Awesome places:

Creando la aplicación:

Ionic start awesome-places

Añadir repositorio de git

Ionic g page …

Se utilizaran 4 paginas

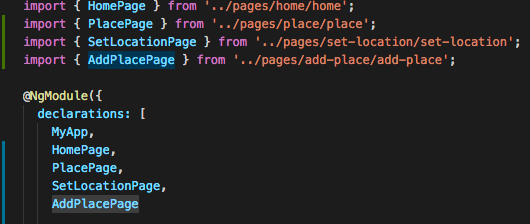
El home donde se desplegaran todos los lugares guardados,

Place: La pagina de cada lugar con sus detalles,

Add-place,

Set-location

Añadirlas al app module en **declarations** y **entryComponents**

****

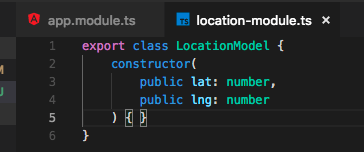
**MODELOS:**

**Se utilizaran 2 modulos para describir la data**

**Location**

**Place (individual)**

**Se crea una carpeta models en src y se crea el location.ts**

****

y luego el PlaceModel:

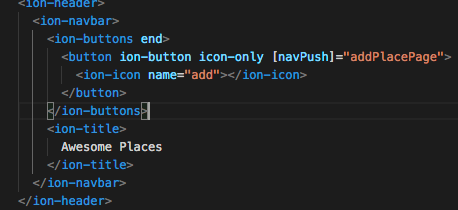
que tendra un location, title, description y imagen, la imagen sera un path osea un string que se almacenara en el dispositivo.

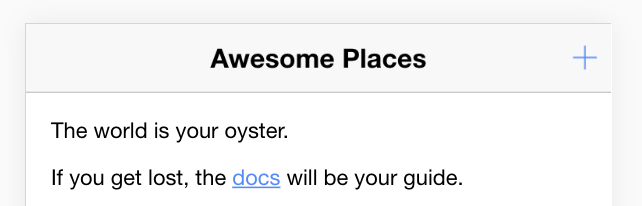


Añadiendo la navegación para el add-place

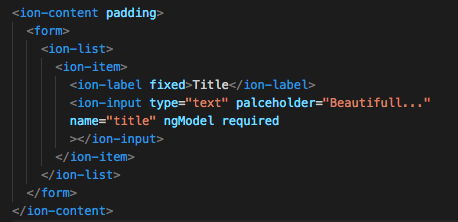
Desde el home page añadiéndole ion-buttons en el header:

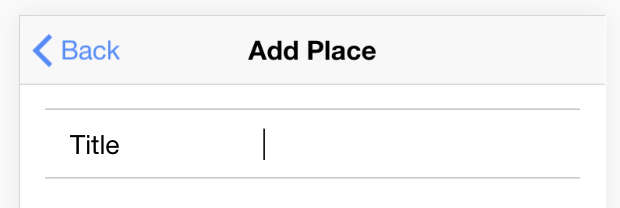






Luego se le añade contenido al add place, con un formulario donde tendrá los inputs

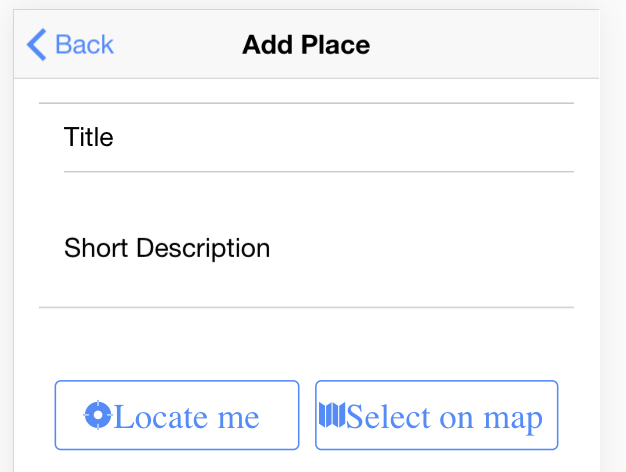




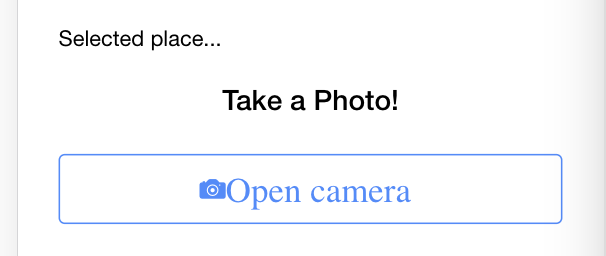
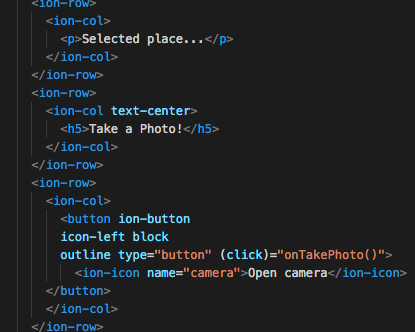
La siguiente parte del formulario estará fuera del list,

Añadiendo los botones para elegir la localización, importante ponerle el tipo al botón ya que al estar dentro de un formulario este haría submit de no ponerle el tipo.

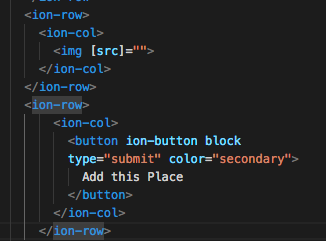


Se añade otro botón con la misma estructura y quedaría:   


Ahora se deberá desplegar un snipet de google maps para ver el lugar seleccionado en el mapa dentro de otro row

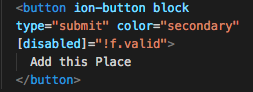


Ahora se va a mostrar un preview de la foto tomada y el botón para hacer submit



Para tener referencia del formulario se le asigna una referencia con # y la función que se ejecutara al presionar el submit, pasándole de parámetro los valores del formulario.  

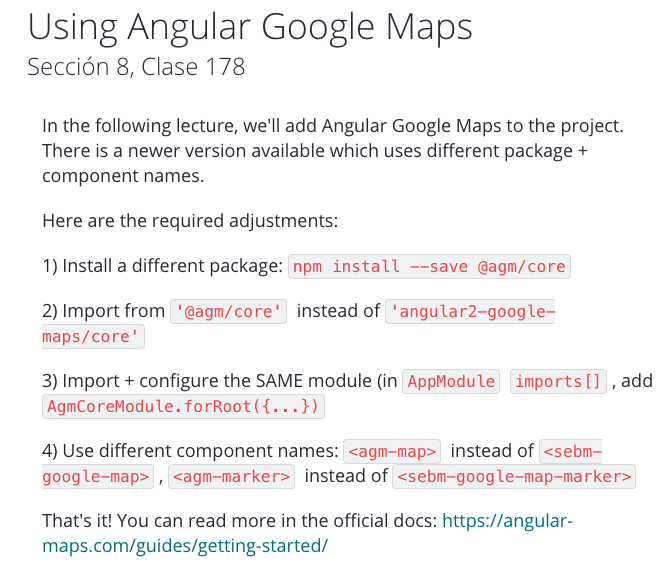

Con esta referencia podemos hacer que el botón de submit este disponible o no, según la validez del formulario:



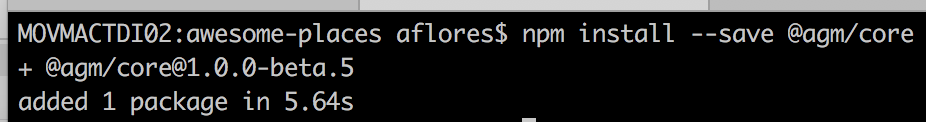
Añadiendo las funciones al ts



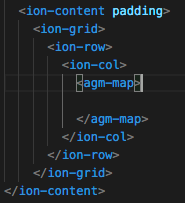
**Utilizando google maps**



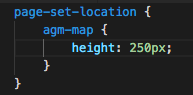
Instalando google maps en el proyecto:



Ahora ya se puede incluir en la aplicación, desde el set-location html:

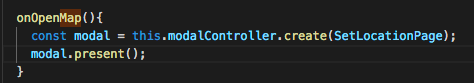


se utiliza el nuevo tag que se instalo con el maps y para visualizarlo hay que modificar el scss.



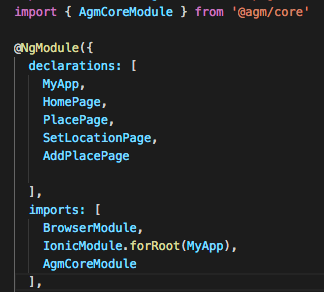
ahora para visualizar este cambio se implementa la función de onOpenMap ubicada en el add-place, que abrirá el set-location como un modal.

Para esto implementamos el modalController.



Haciendo esto aparece un error del template ya que no reconoce el tag “agm-map”

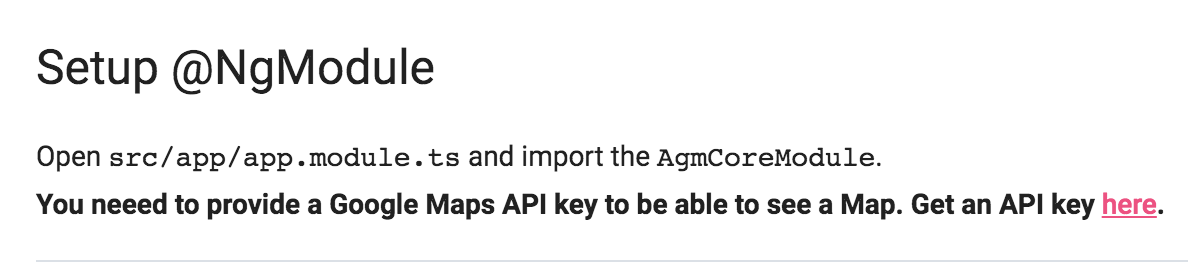
Desde el app module hay que configurarlo, se importa y declara en los imports:



donde en el import hay que configurarlo para que reciba un apikey de maps



desde la pagina de angular maps en getting started hay un link donde se consigue:

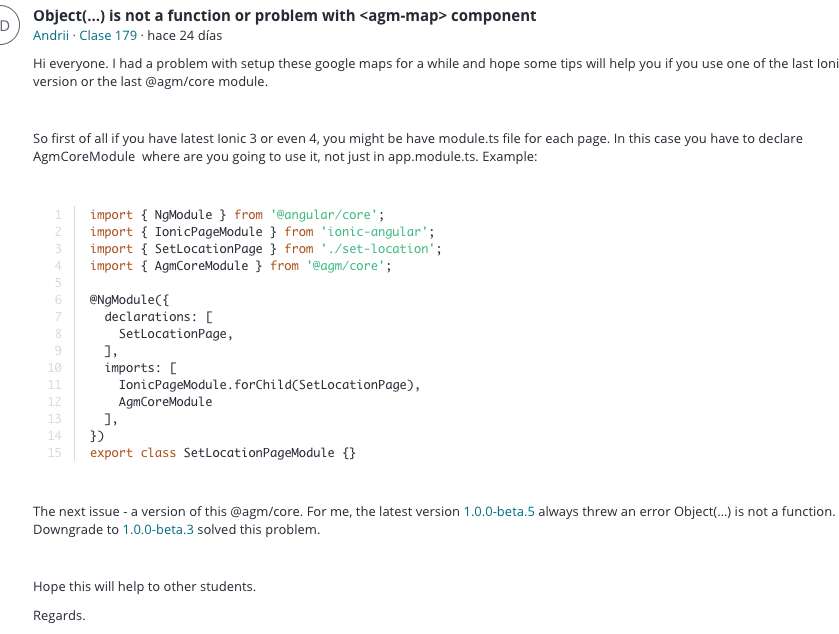


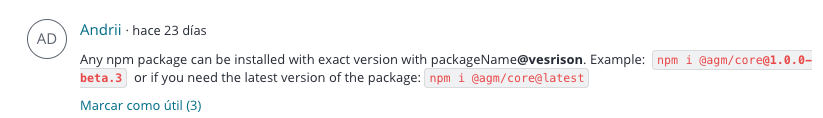
No es “gratuita”, pegarla en como atributo del objeto



DATO:

-Lanza un error al tener la ultima versión beta5, es necesario hacer un downgrade al 1.0.0-beta.3



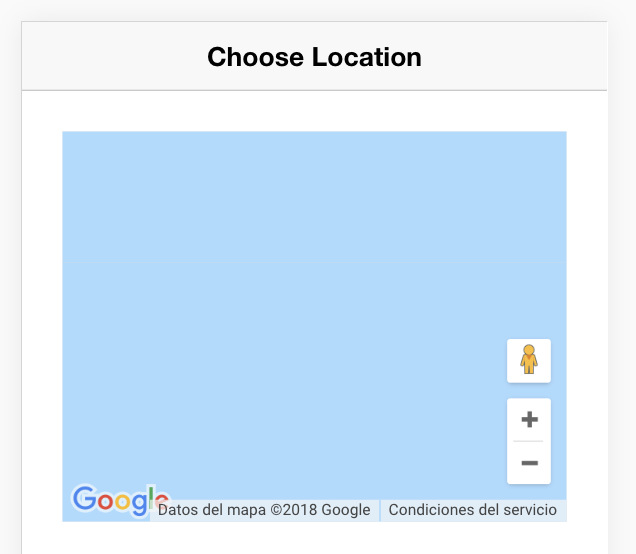


Hay que desinstalar la ultima versión instalada, y luego removerla de los paquetes

npm uninstall –save @agm/core

e instalar la version especifica:

npm i @agm/core@1.0.0-beta.3 –save

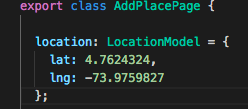


Configurando el mapa:

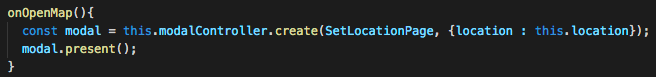
Para esto en el set-location.ts declaramos una variable location del tipo LocationModel.



Pero esta variable también la querremos en el componente de add-places asi que también se declara en ese componente y se le asigna un valor inicial.



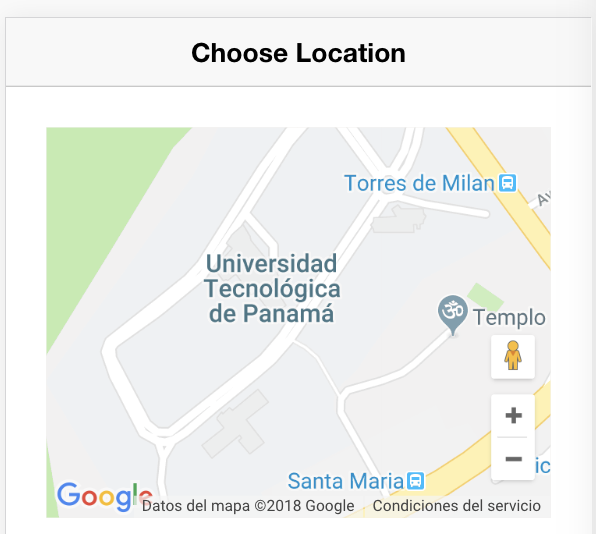
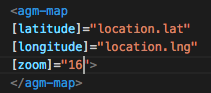
Para luego llamar al set-location, pasándole como parámetro esta dirección.



Para nuevamente en el set-location.ts obtener esta información cuando abra con el navParams desde el constructor:



Ahora desde el html podemos asignarle valores con propiedades del mismo tag:

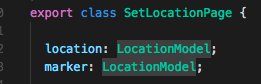


Seleccionando un lugar en el mapa:

Se activa un listener en el mapa con una función propia del api, mapClick

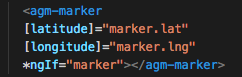


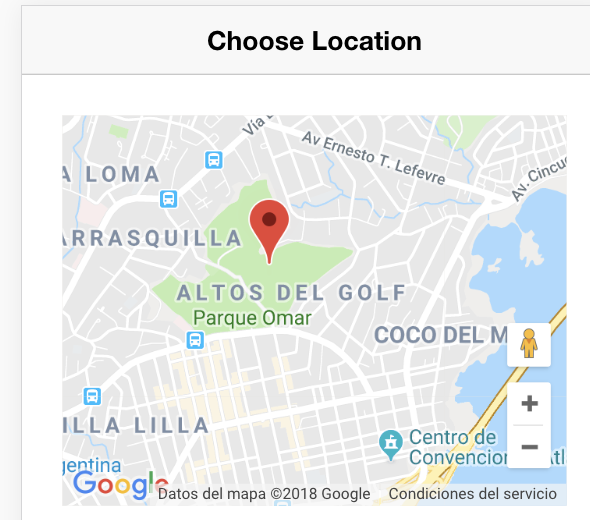
El mapClick recibe y manda un evento donde se puede detectar en que lugar (las coordenadas) fue clickeado y se envía como parámetro a la función a utilizar.





Ahora en el html podemos poner el marcador en las coordenadas obtenidas por el marker y solo mostrarlo si marker tiene valor, osea si ha hecho click en el mapa.



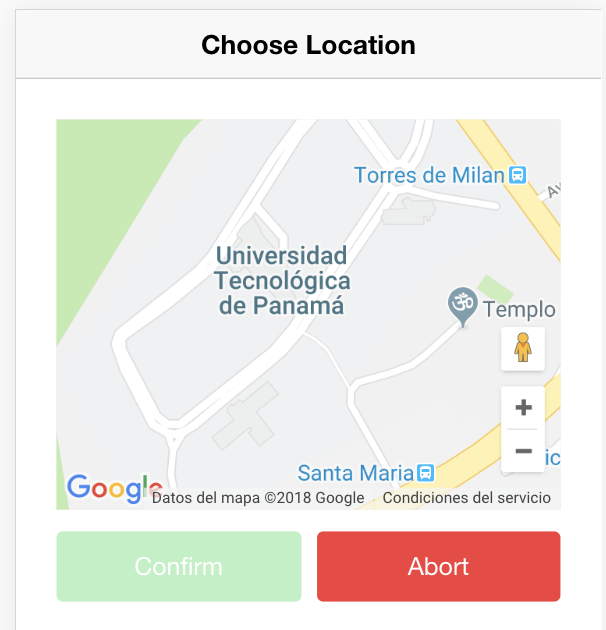


Ahora el siguiente paso es añadir un botón para, resetear o guardar la posición del marcador como nueva dirección inicial.

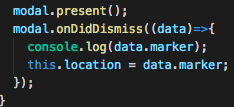


Se crea la función para confirmar la nueva dirección en el marcador y se pasa esta coordenada como parámetro en el dismiss

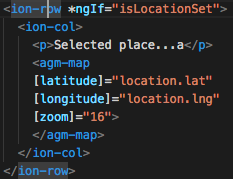
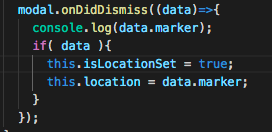




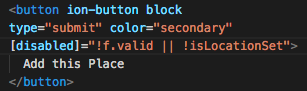
Luego en el add place se recibe este parámetro y se asigna como dirección a guardar del nuevo lugar, Al ser un modal en la pantalla anterior cuando se presento, se puede acceder al valor que devuelve al cerrarlo con la función del modal al dismiss.



Para mostrar el preview del location en el add-place y mostrarlo únicamente cuando se ha cambiado el lugar:

ahora el botón de submit también se va a validar según si esta introducida la dirección:



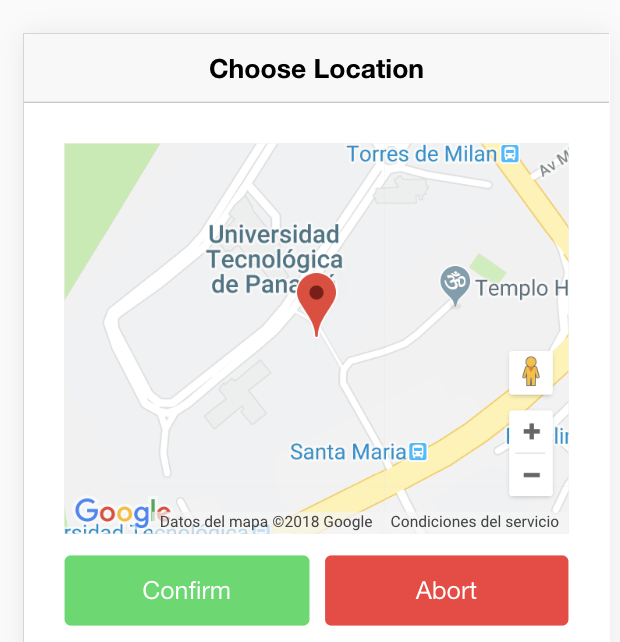
Psandole el valor si esta el location set para mostrar el marcador en el mapa para elegir:

Ad-place

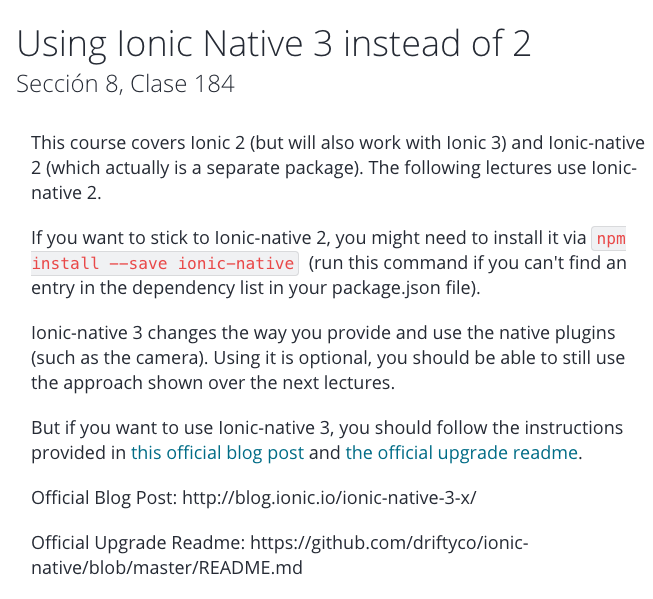


set-location





Funciones nativas, ionic 2 e ionic 3



<http://blog.ionic.io/ionic-native-3-x/>

<https://github.com/ionic-team/ionic-native/blob/master/README.md>

Geolocation

ionic cordova plugin add cordova-plugin-geolocation

npm install --save @ionic-native/geolocation

Importarlo en el app.module:

En los providers

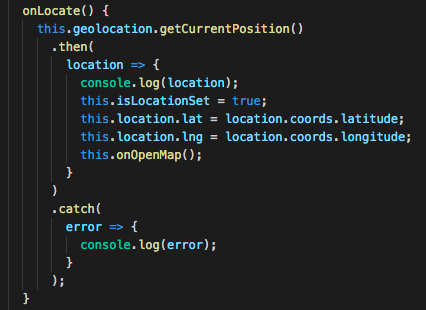


luego implementar la function de onLocate del add-place

importar en el constructor el geolocation desde



y acceder a sus propiedades getCurrentPosition, la que devuelve una promesa a la que se accede y se asignan los valores necesarios para la siguiente pantalla

o cargamos los errores.  


Mientras realiza esta acción demora mientras consulta, por eso es un buen lugar para poner un loading y toast.



Observación el gps del celular debe estar en alta calidad.

Usando la cámara del dispositivo.

En el add-place

Import camera from ionic native.

ionic cordova plugin add cordova-plugin-camera

$ npm install --save @ionic-native/camera

Declarándolo en el constructor y en el app module luego podemos acceder a sus propiedades.

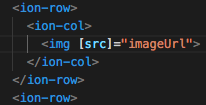
Aplicando la función, se accede a las propiedades del camera, se declara el tipo de imagen que creara y la orientación del dispositivo.

De ser correcto el funcionamiento en este caso imprime el path de la nueva foto





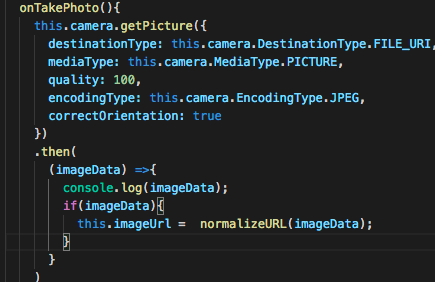
Siendo la idea mostrar la foto en la pagina de añadir lugar, para luego si se desea guardarla permanentemente en el dispositivo.

En el html se crea el espacio donde se mostrara con el src y la variable que se crea con este path  


Y modificando el ts añadiéndole las propiedades de la cámara,

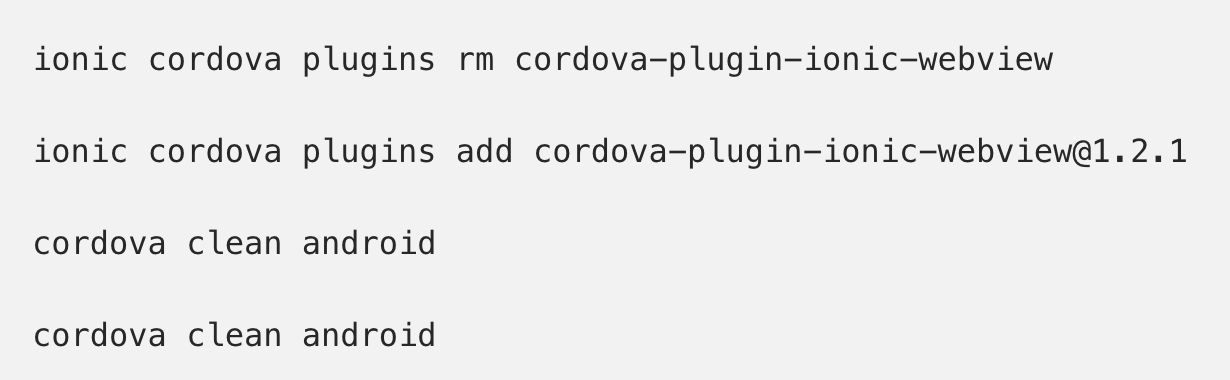
Y utilizando la funcionalidad normalizeURL para convertir el path





(Aparece un error que parece relacionado al webview) <https://medium.com/deyvin/ionic-3-4-fixing-not-allowed-to-load-local-resource-camera-file-uri-3c1db24ad874>

Hay que jugar con las versiones del webview, y si es necesario desinstalarlo:

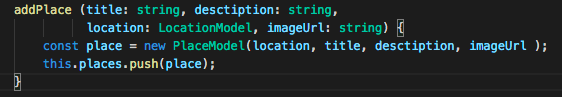


Manejando los datos con el servicio PlacesService:

Se crea una carpeta services y la clase con un arreglo vacio de tipo PlaceModel.



Se crea la función para añadir un lugar a un arreglo local donde se pueda consultar.

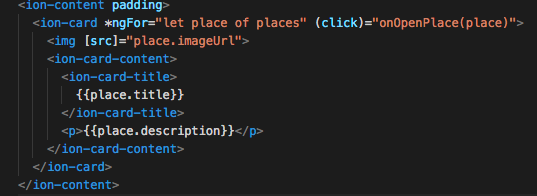


y creando un método que retorne una copia del arreglo de lugares almacenados.



Mostrando los lugares almacenados:

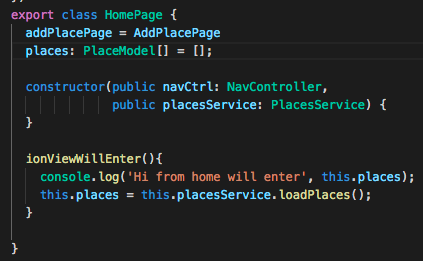
En el home.html se van a mostrar en cards



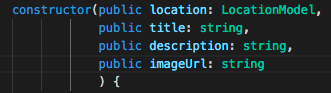
Ahora hay que crear el arreglo de Place para cargarlo con la información del servicio.

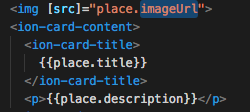
Añadiendo el servicio al app – providers,

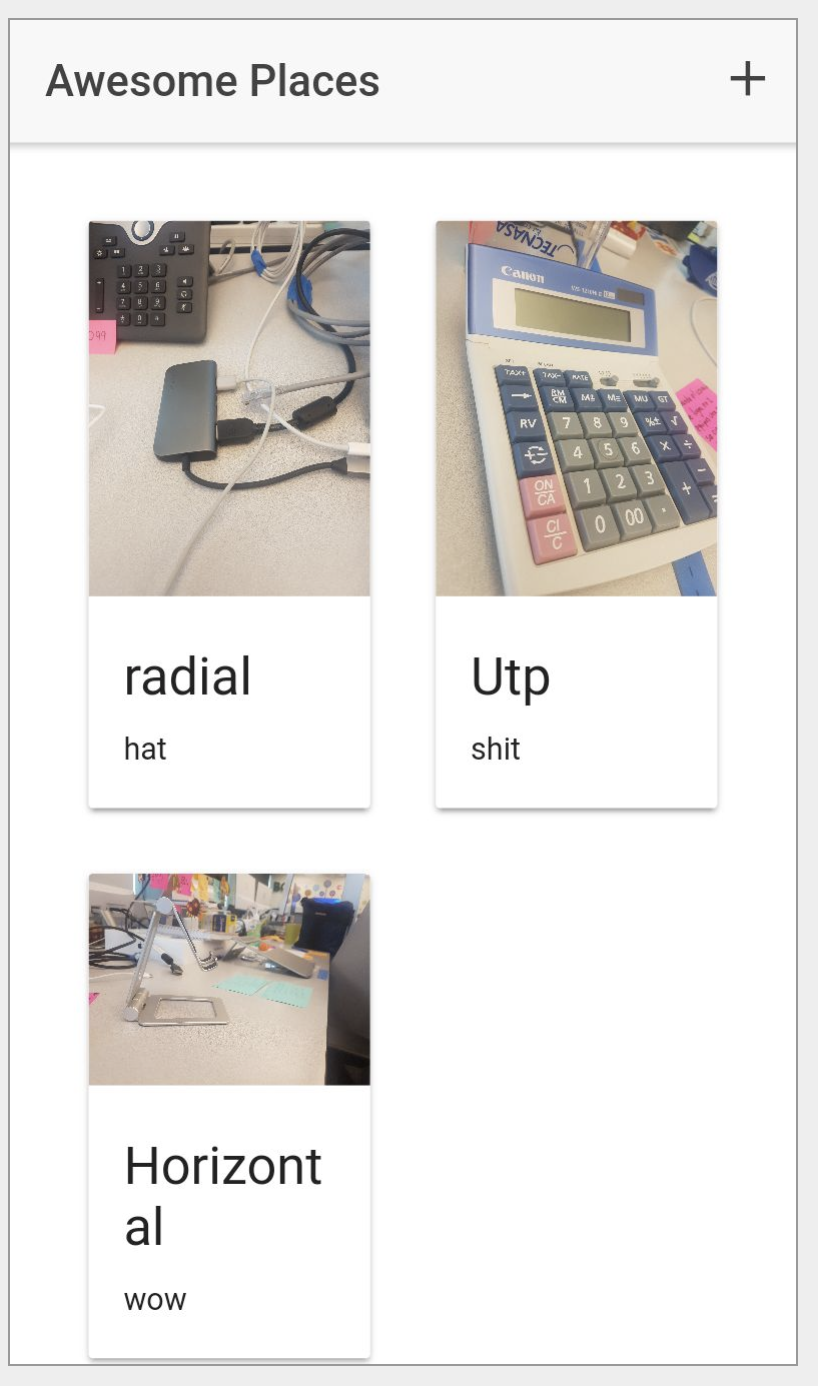
En el constructor de home, y luego accediendo a sus funciones.



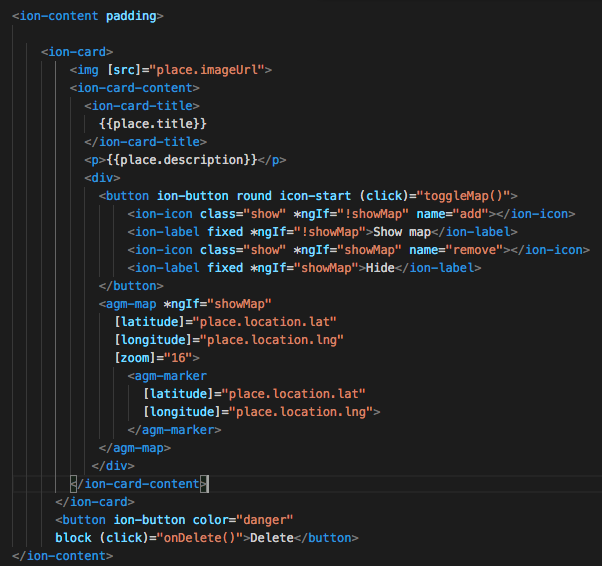
Importante, recordar que el nombre en el model sea el que se utilice en los html a visualizar.:

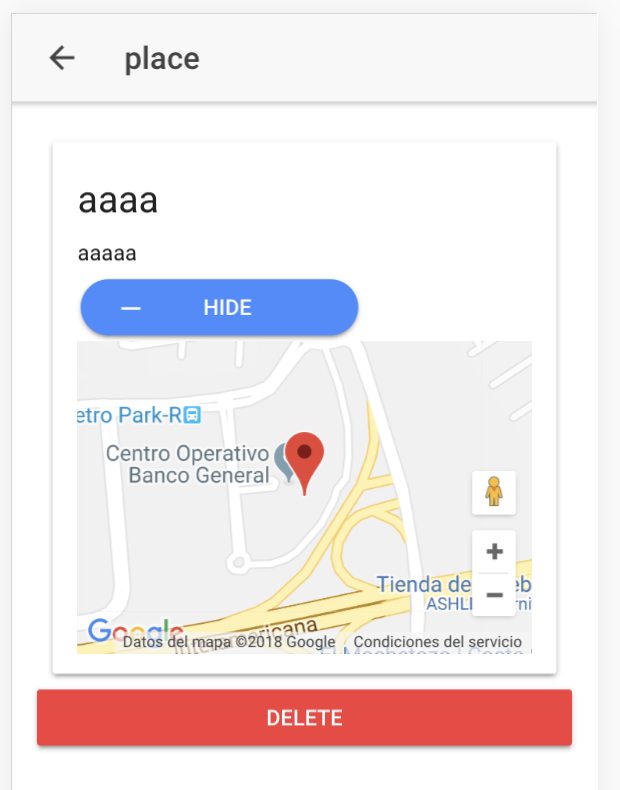






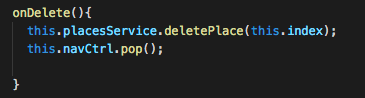
Configurando la vista al hacer click en un lugar:





Añadiendo funcionalidad de borrar:

En el ts



En el servicio



Guardando en memoria del dispositivo.

FileStorage instalar con la documentación de ionic

ionic cordova plugin add cordova-plugin-file

$ npm install --save @ionic-native/file

Desde el add-place  




Empezando cuando tomamos la foto y guardamos queremos moverla a el almacenamiento del dispositivo.

Para encontrar el path y el nombre del archivo es necesario utilizar expresiones regulares

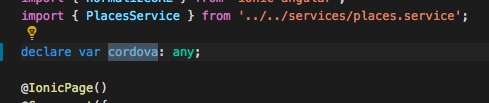


luego utilizando la propiedad de File, la función moveFile, que recibe como parámetro el path, nombre, el nuevo path, y el nuevo nombre.)

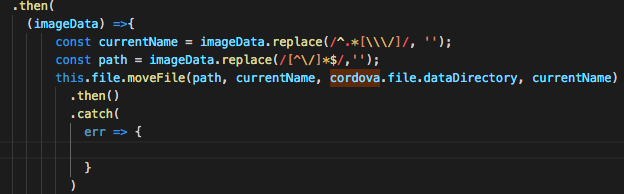


Para poder utilizar cordova hay que declararlo como variable

Antes de la clase,



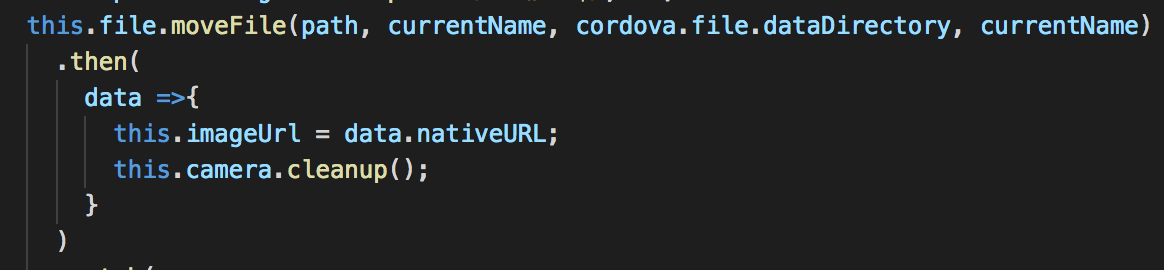
la función retorna una respuesta que puede fallar por esto se maneja de esta forma.



En caso que falle se presenta un toast avisando, se vacia la cadena del imageUrl y se limpia la cámara.



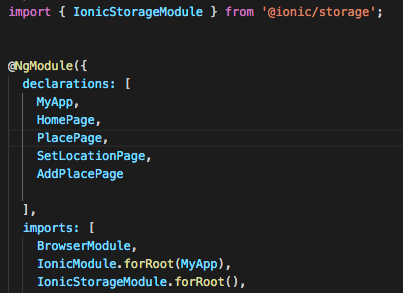
En caso exitoso, esta función devuelve un objeto con la siguiente propiedad



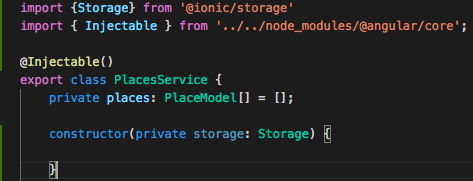
**Using Storage.**

**https://ionicframework.com/docs/storage/**

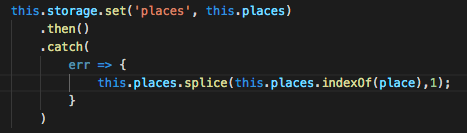
En el app importar IonicStorageModule:



Desde el places.service:  
Importarlo desde ionic/storage y agregar Injectable() a la clase

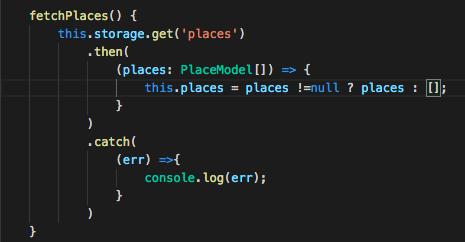


utilizando la function set del storage, aqui se valida con una llave para luego utilizarla, donde se devuelve una promesa. En caso de error al guardar este lugar, también se elimina de la aplicación ya que tiene algún error para que los datos sean congruentes entre la app y el almacenamiento en el dispositivo.

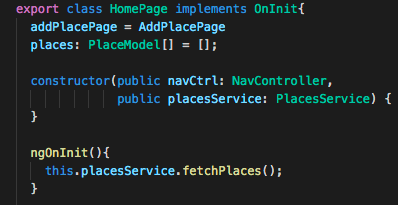


Ahora agregando una función que traiga la información desde el storage utilizando la función get del storage, que responde una promesa, en donde si funciona correctamente, se asigna el valor que responde siendo este un arreglo del modelo places, y se asigna al arreglo local, solo si es diferente a null, : de otro modo asigna un arreglo vacio.

Y en caso de error se imprime este.



Una vez listo el método, se llamará cada vez que el home es creado:



Borrando los datos almacenados en el storage:

Desde places.service

Se crea la función que remueve el place del dispositivo, utilizando la expresión regular para objener el nombre y reemplazarlo con una cadena vacia.

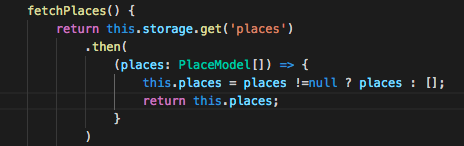
En el caso de un error, como en este momento ya se ha eliminado desde la aplicación y los datos deben ser iguales en el storage, hay que agregarlo nuevamente a la aplicación.



De esta forma al guardar un lugar, cerrar la aplicación y abrirla nuevamente, no carga apenas se entra a ella, es necesario una navegación para ver los datos guardados.

Se modifica el fetchPlaces para que retorne la promesa y poder subscribirse a ella desde el OnInit.

Desde el servicio



y en el home



En este punto se sobreescribe la foto tomada para los anteriores lugares guardados.

Se soluciona cambiando la lógica del add-places, donde se utiliza la función moveFile, el ultimo parámetro que recibe tiene el mismo nombre siempre y por eso sobreescribe, se crea una variable que tendrá siempre un nombre único, utilizando la fecha con milisegundos.

