utilizado. Si no está, agregarlo manualmente.





Recordar que hay que añadir el nuevo modulo que contiene el header, al app.module en los imports.Esto va a lanzar un error ya que como todo esta cargado de forma LazyLoad, el header no ha sido cargado. Es necesario cargarlo en el modulo de la pagina donde se vaya a usar, en este caso y momento: alert y action sheet.

Ahora falta una configuración, si vemos el components.module, nos hace falta decirle que esta configuración va a poder ser usada fuera de este modulo.





Verificar también que este modulo de components tenga el IonicModule en los imports, ya que el código que tengan los componentes seguramente tendrá nomenclaturas de ionic.



Funciona correctamente el header después de pegarle el código que tenían los headers individuales. Ahora falta hacer dinámico el titulo pasandolo como un input, para que reciba este valor desde el componente que lo llama.

En el header.component.html





Y en los componentes donde se usa el tag se le pasa el valor:

Ejemplo AlertComponent  

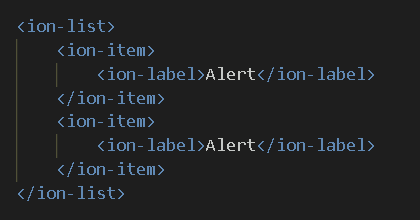

También se le puede añadir la propiedad al header de no border para que no tenga borde de sombra.





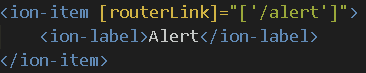
**ION-LIST**

Es una lista de elementos parecida a angular con la estructura





Al cambiar la estructura poniendole el router link al elemento redirecciona a la pagina que apuntemos y pone una flecha al final del ítem para hacer notar que es un elemento que redirecciona a otro lugar, pero esto solo sucede en iphone. Para que aparezca en Android hay que agregarle otra propiedad al ítem “detail”.



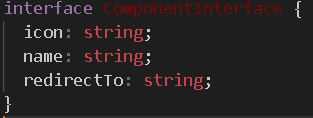




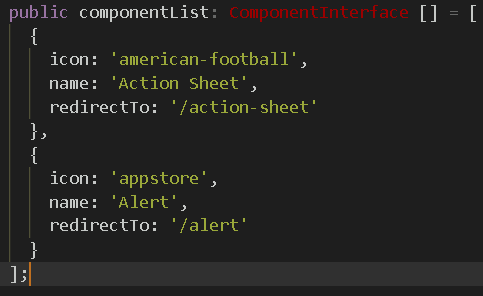
**Automatizando la lista**

Ahora para hacer la lista de los componentes que tenemos, no debe hacerse manual, si no creando un arreglo de componentes para recorrer y se muestre en una lista desde el [home.page.ts](http://home.page.ts)

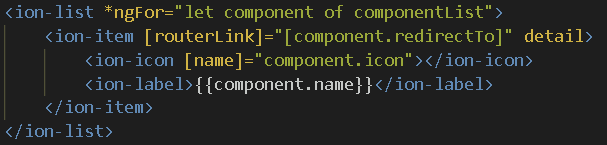
Para esto creamos una interfaz dentro del [home.page.ts](http://home.page.ts) que va a ser de tipo componente.

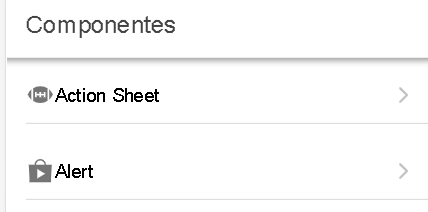


Y dentro de la clase del home creamos el arreglo de este tipo con la informacion de los componentes creados.



Ahora dentro del html podemos iterar en base a este arreglo





Arreglando los iconos, podemos definir su posición con la propiedad que usamos anteriormente “slot”.



