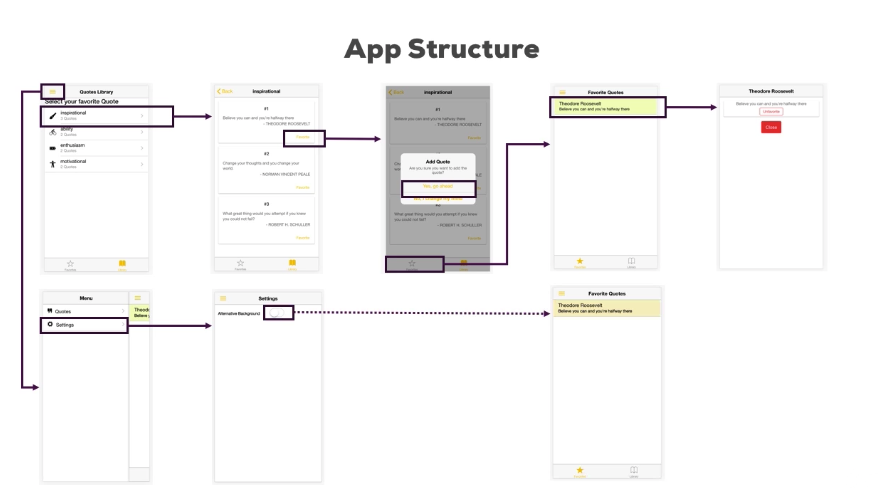
Aplicación de quotes favoritos.



Paginas que se necesitaran:

-Library

-Quotes

-Favorites

-Modal

-Settings

ionic start app-quotes blank

ionic g page …

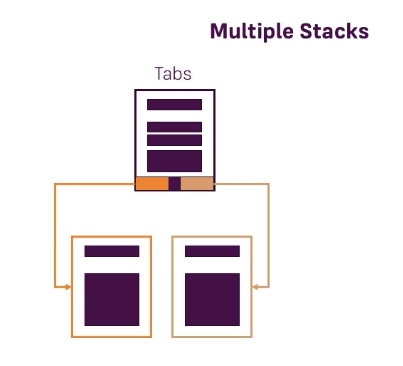
Se borra la carpeta home para luego arreglar las dependencias y tener otra ruta inicial

Esto en el app component y en el app module

Construyendo el tab navigation

Teoria de Multiple Stack

Al tener un menú de 2 opciones debajo , al hacer click en una y luego en otra no queda acumulado la pagina inicial, A y luego la B, si no que alterna entre, pagina inicial y A…y luego pagina inicial y B.



Añadiendo tabs a la aplicación:

Se crea la pagina tabs

La cual tiene la siguiente estructura en el html, no tiene la estructura normal de una pagina ya que solo se espera que este en la parte de abajo y no ocupe el espacio completo de la pantalla.

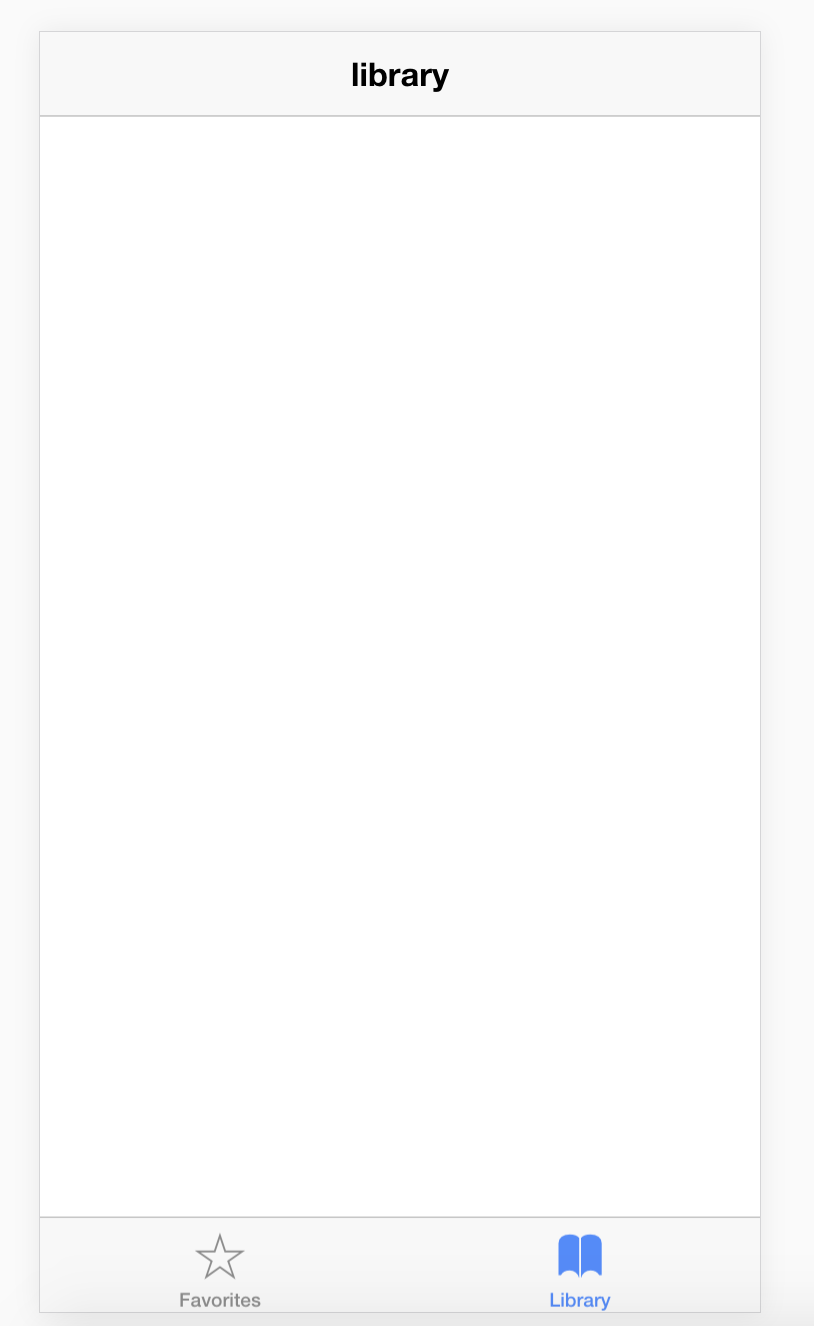


y en el componente una variable que hace referencia hacia la pagina.



El tema de crear un tab page es que al ocupar el espacio de abajo , deja en la parte de arriba libre para al hacer click en un enlace, este espacio se llene con la nueva pagina llamada, siendo siempre la principal el tabs page.

Ahora hay que setear la pagina de tabs como root.



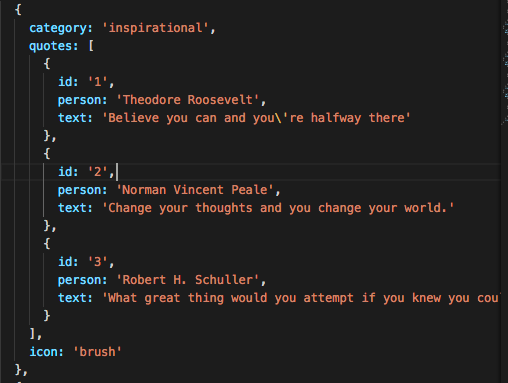
Añadiendo los datos de quotes al app.

Vienen de un archivo y como va a ser utilizado por varios componentes, se crea una carpeta en el src para usarlo, llamada data



en donde existe

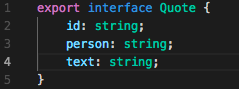
categoría, arreglo de quotes, y un icono.



Utilizando los datos:

En library:

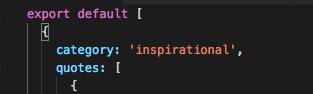
Creamos una variable quotesCollection. Para asignarle el tipo se crea una interfaz en la carpeta de data, que tendrá el modelo que puede recibir este tipo de dato.



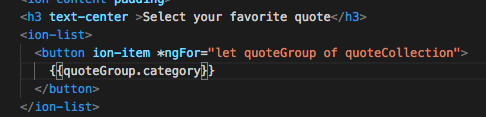
Ahora se crea la variable quotsCollection del tipo

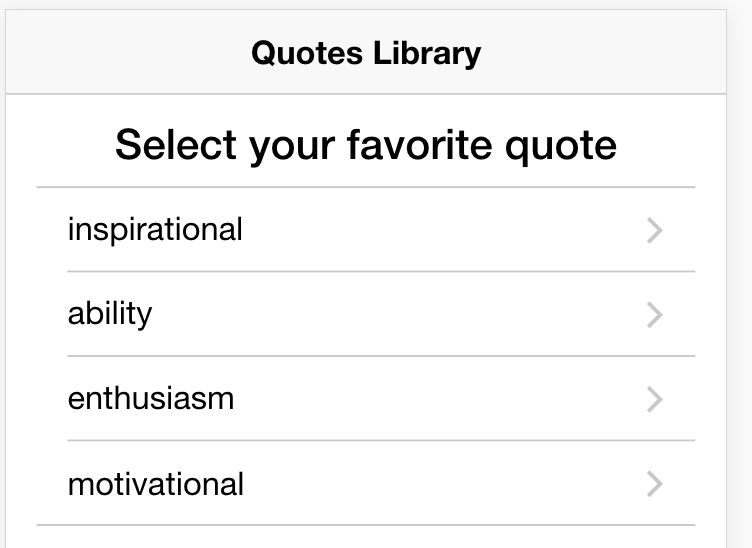
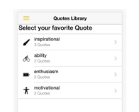


Para asignar el valor a esta variable, como es estatica la información y se esta exportando la data de forma default:

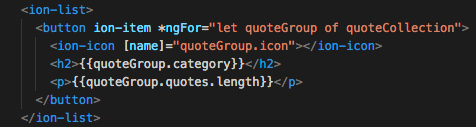


Ahora pasamos a desplegar los datos:

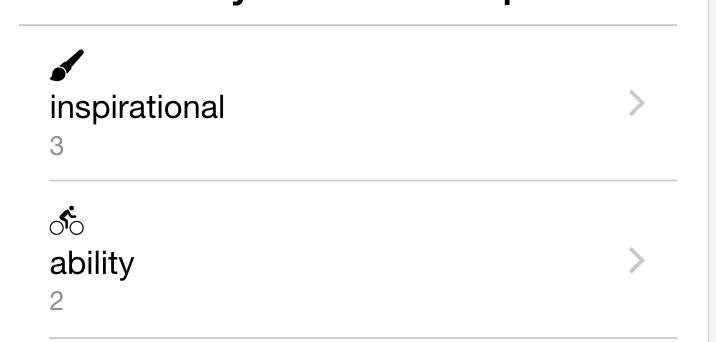


 (Pantalla deseada)

Añadiendo iconos:



que da como resultado:



Mejorandolo:

Para manipular el comportamiento dentro de un ion-item

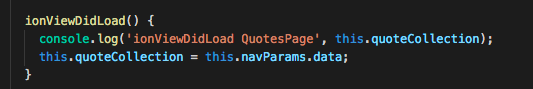
Se puede usar ítem-left



Ahora vamos a pasar los datos de la categoría seleccionada a la siguiente pagina que seria quotes.



Y recibiendo los datos en la otra pagina

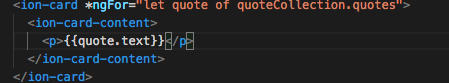


Al poner la variable que se mostrara en el html dentro del ioviewDidLoad, esta tendrá valior luego de que ya se haya pintado el html. Lo cual lanzara un error, para solucionarlo en el lugar donde se pinte se pone un ? para que pregunte si existe o hasta que exista se muestre.

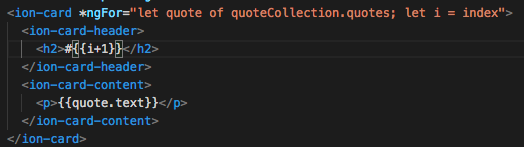
Ya teniendo los datos a mostrar, en el html vamos a utilizar los Cards.

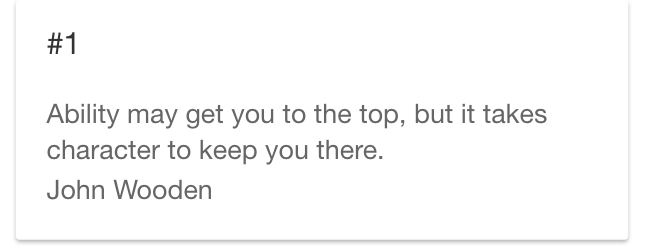
Repitiendo cada card por cada elemento del arreglo que tenga quotes





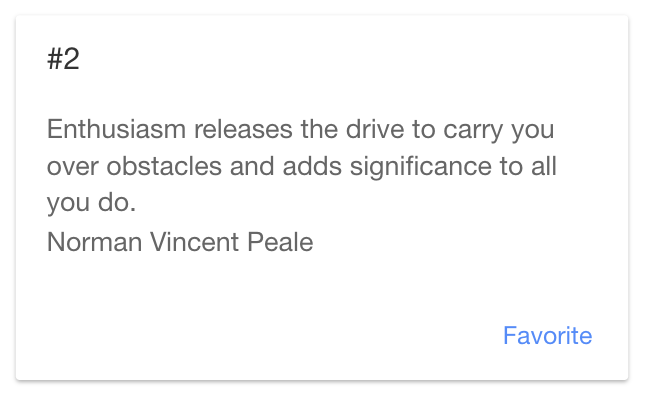
Añadiendole la estructura de ion-card-content, se presenta de una mejor forma.

ç

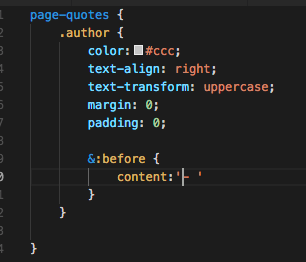


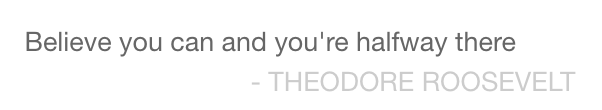
añadiendo un botón para añadir a favorito con ionic grid:





Estilo custom en el css para el autor, teniendo una clase creada en el scss podemos poner mas propiedades dentro.





Añadiendo un alert:

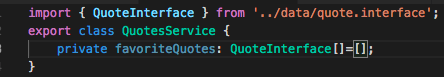
Utilizando el código en la documentación de ionic.

Manejando el servicio de favoritos.:

Se crea una carpeta services en el src, dentro el quote.service.ts



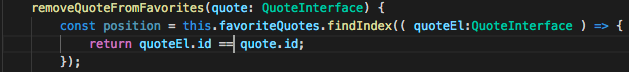
Se declara una variable de tipo arreglo quote donde se almacenaran los quotes favoritos



Ahora se añade un método para añadir un quote recibido al arreglo de quotes almacenados.



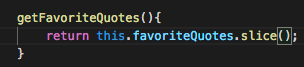
Para eliminar resulta un poco diferente ya que recibimos el quote, luego se debe acceder al arreglo existente y hacer un a busqueda mediante el método findIndex que recibe una función flecha como parámetro. Y compara con el parámetro quote recibido.



y luego removemos del arreglo con la función:



Y ahora una función que devuelva la lista de favoritos , pero una copia de el, con el método slice, para no tener la capacidad de modificarlo.

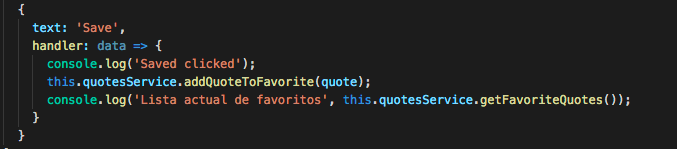


\*Utilizando el servicio para añadir favoritos

al ser un servicio se declara en el app module, providers.

Luego se importa en el componente a utilizar.

Y se pueden llamar a las funciones requeridas.:

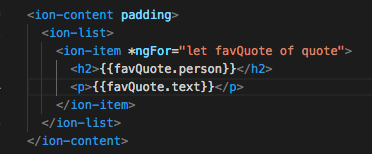


Ya añadidos los quotes favoritos, se pueden mostrar en la pagina inicial de favoritos.

Se declara una variable arreglo de tipo QuoteInterface, que recibirá el arreglo de quotes favoritos.

Para recargar esta pagina y función cada vez que se visualice la pagina, se llama al método de recarga de favoritos en el ionViewWillEnter.





Se ponen como ion ítem para que no aparezca el botón a la derecha que indica abrir otra página. En cambio se mostrara un modal al hacer click en ella.

Hay que tener en cuenta que un modal utiliza la estructura de una pagina, pero no es puesto dentro de la pila de paginas, tampoco se carga con el navController.

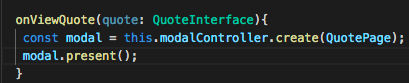
Se hace el llamado al modal desde la página de favorito

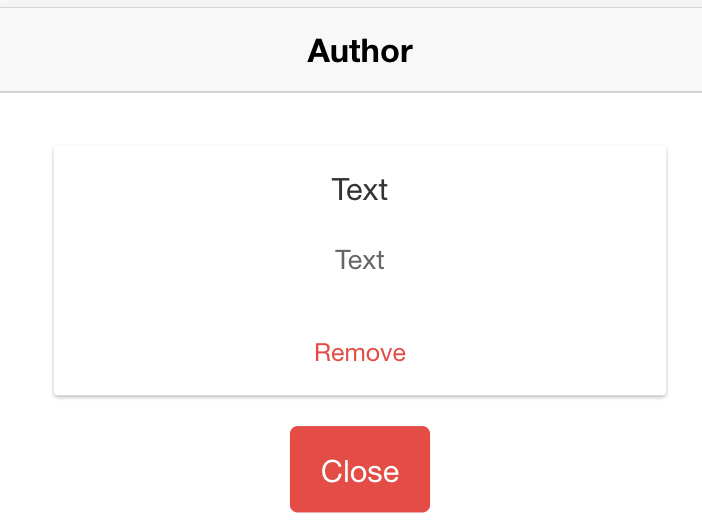


En el componente se importa el modalController.

Se asigna el valor del modalController.create(PARAMETRO), siendo el parámetro la pagina que queremos presentar como modal.

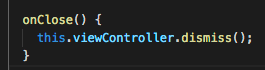
Y luego se llama el present.





Cerrando el modal:

Se necesita el ViewController



Pasando data al modal:

Al igual que el llamado de páginas se puede pasar data al modal con un segundo parámetro.



Ahora pasando los datos desde el modal hasta la pagina que esta atrás.

Desde el dismiss como parámetro se puede pasar el dato.

Al botón de favorito se le añade el evento click que llama al onClose,

El onClose se edita para que reciba el parámetro, y de no ser llamado con este parámetro lo declara falso, de otro modo es true y pasa a eliminarlo de favoritos.

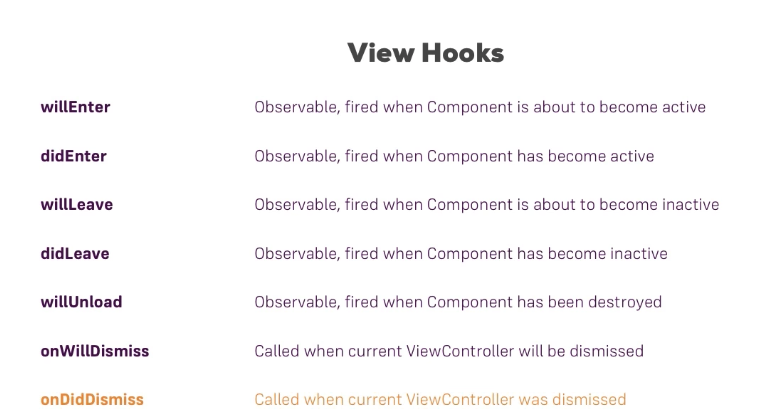


Al modal también se le pueden aplicar las funciones del ciclo de vida del componente.



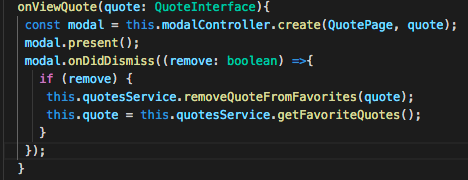
Profundizando:

ViewHooks



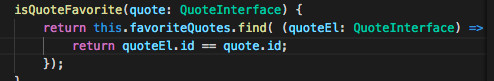
Recibiendo el parámetro, se hace desde la función del modal onDidDismiss.

Se recibe el parámetro, se valida según parámetro, y luego se carga la lista nuevamente.



Procedemos a prevenir que el usuario pueda almacenar un mismo quote como favorito varias veces.

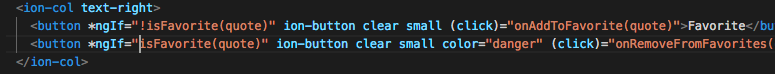
Se agrega al servicio una función para buscar un quote y verificar si ya es favorito



para luego llamarlo en el quotes.component.ts. donde se presenta la lista de quotes y verificar que botón presentar, si el de favorite o unfavorite.



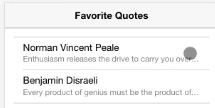
para luego comparar en el dom



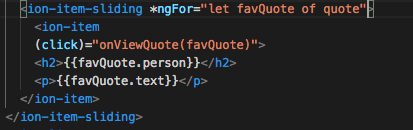
y para removerlo de favorito:



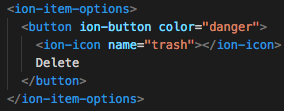
Añadiendo efecto de slide en la pagina de favoritos para tener un botón de remove al hacer el slide.

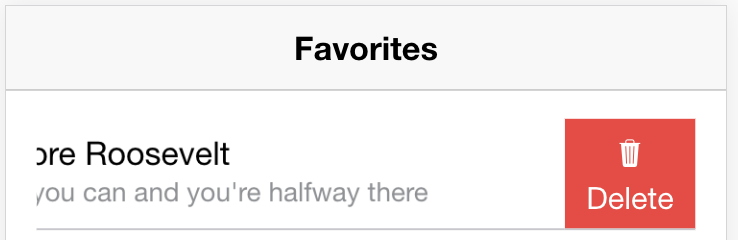


con el elemento ion-item-sliding se logra este efecto, pero debe contener el elemento que se hara slide para mostrar el nuevo. Y mover el ngFor al slider.

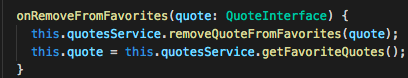


Luego dentro de nuestro wrap del sliding ponemos al mismo nivel del ítem. Estas opciones son reveladas una vez que el usuario hace slide en este ítem.





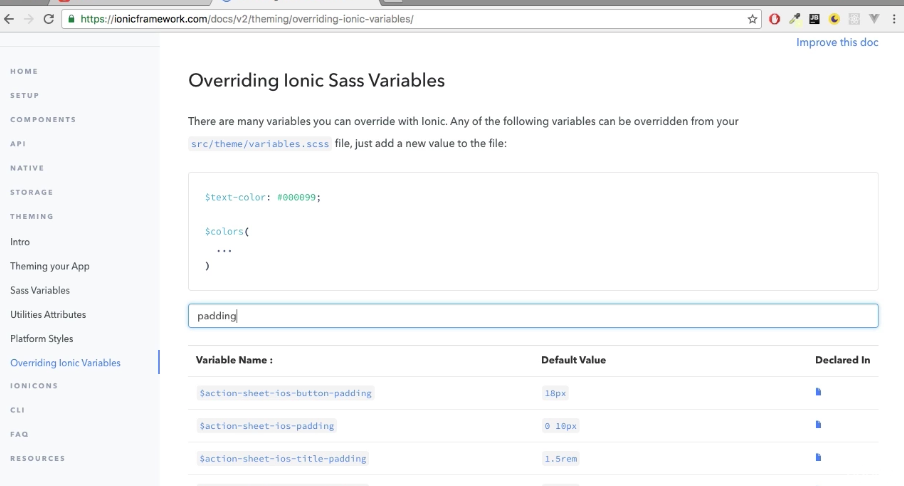
y finalmente dando la funcionalidad al boton



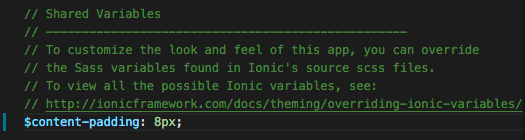
Changing the app theme:



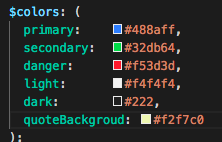
Desde la pagina oficial podemos ver las variables que existen y sus valores:



Las variables globales de la aplicación pueden ser modificadas o creadas desde este archivo. Dependiendo de donde queremos que haga efecto, en este caso en shared variables.



Añadiendo una propiedad a la variable de colores:

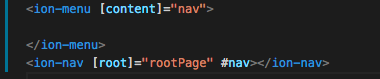


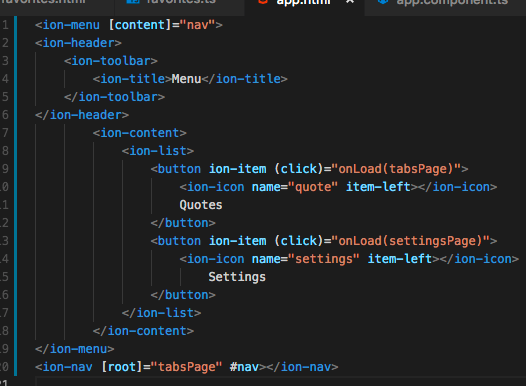
Añadiendo el Sidebar:

Se desea que al abrir alguna de las opciones que muestre el sidebar, se abra y desaparezca el menú de tabs, por lo que debe sobreescribir el rootPage.

Para esto directamente desde el app.html se agrega el ion-menu.

Añadiendo la propiedad con #nav , lo que hace una referencia local en el ion-nav, y unirlo, o referir el menú a este tag con [ ].



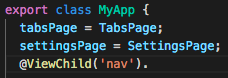


En el ts se declaran las posibles rootpages y se crea el método que la llamara.



Para poder acceder a la propiedad root:

Se el utiliza el ViewChild y en sus parámetros se hace referencia a la que creamos anteriormente en el dom

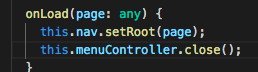




y luego en la función



Luego se importa el menúController para poder cerrar el menú, ya que al seleccionar una opción, este no se cierra automáticamente



Añadiendo el menú a la barra de navegación

Iniciando desde el favorites html se añade:

Dentro del navbar

Un ion-buttons y dentro el botón para mostrar la opción de menú.



Se añade la función al botón y luego en el ts se importa el MenuController





ahora viendo que estas modificaciones solo fueron para que funcione en la pagina de favoritos, hay una forma de hacerlo globalmente.

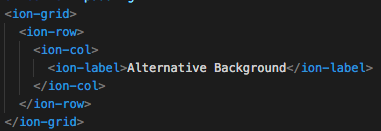
Se elimina la modificación del MenuController y simplemente se añade la directiva

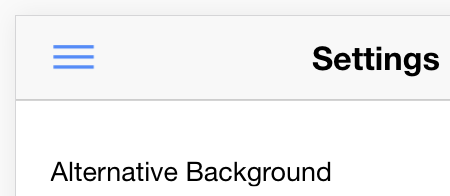
menuToggle



Configurando la pagina de settings:

Añadiendo un label

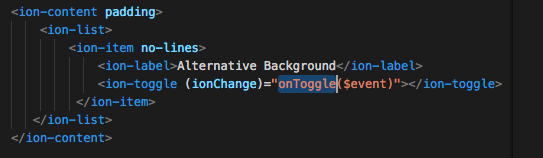




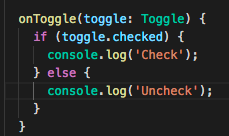
Añadiendole un toggle y luego la función que al activarse cambia el skin de la aplicación

El ionchange es la función que detecta el cambio, que lleva como parámetro un

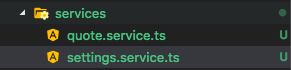
$event, que es el mismo que se detecta en el ionChange



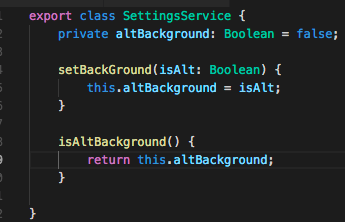
Al crear la función, para recibir el parámetro se declara de tipo Toggle, por que ya se sabe el tipo que se esta recibiendo.



Creando un servicio para guardar la configuración del skin de la aplicaicon.



En donde se tendrá una valor booleano que podrá ser modificado desde la pagina settings y luego validar en las pantallas cual es su valor para asi activar o no el skin alternativo;



Al ser un servicio hay que agregarlo a los providers en el app.module.

Y luego ya se puede importar en el settings.ts en el constructor

Para llamar a la función que activa o desactiva en el servicio el skin



y luego hacer una función que verifique el estado del check en el servicio, para validar en el toggle del html y una vez salga y regrese a es opción, verificar su estado y pintar el toggle en el estado correcto.

Para esto el tag toggle puede recibir una directiva que se asocia a un booleano para setear su estado [checked]

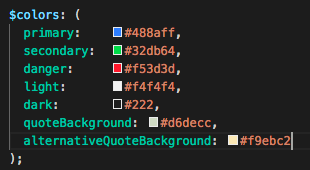




Añadiendo el background alternativo

Desde el archivo variables en theme:

Añadiendo la variable que contiene el color alternativo



En las diferentes paginas donde se presenta un quote se va a validar con una nueva función:

Favorites:

Utilizando un ternario para retornar la cadena deseada



y modificando la propiedad color para que pueda ser accedida en el ítem con la función creada.

