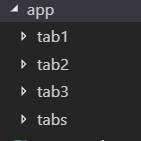
Scanner

<https://www.qrcode.es/es/generador-qr-code/>



Como al momento de crear los tabs los crea directamente en src, creamos manualmente la carpeta pages y metemos ahí los tabs

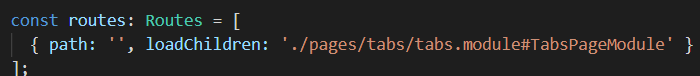


Luego hay que modificar el app routing con la nueva ruta

En el tabs html



Modificamos en el app routing el path de tabs module



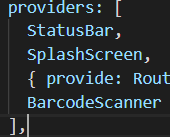
Utilizando el plugin

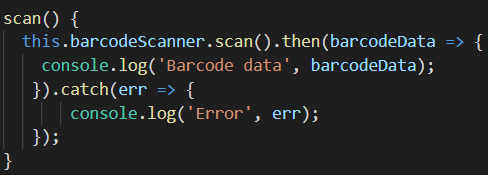
<https://ionicframework.com/docs/native/in-app-browser#installation>

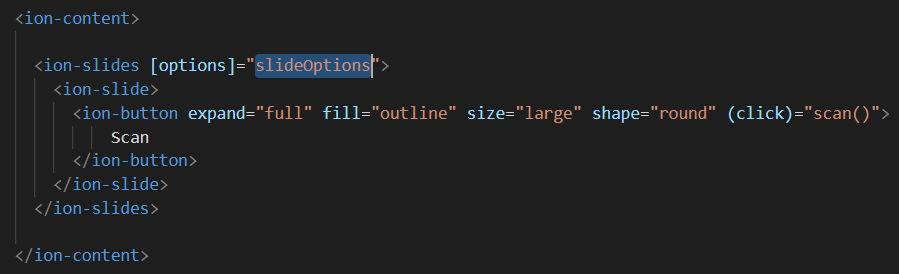
El barcode scanner lee Codigos Qr y codigos de barra



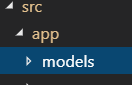
Y luego hay que implementarlo en el app module con los providers

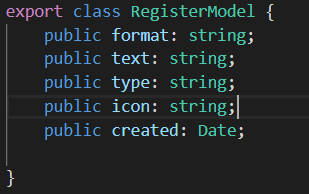




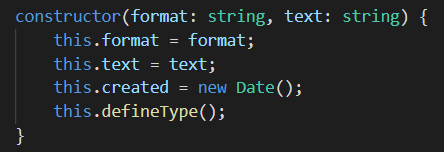


Creamos una carpeta models





Y luego el constructor

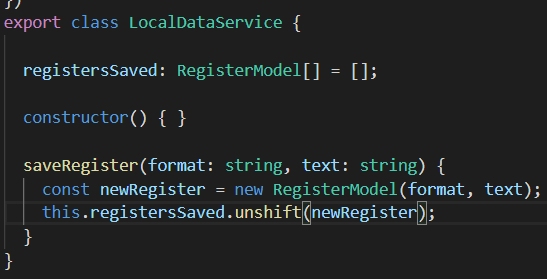


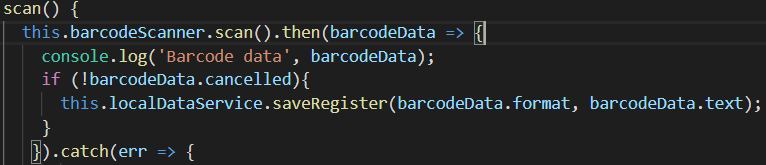
Luego en el método que detecta el tipo de qr definiremos con un switch



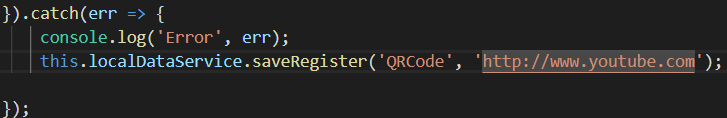
Ahora crearemos un servicio para almacenar los datos escaneados



Creamos el arreglo de tipo register model que almacenará los registros.  
En el método de guardar registros recibimos como parámetro los datos que lee el scanner y lo creamos como un nuevo registro . Luego lo almacenamos en el arreglo en la primera posición con el unshift.  


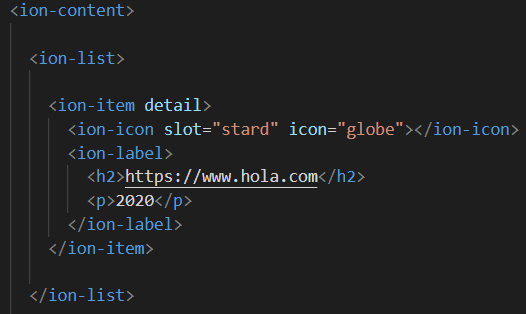
Y desde el tab llamamos al servicio  


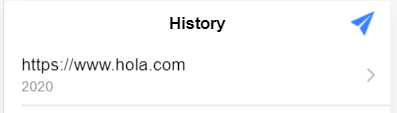
Para efectos de prueba en el error para ver desde la pc:



Mostrar registros en el historial:

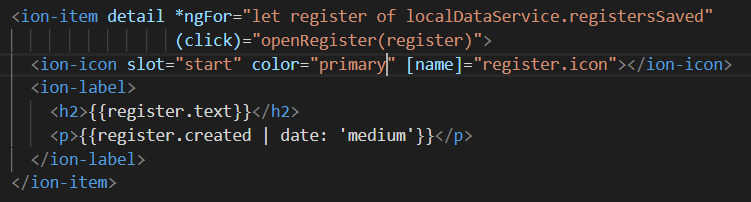
Creamos el header del tab 2  



Ahora importamos el servicio y lo llamamos directamente desde el html



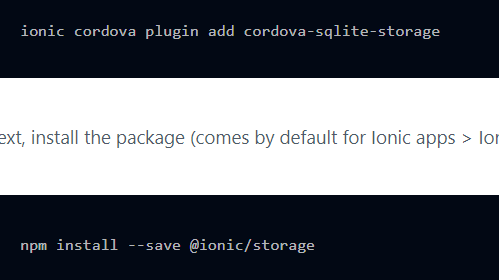


Guardando la información en el storage:

Cargar los registros desde el constructor del servicio

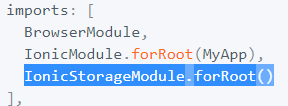
Es necesario instalar el plugin de Storage

<https://ionicframework.com/docs/building/storage>



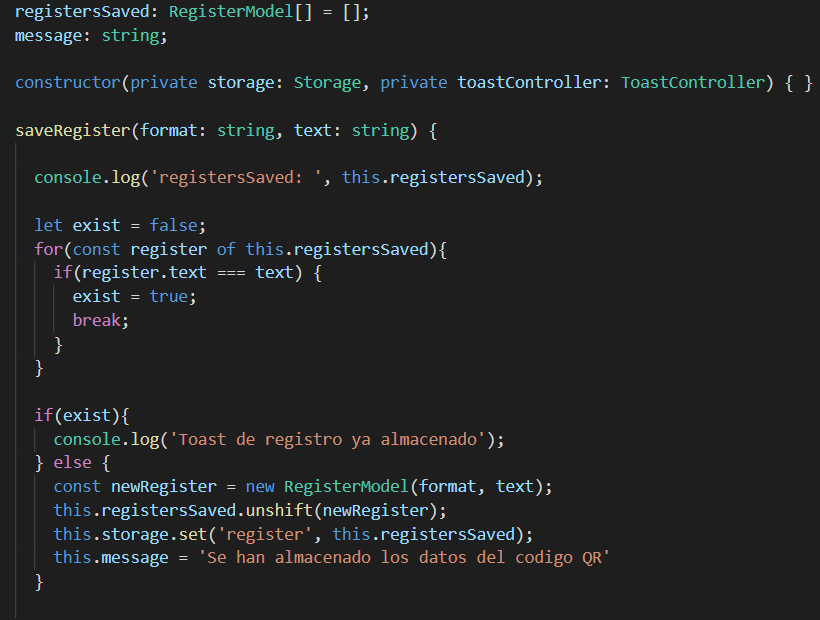
Y luego agegarlo al app.module en los imports





En el servicio se importa  


Se recorre el arreglo actual para verificar que no exista. De no existir se crea el registro y se guarda en el almacenamiento con el key “register”.



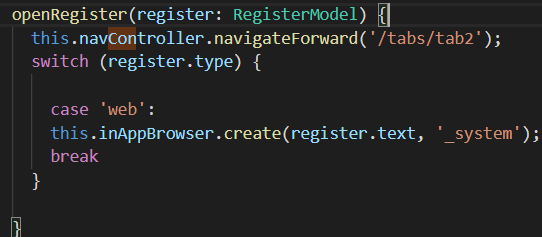
Abriendo el url en el navegador en el dispositivo:  
Para esto necesitamos instalar el plugin de inappbroser  
<https://ionicframework.com/docs/native/in-app-browser>



Colocarlo en el app-modules en los prividers y en el componente a utilizar.

Vamos a leer desde el servicio y redirigir la vista al segundo tab para mostrar la lista actualizada de objetos escaneados.

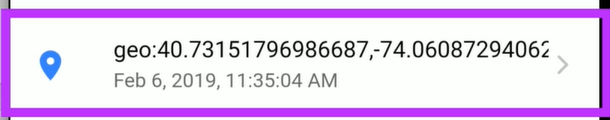
Llamamos a esta función cuando estamos seguros que creamos un nuevo registro desde el servicio: osea desde sabe register.



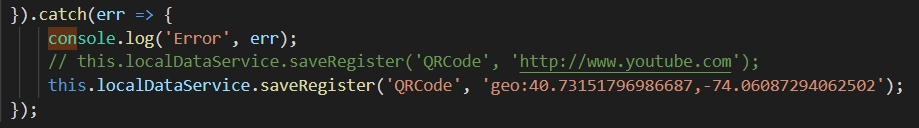
Preparando la aplicación para escanear ubicaciones geográficas.

Google maps necesita registro y tiene un costo al alcanzar una cantidad de consultas, por eso usaremos de practica mapbox.

Al generar un código qr con mapa desde qrgenerator. Si lo vemos desde la aplicación nosmanda la geolocalización por latitud y longitud.

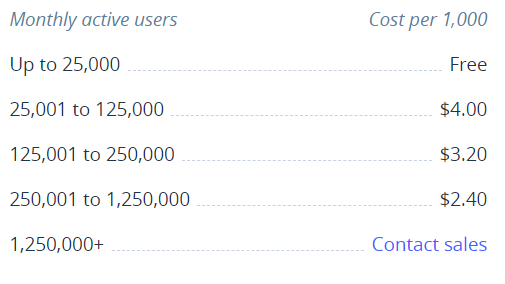


Mockeamos el error desde la pc:



Crearemos un api key con mapbox

<https://www.mapbox.com/>



Nos logueamos en en el navbar sale el token para utilizar.

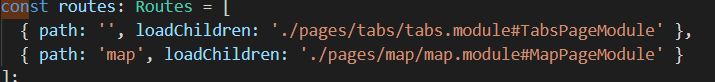
Al tocar un registro de mapa redirigirnos hacia un mapa

Creamos una nueva pagina para mostrar el mapa.



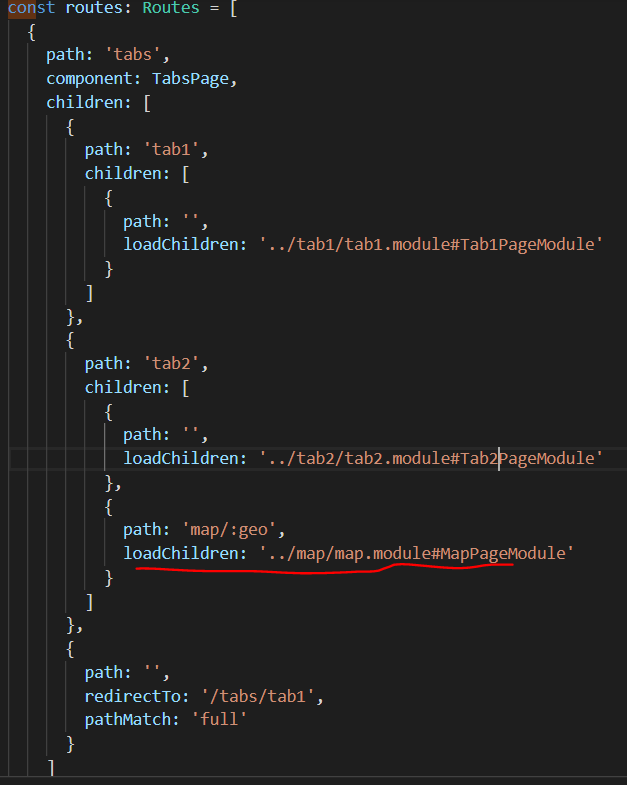
Al crear la pagina actualizó el rounting, creando una ruta para el mapa. No deseamos este comportamiento desde la barrade navegación asi que lo eliminamos.



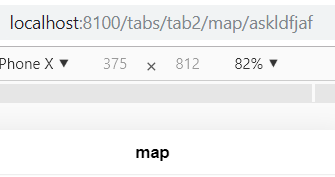


Ahora el comportamiento deseado es que al tocar un registro con mapa nos mueva a esa pagina, respetando los tabs.

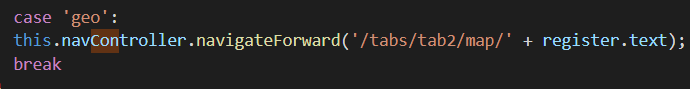
Desde el tabs.router creamosla ruta hija en el tab2, ya que desde ahí se llamará el mapa. Dentro del arreglo children de tab2



Ahora para acceder desde el navegador con la ruta:

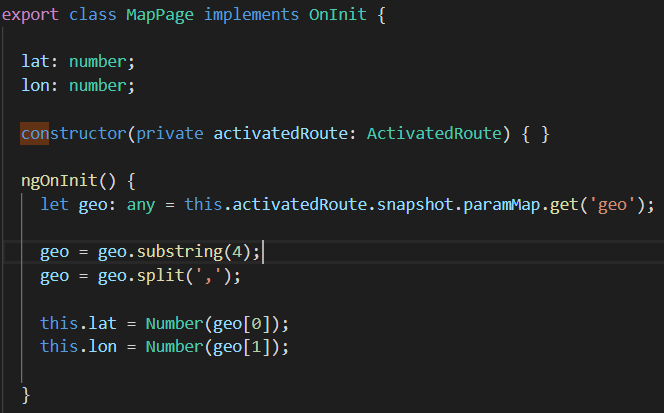


En el servicio agregamos en el openRegister la opción geo y enviamos la vista hacia esa pantalla.



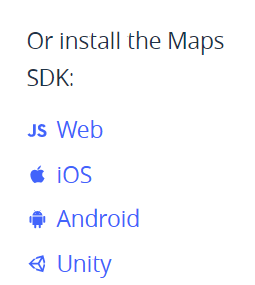
Desde la pagina de mapa

Improtamos el activated route para poder acceder a los parámetros que están llegando por url.  
y los asignamos a la variable geo. Luego dividimos esta variable ya que nos llega “geo:9.1231233 ,-74.123123123” y necesitamos los números separados.

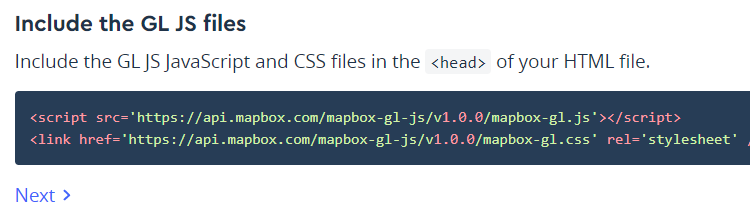


Mostrando las coordenadas en el mapa:

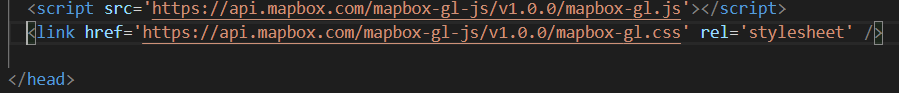
Desde la pagina de mapbox ingresamos al sdk de la web.



Copiamos el código que nos proporciona la pagina.

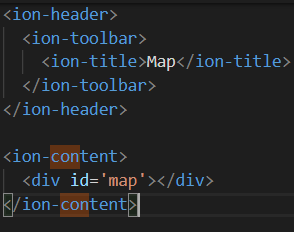


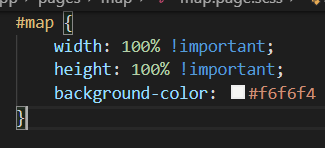
Este código lo ubicamos en el index.html antes de cerrar el header



El siguiente paso es copiar el código html y pegarlo en nuestra pagina de mapa. En este caso solo copiaremos el div.



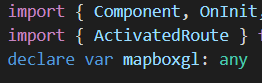


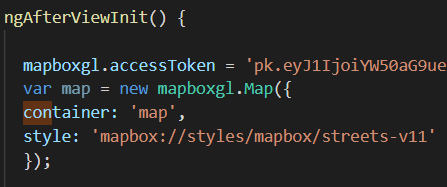


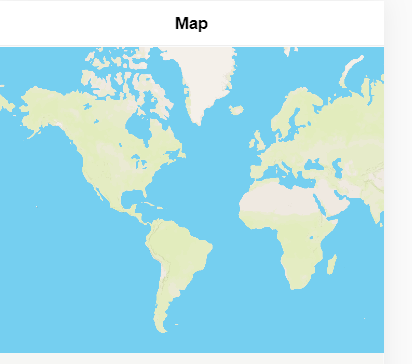
Ahora el script:  
Desde el .ts importamos AfterViewInit que se ejecutara cuando ya haya cargado la pagina, librerias y la latitud y longitud.



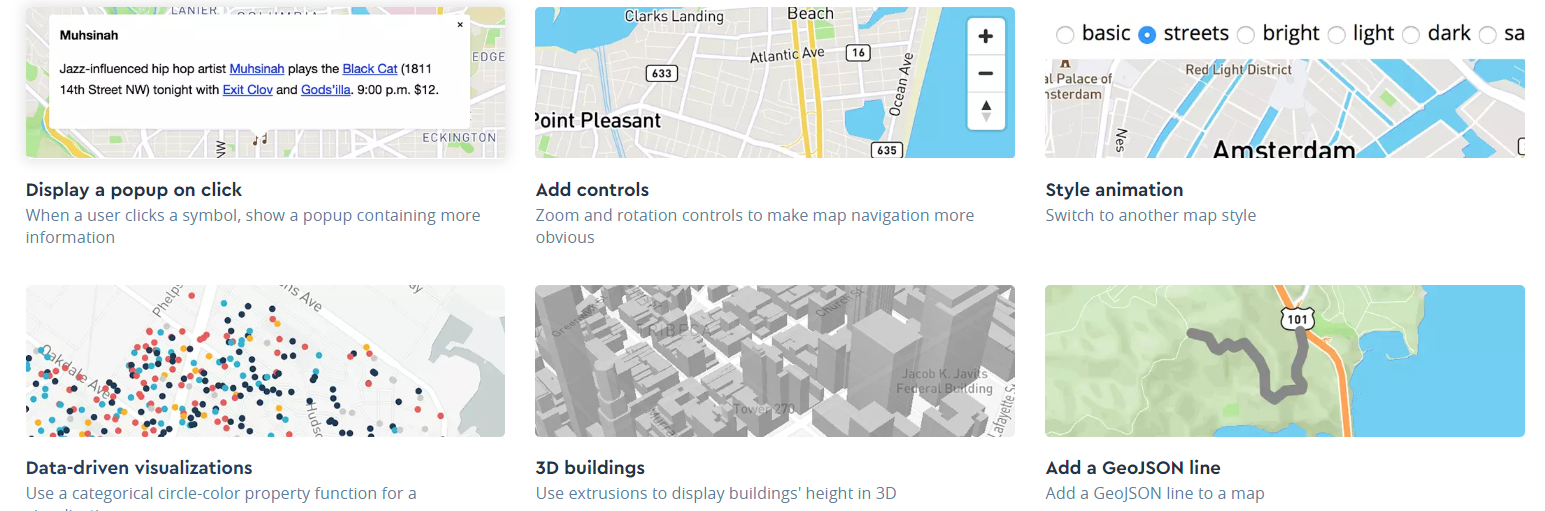
Declaramos la variable mapboxg1 en el componente ya que esta fue importada desde el index.html y luego pegamos el código en la función del ngAfter



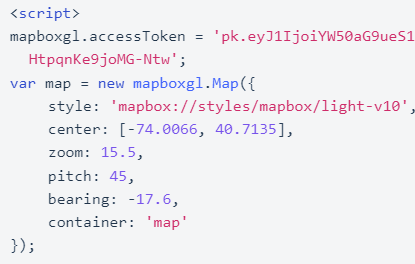




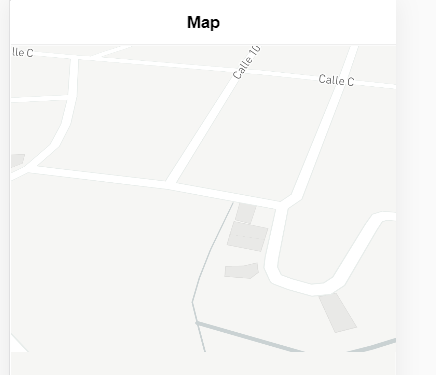
**Mapbox ofrece diferentes tipos de mapa**



Implementando edificios 3D y marcadores  
Seleccionamos el tipo de mapa e implementamos el código del script.



Y reemplazando las variables en el código obtenemos:

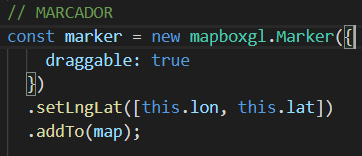


Para implementar los edificios copiamos el código de map.on…

Ahora crearemos un marcador para ver la posición. Para esto en la documentación de mapbox

<https://docs.mapbox.com/mapbox-gl-js/example/drag-a-marker/>

Cambiando únicamente de la documentación la referencia a las longitud y latitud por las variables actuales.



Y la función de callback que existe en el map.on por una función flecha para que no cambie el scope de la palabra reservada this.



Añadiendo un botón para regresar al historial

