ZEROD

Drive safety
WITHOUTACCIDENTS

FRANCISCO JESÚS SÁNCHEZ MUÑOZ

PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Contenido

1. Descripción del problema

Hoy en día, cuando salimos a carretera ya sea para ir o volver del trabajo, por ocio o por cualquier desplazamiento que necesitemos hacer con nuestro vehículo, no podemos saber con exactitud, si la ruta que elegimos diariamente es la más adecuada para llegar a nuestro destino rápidamente y sin sobresaltos. Por ello el solventar la escasa información de nuestras carreteras a la hora de la comunicación de accidentes, atascos, radares y otros tipos de problemas como la climatología nos permitirá que el realizar nuestro traslado sea más rápido, eficiente y seguro.

2. Objetivos del proyecto

Se ha pensado desarrollar una aplicación movil (con soporte web), para que todo aquel que se encuentre algunos de los problemas mencionados anteriormente lo comunique de manera rápida y eficaz, para así ayudar a todo el que necesite realizar algún tipo de desplazamiento con su vehículo, decida la ruta que mas le conviene para llegar a su destino.

3. Recursos necesarios

Para el desarrollo de esta aplicación se va a utilizar un ordenador portátil MSI con estas características:

- Tarjeta Gráfica => GTX 1050Ti 4GB GDDR5
- Memoria Ram => 16GB GDDR4
- Almacenamiento => 256GB SSD y 1TB HDD
- Procesador => i7 7700HQ a 3,8 GHz
- Pantalla => 1920 x 1080p

Además de hardware necesitaremos configurar nuestro entorno de desarrollo:

El sistema operativo utilizado dependerá del gusto de cada programador en mi caso Windows 10 64bits.

Se va a utilizar el editor de código Visual Studio Code.

Vamos a hacer uso de Node, npm y de git.

Para ello descargamos Node desde su web: https://nodejs.org/en/download/

Nos descargamos la última versión (LTS Versión: 10.15.3 incluye npm 6.4.1).

Y ahora Git: https://git-scm.com/downloads

Una vez instalado esto, podremos instalar el módulo de cordova desde npm.

En mi caso se va a hacer uso del comando: npm install –g cordova.

Para otros SO visitar: https://cordova.apache.org/docs/es/latest/guide/cli

Por ultimo instalaremos Ionic desde este comando: npm install -g ionic

Una vez hecho todo esto tendremos todo listo para empezar a trabajar en nuestra aplicación.

Para poner en producción esta aplicación, se va a hacer uso de un dispositivo Android y un PC con Windows 10.

El terminal Android tiene las siguientes especificaciones:

- Versión de Android => 8.0.0 SDK 26.
- Memoria Ram => 3GB
- Procesador => Snapdragon 800 a 2,3 GHz
- Pantalla => 1920 x 1080p

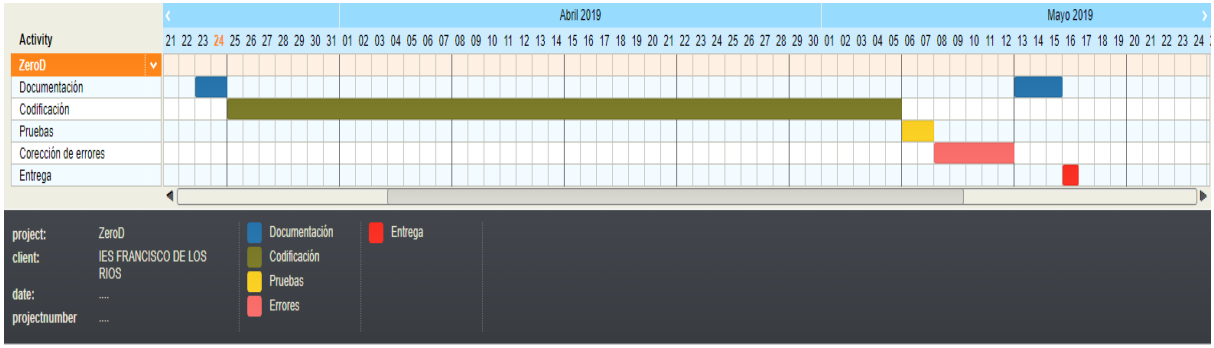
En este dispositivo solo es necesario descargar la aplicación e instalarla para empezar a probarla.

El terminal Windows tiene las siguientes especificaciones:

- Tarjeta Gráfica => GTX 1050Ti 4GB GDDR5
- Memoria Ram => 16GB GDDR4
- Almacenamiento => 256GB SSD y 1TB HDD
- Procesador => i7 7700HQ a 3,8 GHz
- Pantalla => 1920 x 1080p

En este dispositivo solo será necesario tener acceso a internet y un navegador web por ejemplo Chrome.

4. Planificación temporal



5. Desarrollo del proyecto

5.1. Requisitos de la aplicación

La aplicación que se desarrollará deberá poder añadir con un solo clic en un mapa una alerta de cualquier tipo.

Esta aplicación tendrá acceso al GPS del dispositivo, para que cuando se pulse un botón se cree una alerta del tipo que especifique el usuario en la ubicación de este.

Cada vez que se cree una alerta se mandará una notificación a los usuarios de esta aplicación.

Esta aplicación tendrá conexión a internet para poder publicar las alertas, si en algún momento el usuario no tuviera conexión y publicará una alerta, esta se almacenará en nuestro dispositivo y al volver a recuperar la conexión se comprobaría si ha llegado a publicarse, en caso negativo automáticamente se publicaría.

La aplicación contará con registro a través de diferentes plataformas, con el motivo de poder ver las alertas creadas por cada usuario y si siguen activas o no.

5.2. Diseño de la aplicación

5.2.1 Diseño de la arquitectura

La aplicación consta de 3 tabs con funcionalidades diferentes pero que entre sí enlazan el total de requisitos para una aplicación sobre noticias a tiempo real del estado de nuestras carreteras.

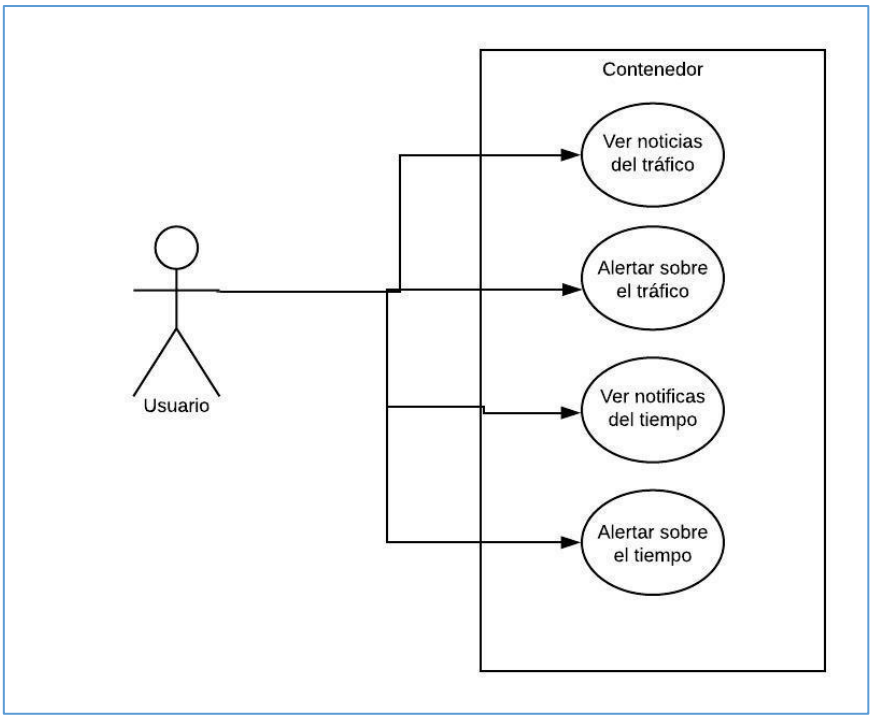
La primera será un resumen de las alertas más cercanas a nuestra ubicación.

La segunda podrá observar el mapa con las alertas activas en ese momento.

La tercera tab tendremos la configuración de la aplicación, que permitirá al usuario adaptar la aplicación a la necesidad que más encaje para él.

Para acceder a la creación de alertas el usuario tendrá que realizar el login, que se encuentra en la tercera tab. Esto le permitirá añadir una serie de configuraciones extras a la aplicación.

Se adjunta el siguiente caso de uso para tener una idea más clara de lo que el usuario quiere hacer en esta aplicación



5.2.2. Diseño de Datos

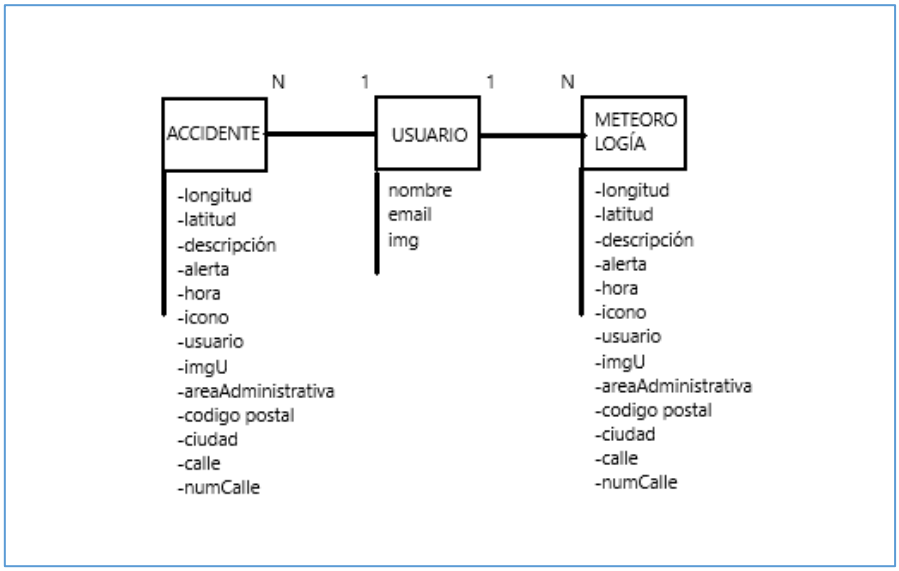
La aplicación almacena los datos que los usuarios guardan en una Base de Datos no relacional. Para esto se está utilizando firebase con 3 tablas:

Dos tablas para el almacenamiento de alertas, ya que se clasifican en meteorología y tráfico

Una tabla para almacenar los perfiles de los usuarios.

En el dispositivo móvil se almacena únicamente los datos de configuración de la aplicación.

Adjunto un pequeño esquema para entender mejor su funcionamiento:



5.3. Codificación

Para su codificación se ha utilizado el lenguaje de programación de typescript, junto a este se han utilizado diferentes librerías de javascript.

Estas librerías son: leaflet que ha permitido el uso del mapa interactivo incorporado en la aplicación y la librería de encriptación de contraseñas XXXXX que nos ha brindado gran ayuda a la hora de la encapsulación de la información.

A la hora de codificar se ha tenido en cuenta la explicación del código en todo momento, los nombres de las variables y de los métodos creados tienen su concordancia entre sí.

Se ha seguido el patrón de diseño de MVC para un mejor rendimiento y fácil estructuración del código.

5.4. Pruebas

INFORMACIÓN GENERAL	
Identificador de caso	1
Nombre de caso	Añadir alerta
Descripción	El usuario añade alerta
PRUEBA	
Requisitos	Sesión iniciada
Instrucciones	El usuario debe pulsar el botón añadir alerta y cuando complete el formulario de añadir debe haber creado una alerta.
Resultado	Crea alerta
Criterios de aceptación	Crea alerta

INFORMACIÓN GENERAL	
Identificador de caso	2
Nombre de caso	Añadir alerta sin conexión
Descripción	El usuario añade alerta sin conexión a internet
PRUEBA	
Requisitos	No tener acceso a internet y usuario creado
Instrucciones	El usuario debe pulsar el botón añadir alerta y cuando complete el formulario de añadir no debe haber creado una alerta.
Resultado	No crea alerta, abre toast con la información de que debe conectarse a internet
Criterios de aceptación	No crea alerta

INFORMACIÓN GENERAL	
Identificador de caso	3
Nombre de caso	Añadir alerta sin gps
Descripción	El usuario añade alerta sin gps
PRUEBA	

Requisitos	No tener acceso a gps y usuario creado
Instrucciones	El usuario debe pulsar el botón añadir alerta y cuando complete el formulario de añadir debe haber creado una alerta.
Resultado	Crea alerta si pulsa sobre el mapa, si es con localización gps abre toast con la información de que debe conectar el gps
Criterios de aceptación	Si crea con mapa, no crea con gps

INFORMACIÓN GENERAL	
Identificador de caso	4
Nombre de caso	Registrarse
Descripción	Crear un nuevo usuario
PRUEBA	
Requisitos	No tener usuario o cerrar sesión
Instrucciones	El nuevo usuario debe pulsar register y crear un nuevo usuario para la aplicación
Resultado	Crea el usuario
Criterios de aceptación	Crea el usuario

INFORMACIÓN GENERAL	
Identificador de caso	5
Nombre de caso	Registrarse con el mismo correo
Descripción	Crear un nuevo usuario con un correo ya en uso
PRUEBA	
Requisitos	No tener usuario o cerrar sesión
Instrucciones	El nuevo usuario debe pulsar register y crear un nuevo usuario para la aplicación con un correo ya en uso
Resultado	No crea el usuario
Criterios de aceptación	No crea el usuario

INFORMACIÓN GENERAL	
Identificador de caso	6
Nombre de caso	Registrarse con algún campo vacio
Descripción	Crear un nuevo usuario con algunos de los campos vacios del formulario
PRUEBA	
Requisitos	No tener usuario o cerrar sesión
Instrucciones	El nuevo usuario debe pulsar register y crear un nuevo usuario para la aplicación con uno o varios campos vacíos
Resultado	No permite crear el usuario, sale toast con información para que rellene campos
Criterios de aceptación	No crea el usuario

INFORMACIÓN GENERAL	
Identificador de caso	7
Nombre de caso	Login con usuario
Descripción	Iniciar sesión en la aplicación con un usuario ya creado
PRUEBA	
Requisitos	No tener usuario o cerrar sesión
Instrucciones	El usario rellena los campos del formulario de inicio de sesión y pulsa login
Resultado	Login en la app
Criterios de aceptación	Inicia sesión

INFORMACIÓN GENERAL	
Identificador de caso	8
Nombre de caso	Login con usuario no creado
Descripción	Iniciar sesión en la aplicación con un usuario no creado
PRUEBA	
Requisitos	No tener usuario o cerrar sesión
Instrucciones	El usario rellena los campos del formulario de inicio de sesión y pulsa login
Resultado	No hace login en la app
Criterios de aceptación	No inicia sesión

INFORMACIÓN GENERAL	
Identificador de caso	9
Nombre de caso	Modo noche
Descripción	El usuario activa modo noche
PRUEBA	
Requisitos	Tener usuario logueado
Instrucciones	El usuario debe pulsar el modo noche manual y activarse
Resultado	Cambia de tema correctamente
Criterios de aceptación	Cambia el tema

INFORMACIÓN GENERAL	
Identificador de caso	10
Nombre de caso	Modo noche automático
Descripción	El usuario activa modo noche automático
PRUEBA	
Requisitos	Tener usuario logueado
Instrucciones	El usuario debe pulsar el modo noche automático y cuando detecte que la luminosidad es baja cambiar el tema
Resultado	Cambia de tema correctamente
Criterios de aceptación	Cambia el tema

5.5. Problemas encontrados

Han sido varios problemas lo que me han llevado a tomar decisiones de no incluir alguna funcionalidad o de mejorar alguna ya incluida.

- Compatibilidad del sistema del almacenamiento nativo: Este problema ha desembarcado a que las funcionalidades principales se hayan acortado en su desarrollo web. El almacenamiento nativo de Android e IOS no es compatible con el web por lo tanto queda pendiente para su futura actualización el incorporar un sistema de almacenamiento también para su despliegue en web. Esto se descubrió en la ultima semana de entrega, por lo que portar toda la app a web era bastante arriesgado.

- Modo noche: El modo noche es una de las características mas importantes para esta aplicación. Esta aplicación utiliza el sensor de luz para realizar el cambio de tema automático, detectando así la luminosidad de la sala y adaptándose al ojo humano. Pero no todo es tan sencillo, la coordinación entre el modo automático y el modo manual es bastante liosa, aunque el usuario solo ve que cuando se activa el modo automático se desactiva el manual y que, si activo el manual cambio de tema, por la parte de la aplicación se requieren de muchas comprobaciones y de activar o apagar sensores, para que esto ocurra a la perfección. Se dio solución almacenando el valor de cada botón cada vez que el usuario los pulsaba. Y así se iban comprobando estos valores para saber que tema activar o si se debía bloquear el botón de manual o desbloquearlo.

- GeocodeReverse: Esta funcionalidad mejora la lectura de las alertas para el usuario. Al principio se mostraban únicamente las coordenadas donde estaba el hecho ocurrido, pero esta funcionalidad ha mejorado esto haciendo que se pueda saber donde ha ocurrido la alerta solo leyendo el aviso, ya que nos brinda la posibilidad de intercambiar esas coordenadas por el nombre de la ciudad y la calle donde ha ocurrido. El problema fue implementarlo, ya que en la API de Ionic estaba y esta mal como hay que utilizarlo, porque los métodos que ellos proponen de uso ya no existen y te las tienes que ingeniar para meterte en todas las clases que otorga este componente. Se soluciono tras varios días investigando las clases y métodos del componente hasta que se dio con el uso correcto.

6. Manual de instalación

Para su instalación únicamente, necesitaremos de un terminal Android con versión superior a Android 4.1.2. luego se descargará a través de este enlace http://zerod.epizy.com

y se procederá a su instalación.

También podemos ir a esta dirección https://zerod-66498.web.app donde podremos probar la versión web.

7. Manual de usuario

Si tienes alguna duda de cómo funciona la aplicación, adjunto el enlace a unos tutoriales para que aprendas mejor su funcionamiento y así se exprimas al máximo sus funcionalidades

http://zerod.epizy.com/manual.html

8. Valoraciones y conclusión

8.1. Evaluación del grado de cumplimiento de los objetivos y finalidad

El cumplimiento de objetivos, a mi parecer ha sido más que superado en la mayoría de los aspectos.

Se han implementado funciones novedosas tales como:

- El registro de usuarios.
- Visualización de cuantas alertas tiene activas el usuario.
- Limitación de crear alertas si no estas registrado.
- Rango de alertas que el usuario quiere visualizar en la pantalla de alertas.
- Modo noche de la aplicación tanto automático, como el modo manual.
- Se han añadido nuevas alertas, cada una con su icono personalizado.
- Se ha mejorado la visualización de las alertas en las listas.
- Se ha implementado el GeocoderReverse, que al facilitarle unas coordenadas nos las traduce a la dirección (nombre de la calle, ciudad, estado, etc.).

Funciones que no se han implementado:

- Guardar eventos de forma local cuando no hay conexión a internet, para subirlos a firebase cuando se recupere (se ha decidido prescindir de esta función porque a casos prácticos no le veo utilidad alguna).
- Sistema de notificación cuando se cree una nueva alerta (se ha prescindido de esta función, debido que han surgido muchos problemas a la hora de realizar el cambio automático y manual del modo noche de la aplicación)

8.2. Evaluación de la planificación temporal y de la toma de decisiones

La planificación temporal se ha sobre pasado por 5 días, los motivos han sido varios:

El primero fue un problema que se encontró con la documentación de IONIC en el componente nativo de GeoCoder. La documentación estaba obsoleta y los métodos utilizados más los imports de las clases utilizadas no estaban. Habían sido reemplazados por otros.

Este componente se ha utilizado para hacer el cambio de latitud y longitud hacia el nombre completo de la localización, como es el nombre de la ciudad y la calle.

Para dar solución a este problema tuve que investigar en todas las clases que el GeoCoder traía, pero, aunque se haya invertido mucho tiempo el resultado ha sido bueno, ya que así se ha hecho más legible la localización de la alerta.

El segundo problema que ha hecho que se retrase unos días más la finalización del proyecto ha sido el theming de la aplicación, el añadir el cambio de tema dinámico y manual ha sido todo un quebradero de cabeza, pero estoy orgulloso con el resultado.

En general se ha sobrepasado un poco el límite estimado por mí al comienzo del proyecto, porque su finalización estaba prevista para el 16 de mayo, pero ha concluido el 21 de mayo, pero se han concluido todos los puntos a tratar con éxito.

8.3. Posibles mejoras a la solución

Aspectos que se podrían mejorar:

El plugin de leaflet, el encargado de mostrar el mapa, podría ser cambiado por el de Google Maps, ya que brinda al usuario una interfaz más amigable y más familiarizada. Por qué no se usó esta tecnología fue simple y llanamente el coste de cada una de las tecnologías que nos brindan el mapa, leaflet con un coste de 0€ y por el contrario gmaps una tarifa plana que puede llegar a costar los 200$

Otro añadido más sería brindar la posibilidad de cambiar la foto de perfil del usuario, que no se ha implementado en esta versión.

8.4. Conclusión final

En conclusión, se ha desarrollado un proyecto muy completo, ya que brinda al usuario la posibilidad de utilizar el mismo servicio en varias diferentes plataformas.

La aplicación cumple a mi parecer con una complejidad media, no es difícil de entender, pero si compleja a la hora de enlazar unos componentes nativos con otros y a la hora de estructurar la inicialización de algunos elementos.

Por otro lado, me hubiera gustado implementar 2 o 3 mejoras más que hubieran dado un atractivo más a la aplicación, pero en estos momentos no ha sido posible por falta de fondos para financiar esos aspectos.

En las próximas versiones se irán ampliando dichas funcionalidades y su próxima salida al mercado en Google Play Store.

9. Bibliografía

Se ha con sultado la API de Ionic para hacer uso de los siguientes componentes Nativos:

- android-permissions
- camera
- diagnostic
- geolocation
- native-geocoder
- native-storage
- network
- sensors

- splash-screen
- status-bar

Se ha consutado la API de Ionic para hacer uso de los siguientes componentes Visuales:

- ion-slides
- ion-range
- ion-item
- ion-list
- ion-tab
- ion-button
- ion-fab
- ion-fab-button
- ion-input
- ion-textarea
- ion-nav
- ion-radiobutton
- ion-select
- ion-backbutton

Se ha consultado la API de Leaflet para hacer uso del mapa.

Se ha consultado la librería CryptoJS para la encriptación de claves.

10. Código fuente

10.1 Custom loading

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LoadingController } from '@ionic/angular';

import { environment } from 'src/environments/environment';




@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
/**
 * Clase que crea nuestro loading personalizado
 */
export class CustomLoading {
  myloading:any;
  timeout;
```

```
  constructor(private loadingController:LoadingController){
  }

  /**
   * metodo que muestra el loading
   * @param msg mensaje a mostrar
   */
  async show(msg) {
    this.myloading = await this.loadingController.create({
    message:msg,
    spinner: null,
    leaveAnimation:null
    });
    this.timeout=setTimeout(()=>{
      this.myloading.dismiss();
    },environment.timemaxloading);
    await this.myloading.present();
  }

  /**
   * metodo que oculta el loading
   */
  hide(){
    if(this.myloading){
      clearTimeout(this.timeout);
      this.myloading.dismiss();
    }
  }
}
```

10.2 Custom-modal

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ModalController } from '@ionic/angular';



@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
/**
 * Clase que nos crea los modales personalizados
 */
export class CustomModalModule {
```

```
  public changes = false;

  constructor(public modalCtrl: ModalController,) { }
/**
 * Metodo que nos crea el modal de add alert
 * recibe el componente add alert, la longitud y latitud y la propia
clase donde se llama
 * @param component recibel el modal
 * @param latitude latitud de la ubicacion
 * @param longitude longitud de la ubicacion
 * @param callback accion que ejecuta al cerrar el modal
 */
  async show(component, latitude,longitude, callback): Promise <any> {

    const modal = await this.modalCtrl.create({
      cssClass: "my-modal",
      backdropDismiss: true,
      component: component,  //El componente que se inyecta en la ventana
modal
      componentProps: {latitude,longitude} //Los parámetros que se le
pasan a la ventana modal
    });

    modal.onDidDismiss().then((d) => {

        if (callback.onModalClose)
          callback.onModalClose()
    });


    return await modal.present();
  }

/**
 * Metodo que nos crea el modal de add alert
 * recibe el componente add alert, la info de la alerta y la propia clase
donde se llama
 * @param component componente a mostrar viewalert
 * @param descripcion descripcion del aviso
 * @param tipo tipo de aviso
 * @param hora fecha del aviso
 * @param latitud latitud de la ubicacion del aviso
 * @param longitud longitud de la ubicacion del aviso
 * @param callback lo que hace la cerrse el modal (this)
 */
  async showInfo(component, descripcion, tipo, hora, ciudad, calle,
latitud,longitud ,callback): Promise <any>{

    const modal = await this.modalCtrl.create({
```

```
      cssClass: "mas-info",
      backdropDismiss: true,
      component: component,  //El componente que se inyecta en la ventana
modal
      componentProps: {descripcion,tipo,hora, ciudad, calle,latitud,
longitud} //Los parámetros que se le pasan a la ventana modal
    });
    modal.onDidDismiss().then((d) => {
        if (callback.onModalClose)
          callback.onModalClose()
    });


    return await modal.present();



  }

  /**
   * muestra el modal de ayuda
   * @param component recibe el componente de la ayuda
   * @param callback this
   */
  async showHelp(component, callback): Promise <any>{

    const modal = await this.modalCtrl.create({
      cssClass: "show-help",
      backdropDismiss: true,
      component: component,  //El componente que se inyecta en la ventana
modal
    });
    modal.onDidDismiss().then((d) => {
        if (callback.onModalClose)
          callback.onModalClose()
    });


    return await modal.present();



  }

  /**
   * muestra el modal de registro
   * @param component recibe el componente del home
   * @param callback this
   */
```

```
  async showRegister(component, callback): Promise <any>{

    const modal = await this.modalCtrl.create({
      cssClass: "show-register",
      backdropDismiss: true,
      component: component,  //El componente que se inyecta en la ventana
modal
    });
    modal.onDidDismiss().then((d) => {
        if (callback.onModalClose)
          callback.onModalClose()
    });


    return await modal.present();



  }

}
```

10.3 Custom Toast

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ToastController } from '@ionic/angular';

@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
/**
 * Clase que nos crea un toast personalizado
 */
export class CustomToast {
  toast;
  constructor(private toastCtrl: ToastController) { }
/**
 * metodo que nos crea el toast en la parte inferior
 * @param msg mensaje a mostrar
 */
  async show(msg) {
    const toast = await this.toastCtrl.create({
      message: msg,
      showCloseButton: true,
      position: 'bottom',
```

```
      closeButtonText: 'Ok',
      duration: 5000
    });
    toast.present();
  }

  /**
   * metodo que nos muestra el toast en la parte de arriba
   * @param msg mensaje a mostrar
   * @param time tiempo en el que se muestra
   */
  async showTop(msg, time?) {
    if (this.toast)
      this.toast.dismiss();
    if (!time) {
      this.toast = await this.toastCtrl.create({
        message: msg,
        showCloseButton: true,
        position: 'top',
        closeButtonText: 'Ok',
      });
      this.toast.present();
    } else {
      this.toast = await this.toastCtrl.create({
        message: msg,
        showCloseButton: false,
        position: 'top',
        closeButtonText: 'Ok',
        duration: time
      });
      this.toast.present();
    }
  }
}
```

10.4 Add alert

- HTML

```
<ion-header>
  <ion-toolbar>
      <ion-icon class="cerrar" name="close-circle-outline"
(click)="cancel()" ></ion-icon>
  <ion-title style="color:var(--ion-color-light-contrast)">{{ "addModal-
H" | translate }}</ion-title>
  </ion-toolbar>
  </ion-header>
  <ion-content padding>
  <form [formGroup]="alerta" (ngSubmit)="uploadForm()">
```

```
    <h6>{{ "addModal-D" | translate }}</h6>
    <ion-item style="--background: transparent !important;">
        <textarea style="color:var(--ion-color-dark)" #myInput
id="myInput" rows="1" maxLength="300" formControlName="descripcion"
placeholder="{{ 'input' | translate}}" (keyup)="resize()" ></textarea>
    </ion-item>
  <ion-item style="--background: transparent !important;">
      <ion-label style="color:var(--ion-color-dark)">{{ "addModal-T" |
translate }}</ion-label>
      <ion-select [interfaceOptions]="customActionSheetOptions"
formControlName="alertType" interface="action-sheet" placeholder="Select
One">
          <ion-select-option value="accidente">{{ "accident" |
translate }}</ion-select-option>
          <ion-select-option value="control">{{ "control" |
translate }}</ion-select-option>
          <ion-select-option value="radar">{{ "radar" |
translate }}</ion-select-option>
          <ion-select-option value="atasco">{{ "atasco" |
translate }}</ion-select-option>
          <ion-select-option value="obras">{{ "obras" |
translate }}</ion-select-option>
          <ion-select-option value="lluvia">{{ "rain" |
translate }}</ion-select-option>
          <ion-select-option value="nieve">{{ "snow" | translate }}</ion-
select-option>
          <ion-select-option value="viento">{{ "wind" |
translate }}</ion-select-option>
          <ion-select-option value="olas">{{ "olas" | translate }}</ion-
select-option>
          <ion-select-option value="niebla">{{ "niebla" |
translate }}</ion-select-option>

      </ion-select>
    </ion-item>
  <ion-button shape="round" style="color:var(--ion-color-primary-shade)"
fill="outline"
  type="submit" [disabled]="!alerta.valid"><ion-icon name="md-
send"></ion-icon></ion-button>
  </form>
  </ion-content>
```

- SCSS

```
ion-button{
    float: right;
```

```
}

ion-item{
    margin-bottom: 10px;
}

ion-input{
    margin-top: 5px;
}

ion-toolbar{
    --ion-toolbar-background:var(--ion-color-primary);
}

ion-title{
    text-transform: uppercase;
    color:var(--ion-color-primary-contrast);
}


#myInput {
    width: calc(100% - 10px);
    border: 0;
    border-radius: 0;
    background: transparent;
}

.cerrar{
    float: right;
    margin-top: -10px;
    margin-right: 3px;
    color:var(--ion-color-light-contrast);
    width: 30px;
    height: 30px;
}

h6{
    color:var(--ion-color-dark);
}


ion-content{
    --background: var(--ion-color-light-shade);
}
```

- TS

```
import { NetworkService } from './../../servicios/network.service';
```

```
import { TranslateService } from '@ngx-translate/core';
import { CustomToast } from './../../custom-modal/custom-toast';
import { CustomLoading } from './../../custom-modal/custom-loading';
import { CloudserviceService } from
'../../servicios/cloudservice.service';
import { ModalController, NavParams } from '@ionic/angular';
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { NativeStorage } from '@ionic-native/native-storage/ngx';
// tslint:disable-next-line: max-line-length
import { NativeGeocoder, NativeGeocoderOptions, NativeGeocoderResult }
from '@ionic-native/native-geocoder/ngx';



@Component({
  selector: 'app-add-alert',
  templateUrl: './add-alert.component.html',
  styleUrls: ['./add-alert.component.scss']
})

export class AddAlertComponent implements OnInit {
  @ViewChild('myInput') myInput: ElementRef;
  public alerta: FormGroup;
  longitud: any;
  latitud: any;
  customActionSheetOptions: any = {
    header: this.translate.instant('Select'),
    subHeader: this.translate.instant('TypeS'),
  };
  usuario: String;
  imgUsuario: String;
  options: NativeGeocoderOptions = {
    useLocale: true,
    maxResults: 5
};
  ngOnInit(): void {
  }


  constructor(public modalcontroller: ModalController,
    private formBuilder: FormBuilder,
    private CloudS: CloudserviceService,
    public loading: CustomLoading,
    public navparams:NavParams,
    public toast: CustomToast,
    private translate: TranslateService,
    private netwoekS: NetworkService,
```

```
    private nativeStorage: NativeStorage,
    private nativeGeocoder: NativeGeocoder ) {

      // recuperamos el usuario
    this.nativeStorage.getItem('user').then(e => {

        this.usuario = e.email.toString();
        this.imgUsuario = e.img.toString();


        console.log(this.usuario);

      }).catch(error => {

        console.log(error);

      });

    // recuperamos a traves de NavParams, la clave valor que tenemos en
la marca
    this.longitud = this.navparams.get('longitude');
    this.latitud = this.navparams.get('latitude');

    this.alerta = this.formBuilder.group({
      alertType: ['', Validators.required],
      descripcion: [''],
    });

  }

  /**
   * metodo que establece la refactorizacion del text area
   */
  resize() {
    this.myInput.nativeElement.style.height =
this.myInput.nativeElement.scrollHeight + 'px';
}



  /**
   * método que quita el modal
   */
  cancel() {
    this.modalcontroller.dismiss();
  }

  /* Se ejecuta al submit el formulario. Crea un objeto proveniente del
  formulario  y llama a la función anadir del
```

```
servicio.
Antes de ejecutar el submit comprobamos si tenemos conexion a internet
Si no tenemos salta un toast para que le demos internet
Si tenemos internet nos permite hacer el submit */
uploadForm() {
  if (this.netwoekS.previousStatus === 1) {
    this.toast.show(this.translate.instant('noNetwork'));
  } else if (this.netwoekS.previousStatus === 0) {

    this.reverseCoor(this.latitud,this.longitud).then(result =>{

      let data = {
        descripcion: this.alerta.get('descripcion').value,
        alert: this.alerta.get('alertType').value,
        longitud: this.longitud,
        latitud: this.latitud,
        hora: new Date().valueOf(),
        user: this.usuario,
        imgU: this.imgUsuario,
        icon: '',
        areaAdministrativa: result[0]['administrativeArea'],
        codigoPostal: result[0]['postalCode'],
        ciudad: result[0]['locality'],
        calle: result[0]['thoroughfare'],
        numCalle: result[0]['subThoroughfare'],

      };



      switch (data.alert) {
        case 'accidente': {
           data.icon = 'car';
           this.addAccident(data);
           break;
        }
        case 'control' : {
          data.icon = 'hand';
          this.addAccident(data);
           break;
        }
        case 'radar'  : {
          data.icon = 'speedometer';
          this.addAccident(data);
          break;
        }
        case 'atasco' : {
          data.icon = 'hourglass';
          this.addAccident(data);
```

```
          break;
        }
        case 'obras' : {
          data.icon = 'hammer';
          this.addAccident(data);
         break;
        }
        case 'lluvia' : {
          data.icon = 'rainy';
          this.addMeteorology(data);
          break;
        }
        case 'nieve' : {
          data.icon = 'snow';
          this.addMeteorology(data);
          break;
        }
        case 'viento' : {

          data.icon = 'swap';
          this.addMeteorology(data);
          break;
        }
        case 'olas' : {

          data.icon = 'boat';
          this.addMeteorology(data);
          break;
        }
        case 'niebla' : {

          data.icon = 'cloudy';
          this.addMeteorology(data);
          break;
        }
        default: {
          this.toast.show(this.translate.instant('errorloading'));
           break;
        }
     }
    }
      );
  }
}

/**
 * Metodo que añade alertas de accidentes
 */
addAccident(data) {
```

```
    /* Mostramos el cargando... */
    this.loading.show('');
    // Llamamos al metodo anadir pasandole  los datos
    this.CloudS.anadirA(data)
      .then((docRef) => {
        /* Cerramos el cargando...*/
       this.loading.hide();
        /*Cerramos el modal*/
        this.cancel();
      })
      .catch((error) => {
        /* Cerramos el cargando...*/
        this.loading.hide();
        this.toast.show(this.translate.instant('errorloading'));

      });

  }

  /**
   * Metodo que añade alertas de meteorología
   */
  addMeteorology(data){
    /* Mostramos el cargando... */
    this.loading.show('');
    // Llamamos al metodo anadir pasandole  los datos
    this.CloudS.anadirM(data)
      .then((docRef) => {
        /* Cerramos el cargando...*/
        this.loading.hide();
        /*Cerramos el modal*/
        this.cancel();
      })
      .catch((error) => {
        /* Cerramos el cargando...*/
        this.loading.hide();
        this.toast.show(this.translate.instant('errorloading'));
      });
  }

/**
 * Devuelve la ubicación de la alerta
 * @param lat latitud
 * @param long longitud
 */
reverseCoor(lat,long): Promise<NativeGeocoderResult[]>{
 return this.nativeGeocoder.reverseGeocode(lat, long, this.options)
   }
```

```
}
```

10.5 Help

- HTML

```
<ion-header color="primary">
  <ion-toolbar color="primary">
      <ion-icon name="arrow-back" (click)="onModalClose()"></ion-icon>
    <ion-title>
      ZeroD
    </ion-title>
  </ion-toolbar>

</ion-header>

<ion-content >

    <ion-slides pager="true" [options]="slidesOpts">
    <ion-slide >
          <ion-img src="assets/help/h1.png"></ion-img>
          <ion-label>{{ "h1" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
        <ion-img src="assets/help/h2.png"></ion-img>
        <ion-label>{{ "h2" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
        <ion-img src="assets/help/h4.png"></ion-img>
        <ion-label>{{ "h4" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
        <ion-img src="assets/help/h5.png"></ion-img>
        <ion-label>{{ "h5" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
          <ion-img src="assets/help/h6.png"></ion-img>
          <ion-label>{{ "h6" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
          <ion-img src="assets/help/h7.png"></ion-img>
          <ion-label>{{ "h7" | translate }}</ion-label>
    </ion-slide>
    <ion-slide>
            <ion-img src="assets/help/h8.png"></ion-img>
            <ion-label>{{ "h8" | translate }}</ion-label>
      </ion-slide>
```

```
  </ion-slides>

</ion-content>
```

- SCSS

```
ion-slides{
  height: 100%;
}



ion-title{
  float: left;
  margin-right: 10px;
  margin-top: 5px;
  color: var(--ion-color-light-contrast);
}
ion-img{
  width: 60%;
  height: 60%;
  display:block;
  margin:auto;
  margin-top:10px;

}

ion-label{
  position: absolute;
  bottom: 10%;
  left: 8px;
  right: 8px;
}

ion-icon{
  color: var(--ion-color-light-contrast);
  height: 30px;
  width: 30px;
  margin-right: 3%;
  float: left;

}
```

- TS

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { IonSlides, ModalController } from '@ionic/angular';
import * as $ from 'jquery';


@Component({
  selector: 'app-help',
  templateUrl: './help.component.html',
  styleUrls: ['./help.component.scss']
})
export class HelpComponent implements OnInit {
  slidesOpts = {
    grabCursor: true,
    cubeEffect: {
      shadow: true,
      slideShadows: true,
      shadowOffset: 20,
      shadowScale: 0.94,
    },
    on: {
      beforeInit: function() {
        const swiper = this;

swiper.classNames.push(`${swiper.params.containerModifierClass}cube`);

swiper.classNames.push(`${swiper.params.containerModifierClass}3d`);

        const overwriteParams = {
          slidesPerView: 1,
          slidesPerColumn: 1,
          slidesPerGroup: 1,
          watchSlidesProgress: true,
          resistanceRatio: 0,
          spaceBetween: 0,
          centeredSlides: false,
          virtualTranslate: true,
        };

        this.params = Object.assign(this.params, overwriteParams);
        this.originalParams = Object.assign(this.originalParams,
overwriteParams);
      },
      setTranslate: function() {
        const swiper = this;
        const {
          $el, $wrapperEl, slides, width: swiperWidth, height:
swiperHeight, rtlTranslate: rtl, size: swiperSize,
```

```
        } = swiper;
        const params = swiper.params.cubeEffect;
        const isHorizontal = swiper.isHorizontal();
        const isVirtual = swiper.virtual &&
swiper.params.virtual.enabled;
        let wrapperRotate = 0;
        let $cubeShadowEl;
        if (params.shadow) {
          if (isHorizontal) {
            $cubeShadowEl = $wrapperEl.find('.swiper-cube-shadow');
            if ($cubeShadowEl.length === 0) {
              $cubeShadowEl = swiper.$('<div class="swiper-cube-
shadow"></div>');
              $wrapperEl.append($cubeShadowEl);
            }
            $cubeShadowEl.css({ height: `${swiperWidth}px` });
          } else {
            $cubeShadowEl = $el.find('.swiper-cube-shadow');
            if ($cubeShadowEl.length === 0) {
              $cubeShadowEl = $('<div class="swiper-cube-
shadow"></div>');
              $el.append($cubeShadowEl);
            }
          }
        }

        for (let i = 0; i < slides.length; i += 1) {
          const $slideEl = slides.eq(i);
          let slideIndex = i;
          if (isVirtual) {
            slideIndex = parseInt($slideEl.attr('data-swiper-slide-
index'), 10);
          }
          let slideAngle = slideIndex * 90;
          let round = Math.floor(slideAngle / 360);
          if (rtl) {
            slideAngle = -slideAngle;
            round = Math.floor(-slideAngle / 360);
          }
          const progress = Math.max(Math.min($slideEl[0].progress, 1), -
1);
          let tx = 0;
          let ty = 0;
          let tz = 0;
          if (slideIndex % 4 === 0) {
            tx = -round * 4 * swiperSize;
            tz = 0;
          } else if ((slideIndex - 1) % 4 === 0) {
            tx = 0;
```

```
            tz = -round * 4 * swiperSize;
          } else if ((slideIndex - 2) % 4 === 0) {
            tx = swiperSize + (round * 4 * swiperSize);
            tz = swiperSize;
          } else if ((slideIndex - 3) % 4 === 0) {
            tx = -swiperSize;
            tz = (3 * swiperSize) + (swiperSize * 4 * round);
          }
          if (rtl) {
            tx = -tx;
          }

           if (!isHorizontal) {
            ty = tx;
            tx = 0;
          }

           const transform$$1 = `rotateX(${isHorizontal ? 0 : -
slideAngle}deg) rotateY(${isHorizontal ? slideAngle : 0}deg)
translate3d(${tx}px, ${ty}px, ${tz}px)`;
          if (progress <= 1 && progress > -1) {
            wrapperRotate = (slideIndex * 90) + (progress * 90);
            if (rtl) wrapperRotate = (-slideIndex * 90) - (progress *
90);
          }
          $slideEl.transform(transform$$1);
          if (params.slideShadows) {
            // Set shadows
            let shadowBefore = isHorizontal ? $slideEl.find('.swiper-
slide-shadow-left') : $slideEl.find('.swiper-slide-shadow-top');
            let shadowAfter = isHorizontal ? $slideEl.find('.swiper-
slide-shadow-right') : $slideEl.find('.swiper-slide-shadow-bottom');
            if (shadowBefore.length === 0) {
              shadowBefore = swiper.$(`<div class="swiper-slide-shadow-
${isHorizontal ? 'left' : 'top'}"></div>`);
              $slideEl.append(shadowBefore);
            }
            if (shadowAfter.length === 0) {
              shadowAfter = swiper.$(`<div class="swiper-slide-shadow-
${isHorizontal ? 'right' : 'bottom'}"></div>`);
              $slideEl.append(shadowAfter);
            }
            if (shadowBefore.length) shadowBefore[0].style.opacity =
Math.max(-progress, 0);
            if (shadowAfter.length) shadowAfter[0].style.opacity =
Math.max(progress, 0);
          }
        }
        $wrapperEl.css({
```

```
          '-webkit-transform-origin': `50% 50% -${swiperSize / 2}px`,
          '-moz-transform-origin': `50% 50% -${swiperSize / 2}px`,
          '-ms-transform-origin': `50% 50% -${swiperSize / 2}px`,
          'transform-origin': `50% 50% -${swiperSize / 2}px`,
        });

         if (params.shadow) {
          if (isHorizontal) {
            $cubeShadowEl.transform(`translate3d(0px, ${(swiperWidth / 2)
+ params.shadowOffset}px, ${-swiperWidth / 2}px) rotateX(90deg)
rotateZ(0deg) scale(${params.shadowScale})`);
          } else {
            const shadowAngle = Math.abs(wrapperRotate) -
(Math.floor(Math.abs(wrapperRotate) / 90) * 90);
            const multiplier = 1.5 - (
              (Math.sin((shadowAngle * 2 * Math.PI) / 360) / 2)
              + (Math.cos((shadowAngle * 2 * Math.PI) / 360) / 2)
            );
            const scale1 = params.shadowScale;
            const scale2 = params.shadowScale / multiplier;
            const offset$$1 = params.shadowOffset;
            $cubeShadowEl.transform(`scale3d(${scale1}, 1, ${scale2})
translate3d(0px, ${(swiperHeight / 2) + offset$$1}px, ${-swiperHeight / 2
/ scale2}px) rotateX(-90deg)`);
          }
        }

        const zFactor = (swiper.browser.isSafari ||
swiper.browser.isUiWebView) ? (-swiperSize / 2) : 0;
        $wrapperEl
          .transform(`translate3d(0px,0,${zFactor}px)
rotateX(${swiper.isHorizontal() ? 0 : wrapperRotate}deg)
rotateY(${swiper.isHorizontal() ? -wrapperRotate : 0}deg)`);
      },
      setTransition: function(duration) {
        const swiper = this;
        const { $el, slides } = swiper;
        slides
          .transition(duration)
          .find('.swiper-slide-shadow-top, .swiper-slide-shadow-
right, .swiper-slide-shadow-bottom, .swiper-slide-shadow-left')
          .transition(duration);
        if (swiper.params.cubeEffect.shadow && !swiper.isHorizontal()) {
          $el.find('.swiper-cube-shadow').transition(duration);
        }
      },
    }
  }
```

```
  @ViewChild('SwipedTabsSlider') SwipedTabsSlider: IonSlides;


  constructor(public modalcontroller: ModalController) { }

  ngOnInit() {

  }
/**
 * Cierra el modal
 */
 onModalClose(){
  this.modalcontroller.dismiss();
 }

}
```

10.6 Register

- HTML

```
<ion-content padding >

    <ion-avatar style="text-align: center;margin:auto;cursor: pointer;
width:200px; height:200px;"  (click)="newPic()">
        <img src="../../../assets/icon/avatar.png" *ngIf="default"
width="300px" height="300px" style="border:2px solid #fff">
      <img [src]="imageV" *ngIf="image" style="height:120px; width:110px"
style="border:2px solid #fff">
    </ion-avatar>

  <form [formGroup]="register" (ngSubmit)="registerForm()">

    <ion-item>
      <ion-label position="floating" >Nombre de Usuario</ion-label>
      <ion-input type="text" formControlName="user"></ion-input>
    </ion-item>

    <ion-item>
      <ion-label position="floating" >Correo</ion-label>
      <ion-input type="email" formControlName="email" required></ion-
input>
    </ion-item>

    <ion-item>
      <ion-label position="floating" >Contraseña</ion-label>
      <ion-input type="password" formControlName="pass"></ion-input>
    </ion-item>

    <ion-item>
```

```
      <ion-label position="floating" >Repite la contraseña</ion-label>
      <ion-input type="password" formControlName="pass2"></ion-input>
    </ion-item>

    <ion-button expand="full"  color="primary" type="submit"
[disabled]="!register.valid">Crear cuenta</ion-button>

</form>

</ion-content>
```

- TS

```
import { Component, OnInit } from '@angular/core';
import { Camera, CameraOptions } from '@ionic-native/camera/ngx';
import { DomSanitizer } from '@angular/platform-browser';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { CustomLoading } from 'src/app/custom-modal/custom-loading';
import { CustomToast } from 'src/app/custom-modal/custom-toast';
import { TranslateService } from '@ngx-translate/core';
import { NetworkService } from 'src/app/servicios/network.service';
import { CloudserviceService } from
'src/app/servicios/cloudservice.service';
import { ModalController } from '@ionic/angular';
import { Router } from '@angular/router';
import { NativeStorage } from '@ionic-native/native-storage/ngx';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent implements OnInit {

  image;
  imageV;
  default = true;
  public register: FormGroup;
  user = Array;

  constructor(public modalcontroller: ModalController,
    private camera: Camera,
    private sanitizer: DomSanitizer,
    private formBuilder: FormBuilder,
    private netwoekS: NetworkService,
    private translate: TranslateService,
    public toast: CustomToast,
    public loading: CustomLoading,
```

```
    private CloudS: CloudserviceService,
    public router: Router,
    private nativeStorage: NativeStorage
    ) {

      this.register = this.formBuilder.group({
        user: ['', Validators.required],
        email: ['', Validators.compose([
          Validators.required,
          Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-
9-.]+$')
        ])],
        pass: ['', Validators.compose([
          Validators.minLength(5),
          Validators.required,
          Validators.pattern('^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-
9]+$')
        ])],
        pass2: ['', Validators.compose([
          Validators.minLength(5),
          Validators.required,
          Validators.pattern('^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-
9]+$')
        ])],
      });
    }

  ngOnInit() {
  }

  /**
   * Sumbit del formulario de registro
   */
  registerForm() {
    if (this.netwoekS.previousStatus === 1) {
      this.toast.show(this.translate.instant('noNetwork'));
    }else if (this.image == null) {
      this.toast.show(this.translate.instant('image'));
    }
     else if (this.netwoekS.previousStatus === 0) {


      this.CloudS.getProfile(this.register.get('email').value).then(d =>{


        if(d.length > 0){
          this.toast.show(this.translate.instant('email'));
        }else{
          const data = {
```

```
            user: this.register.get('user').value,
            email: this.register.get('email').value,
            pass: this.register.get('pass').value,
            pass2: this.register.get('pass2').value,
          };

          if (data.pass !== data.pass2) {

            this.toast.show(this.translate.instant('failPass'));
          } else {

          this.CloudS.createUser(this.image, data.user, data.email,
data.pass);
          this.router.navigate(['/tabs/tab1']);
          this.cancel();

          }

        }

      })

  }

}

/**
 * Metedo que ejecuta la apertura de la camara y la captura de la foto de
perfil
 */
  newPic() {

    const options: CameraOptions = {
      quality: 100,
      destinationType: this.camera.DestinationType.DATA_URL,  /*FILE_URI
*/
      encodingType: this.camera.EncodingType.JPEG,
      mediaType: this.camera.MediaType.PICTURE,
      cameraDirection: 0,
      correctOrientation: true,
      /* allowEdit:true,*/
      saveToPhotoAlbum: true,
      /*sourceType:0 es library, 1 camera, 2 saved */
      /* targetHeight:200,*/
      targetWidth: 200
    };

    this.camera.getPicture(options).then((imageData) => {
      // imageData is either a base64 encoded string or a file URI
```

```
      // If it's base64 (DATA_URL):
      this.image = ' ' + imageData;
      this.imageV = this.sanitizer.bypassSecurityTrustUrl(this.image);
      this.default = false;
    }, (err) => {
      // Handle error
    });
  }

  /**
   * método que quita el modal
   */
  cancel() {
    this.modalcontroller.dismiss();
  }


}
```

10.7 View Card

- HTML

```
<ion-header >
    <ion-toolbar>
        <ion-icon class="cerrar" name="close-circle-outline"
(click)="cancel()" ></ion-icon>
      <ion-title style="color:var(--ion-color-light-
contrast)" >{{this.alert}}</ion-title>
    </ion-toolbar>
</ion-header>

<ion-content padding style="background-color: var(--ion-color-light-
shade)">
    <h6>{{ "fecha" | translate }}</h6>
    <ion-item style="--background: transparent !important;">
        <ion-label color="dark" class="fecha" style="font-
size:1.22rem">{{this.fechaF}}</ion-label>
      </ion-item>
    <h6>{{ "desc" | translate }}</h6>
    <ion-item style="--background: transparent !important;">
      <ion-label color="dark" class="desc">{{this.descripcion}}</ion-
label>
    </ion-item>
    <h6>{{ "ltlng" | translate }}</h6>
```

```
    <ion-item  style="--background: transparent !important;"
class="ltlng">
      <ion-label color="dark"  style="font-size:1.0rem">{{ "city" |
translate }} {{this.ciudad}} </ion-label>
    </ion-item>
    <ion-item style="--background: transparent !important;"
class="ltlng">
      <ion-label color="dark" class="long" style="font-
size:1.0rem" >{{ "street" | translate }} {{this.calle}}</ion-label>
    </ion-item>

    <ion-fab vertical="bottom" horizontal="end" slot="fixed" >
        <ion-fab-button >

          <ion-icon class="openM" name="compass"
(click)="openMap()"></ion-icon>
        </ion-fab-button>


        </ion-fab>



</ion-content>
```

- SCSS

```
ion-header{
    --ion-header-background:var(--ion-color-primary);

}

    ion-title{
        text-transform: uppercase;

    }

    ion-toolbar{
        --ion-toolbar-background:var(--ion-color-primary);
    }

    .desc{
        white-space: normal;
    }

    .fecha{
        text-overflow: clip;
```

```
}

.ubi{
    white-space: nowrap;
    display: inline-block;
    text-overflow: clip;
    overflow: hidden;
}


.long{

    width: 90%;
}



.cerrar{
    float: right;
    margin-top: -10px;
    margin-right: 3px;
    color: var(--ion-color-light-contrast);
    width: 30px;
    height: 30px;
}

.openM{
    color:var(--ion-color-primary-shade);
}

.ltlng{

    width: 85%;
}

ion-fab-button{
    height: 45px;
    --background: none;
    width: 45px;
    margin-bottom: 10px;
}

h6{
    color: var(--ion-color-dark);

}

ion-content{
    --background: var(--ion-color-light-shade);
```

```
    }
```

- TS

```
import { Router, } from '@angular/router';
import { Component, OnInit } from '@angular/core';
import { ModalController, NavParams } from '@ionic/angular';
import { NavegacionService } from 'src/app/servicios/navegacion.service';


@Component({
  selector: 'app-view-card',
  templateUrl: './view-card.component.html',
  styleUrls: ['./view-card.component.scss']
})
export class ViewCardComponent implements OnInit {

  descripcion: any;
  longitud: any;
  latitud: any;
  hora: any;
  alert: any;
  horaS: string;
  fecha: any;
  fechaF: any;
  ciudad: String;
  calle: String;

  constructor(public modalcontroller: ModalController,public navparams:
    NavParams, public router: Router, private openM: NavegacionService) {

      //al iniciar esta clase recibimos los parametos del navparams que
luego usaremos

      this.descripcion=this.navparams.get('descripcion');
      this.alert=this.navparams.get('tipo');
      this.calle=this.navparams.get('calle');
      this.ciudad=this.navparams.get('ciudad');
      this.latitud=this.navparams.get('latitud');
      this.longitud=this.navparams.get('longitud');
      //Formateo de la fecha
      this.hora= new Date(this.navparams.get('hora'));
```

```
      this.horaS= this.hora.toString();
      this.fecha=this.horaS.split(" ",5)
      this.fechaF=this.fecha[0]+" "+this.fecha[1]+" "+this.fecha[2]+"
"+this.fecha[3]+" "+this.fecha[4]

    }

  ngOnInit() {
  }

  /**
   * metodo que nos redirige hacia el mapa con la ubicacion de esa alerta
   */
  openMap(){
    console.log(this.longitud +"hola")
    this.openM.recibeUbicacion(this.latitud,this.longitud);
    this.openM.cargaMapa(true);
    this.router.navigate(['/tabs/tab2']);
    this.modalcontroller.dismiss();


  }

  /**
   * oculta el modal
   */
  cancel(){

    this.modalcontroller.dismiss();

  }

}
```

10.8 Model

- iAccident

```
export interface iAccidente{

    longitud?:any,
    latitud?:any,
    descripcion?:any,
    alert?:any,
    hora?:any,
    icon?:any,
    user?:any,
    imgU?:any,
```

```
    areaAdministrativa?: any,
    codigoPostal?: any
    ciudad?: any,
    calle?: any,
    numCalle?: any,
  }
```

- iMeteorology

```
export interface iMeteorology{

    longitud?:any,
    latitud?:any,
    descripcion?:any,
    alert?:any,
    hora?:any,
    icon?:any,
    user?:any,
    imgU?:any,
    areaAdministrativa?: any,
    codigoPostal?: any
    ciudad?: any,
    calle?: any,
    numCalle?: any,
  }
```

- iUser

```
export interface iUser{

    user:String,
    email:String,
    img:String,

  }
```

10.9 Register/Login

- HTML

```
<ion-header>

</ion-header>

<ion-content padding >

  <ion-img src="assets/sinavisos.png"></ion-img>
```

```
  <form [formGroup]="login" (ngSubmit)="LoginForm()">
    <ion-item>
      <ion-label>email</ion-label>
      <ion-input type="email"  formControlName="email"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>pass</ion-label>
      <ion-input type="password"  formControlName="pass"></ion-input>
    </ion-item>

    <ion-button expand="full"  color="tertiary" type="submit"
[disabled]="!login.valid">Login</ion-button>
  </form>


  <ion-button expand="full" color="primary"
(click)="openRegister()">Register</ion-button>

</ion-content>
```

- TS

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { CloudserviceService } from '../servicios/cloudservice.service';
import { CustomLoading } from '../custom-modal/custom-loading';
import { NetworkService } from '../servicios/network.service';
import { TranslateService } from '@ngx-translate/core';
import { CustomToast } from '../custom-modal/custom-toast';
import { CustomModalModule } from '../custom-modal/custom-modal.module';
import { RegisterComponent } from
'../customComponent/register/register.component';
import { Router } from '@angular/router';

@Component({
  selector: 'app-register-login',
  templateUrl: './register-login.page.html',
  styleUrls: ['./register-login.page.scss'],
})
export class RegisterLoginPage implements OnInit {
  ngOnInit() {
  }

  public login: FormGroup;
  regexp:any

  constructor(
    private formBuilder: FormBuilder,
```

```
    private CloudS: CloudserviceService,
    public loading: CustomLoading,
    private netwoekS: NetworkService,
    private translate: TranslateService,
    public toast: CustomToast,
    public cmm: CustomModalModule,
    public router: Router
    ) {
      this.regexp = new RegExp('^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$');

      this.login = this.formBuilder.group({
        email: ['', Validators.compose([
          Validators.required,
          Validators.pattern('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-
9-.]+$')
        ])],
        pass: ['', Validators.compose([
          Validators.minLength(5),
          Validators.required,
          Validators.pattern('^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])[a-zA-Z0-
9]+$')
        ])],
      });
  }

/**
 * Metodo que se ejecuta cuando se hace el submit del formulario de login
 */
  LoginForm(){

    if (this.netwoekS.previousStatus === 1) {
      this.toast.show(this.translate.instant('noNetwork'));
    } else if (this.netwoekS.previousStatus === 0) {

      const data = {
        email: this.login.get('email').value,
        pass: this.login.get('pass').value,
      };

      this.loading.show('');
      this.CloudS.loginUser(data.email, data.pass);
      this.loading.hide();
  }

}

/**
 * Metodo que abre el modal del registro de usuarios
```

```
 */
openRegister() {

  this.cmm.showRegister(RegisterComponent, this);
}


}
```

10.10 Servicios

- BACKBUTTONSERVICE

```
import { Injectable } from '@angular/core';
import { Platform, NavController } from '@ionic/angular';
import { Router, NavigationEnd } from '@angular/router';


@Injectable({
  providedIn: 'root'
})
export class BackbuttonService {

  currentURL: any = "";
  back: any;
  pressback: boolean =false;
  openModal = false;

  /*Este servicio controla el comportamiento del botón atrás en la
aplicación */
  constructor(private platform: Platform,private router: Router, private
navCtrl: NavController,){


    this.platform.backButton.subscribe(() => {
      // code that is executed when the user pressed the back button
      if(!this.openModal){
        if (this.currentURL == "/" || this.currentURL == "/tabs/tab1"){

          /*En caso de estar en la pestaña tab1 si se pulsa atrás se
cierra la aplicación*/
          navigator['app'].exitApp();
        }else{

          this.pressback=true;
          this.navCtrl.back();
        }
    }
    });
```

```
    this.router.events.subscribe((event) => {
      if (event instanceof NavigationEnd) {
         //es lanzado cuando se termina de navegar
        this.currentURL = event.url;

        }
    });

  }

  }
```

- CLOUDSERVICE

```
import { iMeteorology } from './../model/iMeteorology';
import { iAccidente } from './../model/iAccident';
import { iUser } from './../model/iUser';
import { Injectable } from '@angular/core';
import { AngularFirestore, AngularFirestoreCollection } from
'angularfire2/firestore';
import { environment } from '../../environments/environment';
import * as firebase from 'firebase';
import * as CryptoJS from 'crypto-js';
import { NativeStorage } from '@ionic-native/native-storage/ngx';
import { Router } from '@angular/router';




@Injectable({
  providedIn: 'root'
})
export class CloudserviceService {
  // Variables para las colecciones de firestore

  accidenteCollection: AngularFirestoreCollection<any>;
  meteorologiaCollection: AngularFirestoreCollection<any>;
  userCollection: AngularFirestoreCollection<any>;
  key= '123456$#@$^@1ERF';


  /*Variables para el infiniteScroll de la pestaña Accidentes*/
  lastAccidentLoaded = null;  // último accidente cargado
  lastlastAccidentLoaded = null;  // último cargado esta vez, si es igual
al anterior, entonces no hay más que cargar
```

```
  scrollAccidentEnabled = true;  // está el infiniteScroll habilitado
porque se haya cumplido lo anterior

  /*Variables para el infiniteScroll de la pestaña Meteorologia*/
  lastMeteorologyLoaded = null;  // último accidente cargado
  lastlastMeteorologyLoaded = null;  // último cargado esta vez, si es
igual al anterior, entonces no hay más que cargar
  scrollMeteorologyEnabled = true;  // está el infiniteScroll habilitado
porque se haya cumplido lo anterior

  isConnected = true;  // saber si estamos con red para realizar
conexiones

  user: any;
  distance: Number;
  location: any;

  constructor(private fireStore: AngularFirestore,
    private nativeStorage: NativeStorage,
    public router: Router,
    ) {
    /* Crea una referencia a la colección que empleamos para realizar las
    operaciones CRUD*/
    this.accidenteCollection =
fireStore.collection<any>(environment.accidenteColeccion);
    this.meteorologiaCollection =
fireStore.collection<any>(environment.meteorologiaColeccion);
    this.userCollection =
fireStore.collection<any>(environment.userColeccion);

  }
////////////////////////////////////////////////////////////USUARIOS//////
////////////////////////////////////////////////////////////////////////
/**
 * Crea un usuario nuevo en la aplicación
 * @param img String foto de perfil
 * @param user String nombre de usuario
 * @param email String correo
 * @param pass String contraseña
 */
  createUser(img, user, email, pass) {

    let datos = {
      img,
      user,
      email
    }

    // creamos el usuario
```

```
    firebase.auth().createUserWithEmailAndPassword(email,
this.set(this.key, pass)).then(e => {
      console.log('Usuario creado');
      // si la respuesta es correcta  lo añadimos al almacenamiento
nativo
      this.nativeStorage.setItem('user', {email: email, img: img , user:
user} )
      .then(
        () => console.log('Stored item!'),
        error => console.error('Error storing item', error)
      );

      return this.userCollection.add(datos).then(e => {

       console.log('Todo ok');
     }).catch(err => {
       console.log(err);
     });

     }

    ).catch(error=> {
      // si la respuesta es incorrecta muestra mensaje error
     var errorCode = error.code;
     var errorMessage = error.message;

     console.log(errorCode + ': ' + errorMessage);
   });

 }

 /**
  * Obtiene el perfil del usuario
  * @param email String identificador unico de cada usuario
  */
 getProfile(email): Promise<iUser[]> {

   return new Promise((resolve) => {
   let lreq: iUser[] = [];
   let query;
   query = this.userCollection.ref.where('email', '==', email).get();

   query.then(snapshot => {

     snapshot.forEach( doc => {
       let x = { 'email': doc.id , ...doc.data()};
       lreq.push(x);
     });
     resolve(lreq);
```

```
    }
      );
  });

  }

  /**
   * Login del usuario
   * @param email String identificador unico de cada usuario
   * @param pass String contraseña
   */
  loginUser(email,pass){
    firebase.auth().signInWithEmailAndPassword(email, this.set(this.key,
pass)).then(e =>{
      this.getProfile(email).then(user => {

        this.nativeStorage.setItem('user', {email: user[0].email, img:
user[0].img, user: user[0].user } )
       .then(
         () => this.router.navigate(['/tabs/tab1']),
         error => console.error('Error storing item', error)
       );
      });
    }).catch(function(error) {
      var errorCode = error.code;
      var errorMessage = error.message;
      console.log(errorCode + ': ' + errorMessage);
    });
  }

  /**
   * obtiene el numero de alertas activas que tiene el usuario
   * @param email identificador del usuario
   * @return Promise con el numero de alertas activas
   */
  getNumberOfAlert(email): Promise<Number> {

    return new Promise((resolve) => {
      let lreqM: iMeteorology [] = [];
      let lreqA: iAccidente [] = [];
      let query;
      let query2;
      var n = 0;

      query = this.meteorologiaCollection.ref.where('user', '==',
email).get();
      query.then(snapshot => {
        snapshot.forEach( doc => {
          let x = { 'user': doc.id , ...doc.data()};
```

```
          lreqM.push(x);
        });
        n += lreqM.length;
      }
        );

        query2 = this.accidenteCollection.ref.where('user', '==',
email).get();
        query2.then(snapshot => {
          snapshot.forEach( doc => {
            let x = { 'user': doc.id , ...doc.data()};
            lreqA.push(x);
          });
          n += lreqA.length;
          resolve(n);
        }
          );
    });

  }

  /**
   * Metodo que encripta la contraseña del usuario
   * @param keys String con el formato en el que se va encriptar
   * @param value String valor de la contraseña
   * @returns String con el valor encriptado de la contraseña
   */
  set(keys, value){
    var key = CryptoJS.enc.Utf8.parse(keys);
    var iv = CryptoJS.enc.Utf8.parse(keys);
    var encrypted =
CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(value.toString()), key,
    {
        keySize: 128 / 8,
        iv: iv,
        mode: CryptoJS.mode.CBC,
        padding: CryptoJS.pad.Pkcs7
    });

    return encrypted.toString();
  }

  /**
   * Metodo que desencripta la contraseña del usuario
   * @param keys String con el formato en el que se va desencriptar
   * @param value String valor de la contraseña
   * @returns String con el valor desencriptado de la contraseña
   */
  get(keys, value) {
```

```
    var key = CryptoJS.enc.Utf8.parse(keys);
    var iv = CryptoJS.enc.Utf8.parse(keys);
    var decrypted = CryptoJS.AES.decrypt(value, key, {
        keySize: 128 / 8,
        iv: iv,
        mode: CryptoJS.mode.CBC,
        padding: CryptoJS.pad.Pkcs7
    });

    return decrypted.toString(CryptoJS.enc.Utf8);
  }

  myLocation(latlong){
    this.location = latlong
  }

////////////////////////////////////////OBTENCION, ADICCION,
MODIFICACION Y ELIMINACION DE DATOS EN
FIREBASE////////////////////////////////////

  /**
   * Recibe un objeto y lo guarda como un documento nuevo en la colección
'accidente'
   * Devuelve un Promise
   * @param datos documento a insertar en firebase
   * @return AngularFireStoreCollection Devuelve un promise
   */
  anadirA(datos) {
    return this.accidenteCollection.add(datos).then(e => {

      console.log('Todo ok');
    }).catch(err => {

      console.log(err);
    });
  }

  /**
   * Carga de accidentes en caso de no estar presente la variable reload,
se añaden los siguientes 15 al final de la lista
   * @param reload evento que acciona la carga de mas elementos a la
lista
    */
  getAccident(reload?): Promise<iAccidente[]> {
    if (reload) {
      this.lastlastAccidentLoaded = null;
      this.scrollAccidentEnabled = true;
    }
    this.lastAccidentLoaded = this.lastlastAccidentLoaded;
```

```
    return new Promise((resolve) => {
      let lreq: iAccidente[] = [];
      let query;
      if (this.lastAccidentLoaded == null) {
        /* Obtengo los primeros 15 accidentes ordenados por descripcion.
Para ordenar es necesario
        activar un índice en firebase. Si no se crea dará un error por
consola indicando los pasos
        necesarios para crearlo */
        query = this.accidenteCollection.ref.orderBy('hora',
'asc').limit(15).get();

      } else {
        /* Cargamos 15 a partir del último cargado */
        query = this.accidenteCollection.ref.orderBy('hora',
'asc').startAfter(this.lastAccidentLoaded).limit(15).get();
      }
      query.then((d) => {
        d.forEach((u) => {
          let x = { 'hora': u.id, ...u.data() };
          /*Unicamente ase van a añadir a la vista las alertas que lleven
menos de una hora, por defecto estas no se mostraran y si
          el aviso sigue estando en el lugar de los hechos basta con
volvera a crear la alerta*/
          let horaLocal = new Date().valueOf();

          if (x.hora + 3600000 <= horaLocal) {

            this.accidenteCollection.doc(u.id).delete().then(e => {
              console.log('Document successfully deleted!');
          }).catch(function(error) {
              console.error('Error removing document: ', error);
          });
//falta mejorar con la ubicacion del usuario
          } else if(this.getDistance() >=
this.betwen2Points([this.location.lat,this.location.long],[x.latitud,x.lo
ngitud])){
            lreq.push(x);

          }

        });
        this.lastlastAccidentLoaded = d.docs[d.docs.length - 1];
        if (d.docs.length < 10) {
          this.scrollAccidentEnabled = false;
        }
        resolve(lreq);

      });
```

```
    });
  }

  /* Carga de accidentes en el mapa (Crea los marcadores) unicamente se
mostraran hasta 1000 avisos*/
  getMarkAccident(): Promise<iAccidente[]>{

    return new Promise((resolve, reject) => {
      let lreq: iAccidente[] = [];
      let query;

      query = this.accidenteCollection.ref.orderBy('hora',
'asc').limit(1000).get();
      query.then((d) => {
        d.forEach((u) => {
          let x = { 'hora': u.id, ...u.data() };
          /*Unicamente ase van a añadir a la vista las alertas que lleven
menos de una hora, por defecto estas no se mostraran y si
          el aviso sigue estando en el lugar de los hechos basta con
volvera a crear la alerta*/

          let horaLocal = new Date().valueOf();

          if (x.hora + 3600000 <= horaLocal) {

            this.accidenteCollection.doc(u.id).delete().then(e => {
              console.log('Document successfully deleted!');
          }).catch(function(error) {
              console.error('Error removing document: ', error);
          });
          } else {

            lreq.push(x);

          }
        });
        resolve(lreq);

      });
    });

  }

/**
 *  Recibe un objeto y lo guarda como un documento nuevo en la colección
'meteorologia'
  Devuelve un Promise
  @param datos documento a insertar en firebase
  @return Devuelve un promise
```

```
 */
  anadirM(datos) {
    return this.meteorologiaCollection.add(datos).then(e =>{

      console.log('Todo ok');
    }).catch(err =>{

      console.log(err);
    });
  }

  /**
   * Carga de accidentes en caso de no estar presente la variable reload,
se añaden los siguientes 15 al final de la lista
   *  @param reload evento que acciona la carga de mas elementos a la
lista
   */
   getMeteorology(reload?): Promise<iMeteorology[]> {
    if (reload) {
      this.lastlastMeteorologyLoaded = null;
      this.scrollMeteorologyEnabled = true;
    }
    var distance = this.getDistance();
    this.lastMeteorologyLoaded = this.lastlastMeteorologyLoaded;
    return new Promise((resolve, reject) => {
      let lreq: iMeteorology[] = [];
      let query;
      if (this.lastMeteorologyLoaded == null) {
        /* Obtengo los primeros 15 eventos de meteorologia ordenados por
descripcion. Para ordenar es necesario
        activar un índice en firebase. Si no se crea dará un error por
consola indicando los pasos
        necesarios para crearlo */
        query = this.meteorologiaCollection.ref.orderBy("hora",
"asc").limit(15).get();

      } else {
        /* Cargamos 15 a partir del último cargado */
        query = this.meteorologiaCollection.ref.orderBy("hora",
"asc").startAfter(this.lastMeteorologyLoaded).limit(15).get();
      }
      query.then((d) => {
        d.forEach((u) => {
          let x = { "hora": u.id, ...u.data() };
          let horaLocal = new Date().valueOf();

          /*Unicamente ase van a añadir a la vista las alertas que lleven
menos de una hora, por defecto estas no se mostraran y si
```

```
          el aviso sigue estando en el lugar de los hechos basta con
volvera a crear la alerta*/

          if(x.hora+3600000<= horaLocal){
              //Se elimina ese aviso
              this.meteorologiaCollection.doc(u.id).delete().then(e => {
                console.log("Document successfully deleted!");
            }).catch(function(error) {
                console.error("Error removing document: ", error);
            });

            //falta mejorar con la ubicacion del usuario
          } else if(this.getDistance() >=
this.betwen2Points([this.location.lat,this.location.long],[x.latitud,x.lo
ngitud])){

            lreq.push(x);
          }
        });
        this.lastlastMeteorologyLoaded = d.docs[d.docs.length - 1];
          if (d.docs.length < 10) {
            this.scrollMeteorologyEnabled = false;
          }
        resolve(lreq);

      });
    });
  }


/**
 *  Carga las marcas de meteorologia en el mapa. Como maximo se van a
maracar 1000 marcas
 * */
  getMarkMeteorology(): Promise<iMeteorology[]>{

    return new Promise((resolve, reject) => {
      let lreq: iMeteorology[] = [];
      let query;

      query = this.meteorologiaCollection.ref.orderBy("hora",
"asc").limit(1000).get();

      query.then((d) => {
        d.forEach((u) => {
          let x = { "hora": u.id, ...u.data() };
          let horaLocal = new Date().valueOf();
/*Unicamente ase van a añadir a la vista las alertas que lleven menos de
una hora, por defecto estas no se mostraran y si
```

```
          el aviso sigue estando en el lugar de los hechos basta con
volvera a crear la alerta*/

          if (x.hora + 3600000 <= horaLocal) {
              this.meteorologiaCollection.doc(u.id).delete().then(e => {
              console.log("Document successfully deleted!");
              }).catch(function(error) {
              console.error("Error removing document: ", error);
              });
          } else {

              lreq.push(x);

          }
        });
        resolve(lreq);

      });
    });

  }
  ////////////////////////////////////////////////////////////FUNCIONES
VARIAS//////////////////////////////////////////////
  /**
 * Calcula la distancia entre 2 puntos
 * @param origin Array Number con las coordenadas donde esta ubicado el
usuario
 * @param destination Array Number con las coordenadas de la alerta
 * @returns distancia en Km entre los dos puntos
 */
betwen2Points(origin, destination) {
  // return distance in meters
  var lon1 = this.toRadian(origin[1]),
      lat1 = this.toRadian(origin[0]),
      lon2 = this.toRadian(destination[1]),
      lat2 = this.toRadian(destination[0]);

  var deltaLat = lat2 - lat1;
  var deltaLon = lon2 - lon1;

  var a = Math.pow(Math.sin(deltaLat/2), 2) + Math.cos(lat1) *
Math.cos(lat2) * Math.pow(Math.sin(deltaLon/2), 2);
  var c = 2 * Math.asin(Math.sqrt(a));
  var EARTH_RADIUS = 6371;
  var betwen2Points: Number;
  betwen2Points = c * EARTH_RADIUS;
  return betwen2Points;
}
/**
```

```
* Calcula los radianes de cada coordenada recibida en grados
* @param degree grados a convertir
*/
toRadian(degree) {
  return degree*Math.PI/180;
}

getDistance(){
  this.nativeStorage.getItem('distance').then(distance => {

    this.distance= distance.km ;

  }).catch(e => {
    this.distance = 1000 ;
  });
  return this.distance;
}

/**
   * Habilita el scroll para la pestaña meteorologia
   * @returns devuelve si se ha habilitado el scroll
   *  */
  misInfinityScrollEnabled() {
    return this.scrollMeteorologyEnabled;
  }


/**
   * Habilita el scroll para la pestaña accidentes
   * @returns devuelve si se ha habilitado el scroll
   */
  isInfinityScrollEnabled() {
    return this.scrollAccidentEnabled;
  }

}
```

- NAVEGACIONSERVICE

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
/**
 * Clase utilizada para la navegacion del modal hacia la marca en el mapa
 */
export class NavegacionService {
```

```
 private cargaM: boolean
 private latitud: any;
 private longitud: any;
 private addM: boolean;
  constructor() {

  this.cargaM=false;

   }

/**
 * recoge la ubicacion del modal
 * @param lat latitud de la ubicación del modal
 * @param lng longitud de la ubicación del modal
 */
recibeUbicacion(lat:any, lng:any){

  this.latitud= lat;
  this.longitud=lng;
}
/**
 * Devuelve la latitud
 * @return this.latitud
 */
getLatitud(){

  return this.latitud;
}

/**
 * Devuelve la longitud
 * @returns this.longitud
 */
getLongitud(){

  return this.longitud;
}

/**
 *  Devuelve cargar mapa
 * @returns this.cargaM
 **/
getCargarMapa(){

  return this.cargaM;
}

/**
 * Establece el valor de cargar mapa a false
```

```
 */
setCargaMapa(){

  this.cargaM=false;
}

/**
 * Devuelve añadir mapa
 * @returns this.addM
 */
getAddM(){
  return this.addM;
}

/*
Establece el valor de añadir mapa a false
*/
setAddM(){
  this.addM=false;
}


/**
 * recibe true en el caso de que se haya la navegacion hacia el mapa
desde el modal, por defecto esta a falso
 * @param verdad recibe true
   */
cargaMapa(verdad?: boolean){

this.cargaM=verdad;

}

/**
 * recibe true en el caso de que se haya la navegacion hacia el mapa
desde el boton add, por defecto esta a falso
 * @param verdad recibe true
 */
addTab1Mark(verdad?: boolean){
  this.addM=verdad;
}


}
```

- NETWORKSERVICE

```
import { Diagnostic } from '@ionic-native/diagnostic/ngx';
```

```
import { Injectable } from '@angular/core';
import { Network } from '@ionic-native/network/ngx';
import { AlertController, Events } from '@ionic/angular';


/*
Enumeracion con dos valores
online para cuando la conexion a internet esta activa
offline para cuando la conexion a internet esta desactivada
*/
export enum ConnectionStatusEnum {
    Online,
    Offline
}



/*Proveedor de conexion a internet
con esta clase vamos a comprobar si tenemos internet o no
al iniciar la app o al realizar conexiones con firebase
*/
@Injectable()
export class NetworkService {

  public previousStatus;
  public colorN;



  constructor(public alertCtrl: AlertController,
              public network: Network,
              public eventCtrl: Events,
              public diagnostic: Diagnostic) {

    this.previousStatus = ConnectionStatusEnum.Online;

  }

  /*
  Metodo que nos va a comprobar la conexion a internet
  Cambia el valor de la enumeracion en funcion se active o se
  desactive el internet
  */
    public initializeNetworkEvents(): void {
        this.network.onDisconnect().subscribe(() => {
            if (this.previousStatus === ConnectionStatusEnum.Online) {
                this.eventCtrl.publish('network:offline');
            }
            this.previousStatus = ConnectionStatusEnum.Offline;
```

```
        });
        this.network.onConnect().subscribe(() => {
            if (this.previousStatus === ConnectionStatusEnum.Offline) {
                this.eventCtrl.publish('network:online');
            }
            this.previousStatus = ConnectionStatusEnum.Online;
        });
    }

/**
 * Metodo que comprueba el gps y el estado que tiene
 */
    publicShowGPSEvent(): void{
        this.diagnostic.getLocationMode().then(locationMode =>{

            switch(locationMode){
                case this.diagnostic.locationMode.HIGH_ACCURACY:
                    this.eventCtrl.publish('High accuracy');
                    break;
                case this.diagnostic.locationMode.BATTERY_SAVING:
                    this.eventCtrl.publish('Battery saving');
                    break;
                case this.diagnostic.locationMode.DEVICE_ONLY:
                    this.eventCtrl.publish('Device only');
                    break;
                case this.diagnostic.locationMode.LOCATION_OFF:
                    this.eventCtrl.publish('Location off');
                    break;
            }

        });
        }

/**
 * Metodo que cambia el color del icono de la señal del gps
 * @param color recibe el color al que se cambia el icono de la señal
 */
        colorSignalGPS(color? ) {

           if(color == 'green'){
               console.log("hola soy el color "+color);
            this.colorN='success';

            }else if( color == 'yellow'){
                console.log("hola soy el color "+color);
                this.colorN='warning';

            }else if(color == 'orange'){
                console.log("hola soy el color "+color);
```

```
                this.colorN='danger';

            }else if(color == 'red'){
                console.log("hola soy el color "+color);
                this.colorN='dark';

            }



        }

}
```

10.11 Tab1

- HTML

```
<ion-header color="primary">
  <ion-toolbar color="primary">
    <ion-title>
      ZeroD
    </ion-title>
    <ion-icon class="help" name="help-circle-outline"
(click)="presentModal()"></ion-icon>
  </ion-toolbar>
  <ion-segment [(ngModel)]="category"
(ionChange)="slides.slideTo(category)" style="background-color:var(--ion-
color-primary)"
  color="light">
  <ion-segment-button layout="icon-start" value="0" style="--padding-
end:0px;--padding-start:0px">
    <ion-icon name="ios-car" style="margin-right:1px;zoom:1"></ion-icon>

  </ion-segment-button>
  <ion-segment-button layout="icon-start" value="1" style="--padding-
end:0px;--padding-start:0px">
    <ion-icon name="ios-umbrella" style="margin-right:1px;zoom:1"></ion-
icon>

  </ion-segment-button>
</ion-segment>
<div id='indicator' class="SwipedTabs-indicatorSegment" [ngStyle]="
  {'width.%': (100/this.tabs.length)}" ></div>

</ion-header>

<ion-content >
```

```
    <ion-slides #slides
(ionSlideTransitionStart)="updateIndicatorPosition();updateCat(slides.get
ActiveIndex())"
    (ionSlideWillChange)="updateIndicatorPosition()"
(ionSlideDidChange)="updateIndicatorPosition()" #SwipedTabsSlider
    (ionSlideDrag)="animateIndicator($event)">
    <ion-slide>

      <ion-img *ngIf="listAvacia" src="assets/OK.png" style="heigh:auto;
width:auto;"></ion-img>
      <div *ngIf="!listAvacia">
            <ion-refresher
(ionRefresh)="this.updateAccident($event,true)">
              <ion-refresher-content pullingIcon="arrow-dropdown"
                refreshingSpinner="null" refreshingText="{{ 'refresher' |
translate }}">

              </ion-refresher-content>
            </ion-refresher>

            <ion-list >
              <!--Se puede mejorar el uso de la funcion de showInfo y por
lo consiguiente todo lo que es el modal, si solo pasamos el item entero y
luego sacamos la información necesaria-->
              <ion-item style="--background: transparent !important;"
class="item ios in-list ion-focusable item-label hydrated" *ngFor="let
item of listadoAccidentes;let i = index"
(click)="showInfo(item.descripcion, item.alert, item.hora, item.ciudad,
item.calle, item.latitud, item.longitud)">

                <ion-avatar slot="start" class="ios hydrated">
                  <img [src]="safeImage(item.imgU)">
                </ion-avatar>
                <ion-label class="sc-ion-label-ios-h sc-ion-label-ios-s
ios hydrated">
                  <h2>{{item.descripcion}}</h2>
                  <p>
                  <ion-chip outline="" color="primary" class="ion-color
ion-color-primary md chip-outline ion-activatable hydrated">
                    <ion-icon name="pin" role="img" class="md hydrated
iconbadge" aria-label="pin"></ion-icon>
                    <ion-label class="sc-ion-label-md-h sc-ion-label-md-s
md hydrated">{{item.ciudad}} {{item.calle}}</ion-label>
                  </ion-chip>
                </p>
                </ion-label>
              </ion-item>
            </ion-list>
```

```
            <ion-infinite-scroll #infiniteScroll threshold="10px"
(ionInfinite)="updateAccident($event,false)">
              <ion-infinite-scroll-content loadingSpinner="null"
loadingText="{{ 'refresher' | translate }}">

              </ion-infinite-scroll-content>
            </ion-infinite-scroll>
          </div>


    </ion-slide>
    <ion-slide>
        <ion-img *ngIf="listMvacia" src="assets/OK.png"
style="heigh:auto; width:auto;"></ion-img>

         <div *ngIf="!listMvacia">
        <ion-refresher
(ionRefresh)="this.updateMeteorology($event,true)">
          <ion-refresher-content pullingIcon="arrow-dropdown"
            refreshingSpinner="null" refreshingText="{{ 'refresher' |
translate }}">

          </ion-refresher-content>
        </ion-refresher>


        <ion-list *ngIf="!listMvacia">
           <ion-item style="--background: transparent !important;"
class="item ios in-list ion-focusable item-label hydrated" *ngFor="let
item of listadoMeteorologia;let i = index"
(click)="showInfo(item.descripcion, item.alert, item.hora, item.ciudad,
item.calle)">

                <ion-avatar slot="start" class="ios hydrated">
                  <img [src]="safeImage(item.imgU)">
                </ion-avatar>
                <ion-label class="sc-ion-label-ios-h sc-ion-label-ios-s
ios hydrated">
                  <h2>{{item.descripcion}}</h2>
                  <p>
                  <ion-chip outline="" color="primary" class="ion-color
ion-color-primary md chip-outline ion-activatable hydrated">
                    <ion-icon name="pin" role="img" class="md hydrated
iconbadge" aria-label="pin"></ion-icon>
                    <ion-label class="sc-ion-label-md-h sc-ion-label-md-s
md hydrated">{{item.ciudad}} {{item.calle}}</ion-label>
                  </ion-chip>
```

```
                </p>
                </ion-label>
              </ion-item>
        </ion-list>

        <ion-infinite-scroll #infiniteScroll threshold="10px"
(ionInfinite)="updateMeteorology($event,false)">
          <ion-infinite-scroll-content loadingSpinner="null"
loadingText="{{ 'refresher' | translate }}">

          </ion-infinite-scroll-content>
        </ion-infinite-scroll>

      </div>
    </ion-slide>
  </ion-slides>

  <ion-fab vertical="bottom" horizontal="end" slot="fixed" >

    <ion-fab-button (click)="anadeMarca()">
      <ion-icon name="add" style="color: var(--ion-color-success-
contrast)" ></ion-icon>

    </ion-fab-button>


  </ion-fab>

</ion-content>
```

- SCSS

```
/* la "rayita" que indica la navegación entre slides */
.SwipedTabs-indicatorSegment{
  -webkit-transition: 0s all;
  -moz-transition:  0s all;
  -o-transition:  0s all;
  transition:  0s all;
  transform-origin: top 0 left 0;
  height: 2px;
  position: relative;
  top: -2px;
  background-color: var(--ion-color-light) !important;
}

/* para que el slide ocupe toda la pantalla */
ion-slide.swiper-slide {
```

```
  display:block;
}
/*Para que el slide pueda tener scroll en una lista*/
ion-slides{
  height: initial;
}

::-webkit-scrollbar,
*::-webkit-scrollbar {
display: none;
}

ion-img{
  display: initial;
}

.item .sc-ion-label-md-h {

  overflow: visible;

}
.sc-ion-label-md-s h2{
  overflow: hidden;
  text-overflow: ellipsis;
}



ion-segment-button{

  --background: var(--ion-color-primary);

}

ion-icon{

  float: left;
  height: 30px;
  width: 30px;
  margin-right: 3%;

}

.iconbadge{

  float: auto;
  height: 15px;
  width: 15px;
  margin-right: 10px;
```

```
}

.iconlist{

  height: 35px;
  width: 35px;

}


.list-md{

  background: var( --ion-color-light-shade)!important;
}



ion-title{
  float: left;
  margin-right: 10px;
  margin-top: 5px;
  color: var(--ion-color-light-contrast);
}


.item .item-native{

  background-color: transparent !important;
}

.help{
  float: right;
  color: var(--ion-color-light-contrast);
}

.swiper-slide img {
  width: max-content;
  max-width: 100%;
  height: max-content;
  max-height: 100%;
}
```

- TS

```
import { HelpComponent } from './../customComponent/help/help.component';
import { NetworkService } from './../servicios/network.service';
```

```
import { CustomModalModule } from './../custom-modal/custom-
modal.module';
import { iAccidente } from './../model/iAccident';
import { CloudserviceService } from
'./../servicios/cloudservice.service';
import { Component, ViewChild } from '@angular/core';
import { IonInfiniteScroll, IonSlides, ModalController} from
'@ionic/angular';
import { Router } from '@angular/router';
import { iMeteorology } from '../model/iMeteorology';
import { CustomLoading } from '../custom-modal/custom-loading';
import { WeatherService } from '../servicios/weather.service';
import { ViewCardComponent } from '../customComponent/view-card/view-
card.component';
import { ScrollHideConfig } from '../directives/scroll-hide.directive';
import { NavegacionService } from '../servicios/navegacion.service';
import { Geolocation } from '@ionic-native/geolocation/ngx';
import * as $ from 'jquery';
import { DomSanitizer } from '@angular/platform-browser';


@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss']
})
export class Tab1Page {
  @ViewChild('infiniteScroll') ionInfiniteScroll: IonInfiniteScroll;
  @ViewChild('SwipedTabsSlider') SwipedTabsSlider: IonSlides;
  SwipedTabsIndicator: any = null;
  tabs = ['selectTab(0)', 'selectTab(1)'];
  ntabs = 2;
  public category: any = '0';
  listadoAccidentes: iAccidente[] = [];
  listadoMeteorologia: iMeteorology[] = [];
  footerScrollConfig: ScrollHideConfig = { cssProperty: 'margin-bottom',
maxValue: undefined };
  headerScrollConfig: ScrollHideConfig = { cssProperty: 'margin-top',
maxValue: 44 };
  listMvacia;
  listAvacia;

slidesOpts = {
  grabCursor: true,
  cubeEffect: {
    shadow: true,
    slideShadows: true,
    shadowOffset: 20,
    shadowScale: 0.94,
```

```
  },
  on: {
    beforeInit: function() {
      const swiper = this;

swiper.classNames.push(`${swiper.params.containerModifierClass}cube`);

swiper.classNames.push(`${swiper.params.containerModifierClass}3d`);

      const overwriteParams = {
        slidesPerView: 1,
        slidesPerColumn: 1,
        slidesPerGroup: 1,
        watchSlidesProgress: true,
        resistanceRatio: 0,
        spaceBetween: 0,
        centeredSlides: false,
        virtualTranslate: true,
      };

      this.params = Object.assign(this.params, overwriteParams);
      this.originalParams = Object.assign(this.originalParams,
overwriteParams);
    },
    setTranslate: function() {
      const swiper = this;
      const {
        $el, $wrapperEl, slides, width: swiperWidth, height:
swiperHeight, rtlTranslate: rtl, size: swiperSize,
      } = swiper;
      const params = swiper.params.cubeEffect;
      const isHorizontal = swiper.isHorizontal();
      const isVirtual = swiper.virtual && swiper.params.virtual.enabled;
      let wrapperRotate = 0;
      let $cubeShadowEl;
      if (params.shadow) {
        if (isHorizontal) {
          $cubeShadowEl = $wrapperEl.find('.swiper-cube-shadow');
          if ($cubeShadowEl.length === 0) {
            $cubeShadowEl = swiper.$('<div class="swiper-cube-
shadow"></div>');
            $wrapperEl.append($cubeShadowEl);
          }
          $cubeShadowEl.css({ height: `${swiperWidth}px` });
        } else {
          $cubeShadowEl = $el.find('.swiper-cube-shadow');
          if ($cubeShadowEl.length === 0) {
            $cubeShadowEl = $('<div class="swiper-cube-shadow"></div>');
            $el.append($cubeShadowEl);
```

```
          }
        }
      }

      for (let i = 0; i < slides.length; i += 1) {
        const $slideEl = slides.eq(i);
        let slideIndex = i;
        if (isVirtual) {
          slideIndex = parseInt($slideEl.attr('data-swiper-slide-index'),
10);
        }
        let slideAngle = slideIndex * 90;
        let round = Math.floor(slideAngle / 360);
        if (rtl) {
          slideAngle = -slideAngle;
          round = Math.floor(-slideAngle / 360);
        }
        const progress = Math.max(Math.min($slideEl[0].progress, 1), -1);
        let tx = 0;
        let ty = 0;
        let tz = 0;
        if (slideIndex % 4 === 0) {
          tx = -round * 4 * swiperSize;
          tz = 0;
        } else if ((slideIndex - 1) % 4 === 0) {
          tx = 0;
          tz = -round * 4 * swiperSize;
        } else if ((slideIndex - 2) % 4 === 0) {
          tx = swiperSize + (round * 4 * swiperSize);
          tz = swiperSize;
        } else if ((slideIndex - 3) % 4 === 0) {
          tx = -swiperSize;
          tz = (3 * swiperSize) + (swiperSize * 4 * round);
        }
        if (rtl) {
          tx = -tx;
        }

         if (!isHorizontal) {
          ty = tx;
          tx = 0;
        }

// tslint:disable-next-line: max-line-length
         const transform$$1 = `rotateX(${isHorizontal ? 0 : -
slideAngle}deg) rotateY(${isHorizontal ? slideAngle : 0}deg)
translate3d(${tx}px, ${ty}px, ${tz}px)`;
        if (progress <= 1 && progress > -1) {
          wrapperRotate = (slideIndex * 90) + (progress * 90);
```

```
          if (rtl) wrapperRotate = (-slideIndex * 90) - (progress * 90);
        }
        $slideEl.transform(transform$$1);
        if (params.slideShadows) {
          // Set shadows
          let shadowBefore = isHorizontal ? $slideEl.find('.swiper-slide-
shadow-left') : $slideEl.find('.swiper-slide-shadow-top');
          let shadowAfter = isHorizontal ? $slideEl.find('.swiper-slide-
shadow-right') : $slideEl.find('.swiper-slide-shadow-bottom');
          if (shadowBefore.length === 0) {
            shadowBefore = swiper.$(`<div class="swiper-slide-shadow-
${isHorizontal ? 'left' : 'top'}"></div>`);
            $slideEl.append(shadowBefore);
          }
          if (shadowAfter.length === 0) {
            shadowAfter = swiper.$(`<div class="swiper-slide-shadow-
${isHorizontal ? 'right' : 'bottom'}"></div>`);
            $slideEl.append(shadowAfter);
          }
          if (shadowBefore.length) shadowBefore[0].style.opacity =
Math.max(-progress, 0);
          if (shadowAfter.length) shadowAfter[0].style.opacity =
Math.max(progress, 0);
        }
      }
      $wrapperEl.css({
        '-webkit-transform-origin': `50% 50% -${swiperSize / 2}px`,
        '-moz-transform-origin': `50% 50% -${swiperSize / 2}px`,
        '-ms-transform-origin': `50% 50% -${swiperSize / 2}px`,
        'transform-origin': `50% 50% -${swiperSize / 2}px`,
      });

       if (params.shadow) {
        if (isHorizontal) {
          $cubeShadowEl.transform(`translate3d(0px, ${(swiperWidth / 2) +
params.shadowOffset}px, ${-swiperWidth / 2}px) rotateX(90deg)
rotateZ(0deg) scale(${params.shadowScale})`);
        } else {
          const shadowAngle = Math.abs(wrapperRotate) -
(Math.floor(Math.abs(wrapperRotate) / 90) * 90);
          const multiplier = 1.5 - (
            (Math.sin((shadowAngle * 2 * Math.PI) / 360) / 2)
            + (Math.cos((shadowAngle * 2 * Math.PI) / 360) / 2)
          );
          const scale1 = params.shadowScale;
          const scale2 = params.shadowScale / multiplier;
          const offset$$1 = params.shadowOffset;
```

```
          $cubeShadowEl.transform(`scale3d(${scale1}, 1, ${scale2})
translate3d(0px, ${(swiperHeight / 2) + offset$$1}px, ${-swiperHeight / 2
/ scale2}px) rotateX(-90deg)`);
        }
      }

      const zFactor = (swiper.browser.isSafari ||
swiper.browser.isUiWebView) ? (-swiperSize / 2) : 0;
      $wrapperEl
        .transform(`translate3d(0px,0,${zFactor}px)
rotateX(${swiper.isHorizontal() ? 0 : wrapperRotate}deg)
rotateY(${swiper.isHorizontal() ? -wrapperRotate : 0}deg)`);
    },
    setTransition: function(duration) {
      const swiper = this;
      const { $el, slides } = swiper;
      slides
        .transition(duration)
        .find('.swiper-slide-shadow-top, .swiper-slide-shadow-
right, .swiper-slide-shadow-bottom, .swiper-slide-shadow-left')
        .transition(duration);
      if (swiper.params.cubeEffect.shadow && !swiper.isHorizontal()) {
        $el.find('.swiper-cube-shadow').transition(duration);
      }
    },
  }
}

/**
 * @param cloudS objeto del servicion del almacenamiento
 * @param loading objeto de la clase loading, nos permite mostrar un
cargando
 * @param aemet objeto del servicio que nos da la informacion del tiempo
 * @param cmm objeto de la calse custom modal, nos permite abrir el modal
 * @param netwoekS objeto que comprueba el funcionamiento de internet y
gps
 * @param navegacion objeto del servicio que almacena variables
temporales para la navegacion entre tab
 * @param router objeto que nos permite hacer navegacion hacia otra tab
 * @param modalController objeto del controlador del modal
 */
  constructor(
    private cloudS: CloudserviceService,
    private loading: CustomLoading,
    private aemet: WeatherService,
    private cmm: CustomModalModule,
    private netwoekS: NetworkService,
    private navegacion: NavegacionService,
    private router: Router,
```

```
    public modalController: ModalController,
    private geolocation: Geolocation,
    private sanitizer: DomSanitizer,

  ) {}

  /**
   * Metodo que comprueba la posicion del slider y la conexion a internet
   * cada vez que entra a la tab1
   */
  ionViewWillEnter() {
    // Comprobamos la posicion del slider
    this.SwipedTabsIndicator = document.getElementById('indicator');
    this.SwipedTabsSlider.length().then(l => {
      this.ntabs = l;
    });
    // Comprobamos la conexion a internet
    if (this.netwoekS.previousStatus === 1) {
    } else if (this.netwoekS.previousStatus === 0) {
      this.updateAllAccident();
      this.updateAllMeteorology();
      // obtiene los datos del weather service (Falta implementacion)
      //this.aemet.getRemoteData();
      //
    this.geolocation.getCurrentPosition().then(location =>{
      let geoLocation ={
        lat: location.coords.latitude,
        long: location.coords.longitude
      }
      this.cloudS.myLocation(geoLocation);
    });
    }
  }

  /**
   * Se encarga de comprobar si las listas estan vacias
   * cuando es asi muestra la imagen de que todo esta ok
   */
  checkListA() {
    if (this.listadoAccidentes.length === 0) {
      this.listAvacia = true;
    } else {
      this.listAvacia = false;
    }
  }

/**
   * Se encarga de comprobar si las listas estan vacias
   * cuando es asi muestra la imagen de que todo esta ok
```

```
   */
  checkListM() {
    if (this.listadoMeteorologia.length === 0) {
      this.listMvacia = true;
    } else {
      this.listMvacia = false;
    }
  }


  /**
   * Metodo de carga de los accidentes, se activa cuando se entra en la
tab1
   * Muestra un loading que se quita cuando la carga desde firebase esta
completa
   * @param event evento que desencadena el metodo
   */
  updateAllAccident(event?) {
    // Cargamos de la bd
    this.loading.show('');
      this.cloudS.getAccident(true).then(d => {
        this.listadoAccidentes = d;

        this.checkListA();
        this.updateAccident(event,true)

        if (event) {
          event.target.complete();
        }
        this.loading.hide();
      });


  }

/**
 * Se activa cuando accionamos el refresher de la lista.
 * Carga los accidentes nuevos que se han añadido
 * @param event evento que desencadena el metodo
 * @param reload evento que activa la carga de la bd
 */
  updateAccident(event?, reload?) {
    this.cloudS.getAccident(reload).then(d => {
      if (reload) {
        this.listadoAccidentes = d;
        this.checkListA();
      } else {
        d.forEach((u) => {
          this.listadoAccidentes.push(u);
```

```
        });
      }
      if (event) {
        event.target.complete();
      }
      this.ionInfiniteScroll.disabled
= !this.cloudS.isInfinityScrollEnabled();
    });
  }

   /**
   * Metodo de carga de la meteorologia, se activa cuando se entra en la
tab1
   * Muestra un loading que se quita cuando la carga desde firebase esta
completa
   * @param event evento que desencadena el metodo
   */
  updateAllMeteorology(event?) {
       this.cloudS.getMeteorology(true).then(d => {
         this.listadoMeteorologia = d;

         this.checkListM();
         this.updateMeteorology(event,true)
         if (event) {
           event.target.complete();
         }
       });
   }

  /**
 * Se activa cuando accionamos el refresher de la lista.
 * Carga la meteorologia nueva que se han añadido
 * @param event evento que desencadena el metodo
 * @param reload evento que activa la carga de la bd
 */
   updateMeteorology(event?, reload?) {
     if (!event) {

     }
     this.cloudS.getMeteorology(reload).then(d => {
       if (reload) {
         this.listadoMeteorologia = d;
         this.checkListM();
       } else {
         d.forEach((u) => {
           this.listadoMeteorologia.push(u);
         });
       }
```

```
      if (!event) {

      }
      if (event) {
        event.target.complete();
      }
      this.ionInfiniteScroll.disabled
= !this.cloudS.isInfinityScrollEnabled();
    });
  }


  /* Actualiza la categoría que esté en ese momento activa*/
  updateCat(cat: Promise<any>) {
    cat.then(dat => {
      this.category = dat;
      this.category = +this.category; // to int;
    });
  }

  /* El método que permite actualizar el indicado cuando se cambia de
slide*/
  updateIndicatorPosition() {
    this.SwipedTabsSlider.getActiveIndex().then(i => {
      if (this.ntabs > i) {  // this condition is to avoid passing to
incorrect index
        this.SwipedTabsIndicator.style.webkitTransform = 'translate3d(' +
(i * 100) + '%,0,0)';
      }
    });
  }
  /* El método que anima la "rayita" mientras nos estamos deslizando por
el slide*/
  animateIndicator(e) {
    console.log(e.target.swiper.progress);
    if (this.SwipedTabsIndicator) {
      this.SwipedTabsIndicator.style.webkitTransform = 'translate3d(' +
        ((e.target.swiper.progress * (this.ntabs - 1)) * 100) + '%,0,0)';
    }
  }



  /**
   * Acciona el modal que muestra la informacion completa de la alerta
   * Recibe por parametros todos los campos de nuestra alerta
   * @param descripcion descripcion de la alerta
   * @param tipo tipo de alerta recibida
   * @param hora hora en la que se creo el evento
```

```
   * @param ciudad de la ubicación del evento
   * @param calle de la ubicación del evento
   * Estas dos variables representan la ubicacion
   */

  showInfo(descripcion: any, tipo: any, hora: any, ciudad: any, calle:
any, latitud:any, longitud: any) {

    this.cmm.showInfo(ViewCardComponent, descripcion, tipo, hora, ciudad,
calle, latitud,longitud,this);
  }

  /**
   * Metodo que inicia la navegacion hasta el mapa
   * añadiendo una marca
   */
  anadeMarca() {
    this.navegacion.addTab1Mark(true);
    this.router.navigate(['/tabs/tab2']);
  }

  /**
   * Metodo que abre el modal de ayuda
   */
   presentModal() {
    this.cmm.showHelp(HelpComponent, this);

  }

  /**
   * Metodo que convierte a una img segura nuestra cadena en base64
   * @param img imagen a recibir
   * @return this.sanitizer.bypassSecurityTrustUrl(img)
   */
  safeImage(img){
    return this.sanitizer.bypassSecurityTrustUrl(img)
  }
}
```

10.12 Tab2

- HTML

```
<ion-content>


  <div id="map" style="width:100%; height:100%;"></div>
  <ion-fab class="signal" vertical="top" horizontal="start" slot="fixed">
      <!--<ion-icon name="wifi" color={{this.colorB}}></ion-icon>-->
  </ion-fab>
```

```
  <ion-fab vertical="top" horizontal="end" slot="fixed" >

  <ion-fab-button class="menu" >
      <ion-icon name="md-settings" color="primary"></ion-icon>
  </ion-fab-button>
  <ion-fab-list  side="start">
    <ion-fab-button class="menudesp"><ion-icon name="ios-car"
(click)="chargeAllMarkAccident(ocultaA)" [ngStyle]="{'color':ocultaA  ?
'DARKGRAY' : '#71A1F0' }"></ion-icon></ion-fab-button>
  </ion-fab-list>
  <ion-fab-list class="hide" side="start">
    <ion-fab-button class="menudesp" ><ion-icon name="ios-umbrella"
(click)="chargeAllMarkMeteorology(ocultaM)"
[ngStyle]="{'color':ocultaM  ? 'DARKGRAY' : '#71A1F0' }"></ion-
icon></ion-fab-button>
  </ion-fab-list>
</ion-fab>

  <ion-fab vertical="bottom" horizontal="end" slot="fixed" >
    <ion-fab-button  color="light" [disabled]="this.disablebutton"
(click)="locateme()">
      <ion-icon name="md-locate" color="primary" ></ion-icon>
    </ion-fab-button>
    <ion-fab-button [disabled]="this.disablebutton" (click)="addMark()">
      <ion-icon name="add" style="color: var(--ion-color-success-
contrast)" ></ion-icon>

    </ion-fab-button>


  </ion-fab>
</ion-content>
```

- SCSS

```
ion-fab-button{

    height: 55px;

    width: 55px;

    margin-bottom: 10px;
}

.hide{

    margin-right: 100px;
```

```
}

.menu{

    margin-top: 50px;
    height: 30px;
    width: 30px;
    --background: none;
}

.menudesp{
    margin-top: 40px;
    height: 45px;
    width: 45px;
    --background: var(--ion-color-primary-contrast);

}

ion-fab-list{
    margin-right: 40px;


}



ion-icon{

    height: 25px;
    width: 25px;
}

.signal{
    margin-top:30px;
}
```

- TS

```
import { CustomToast } from './../custom-modal/custom-toast';
import { CloudserviceService } from
'./../servicios/cloudservice.service';
import { AddAlertComponent } from './../customComponent/add-alert/add-
alert.component';
```

```
import { CustomModalModule } from './../custom-modal/custom-
modal.module';
import { Component, ViewChild, ElementRef} from '@angular/core';
import leaflet from 'leaflet';
import { CustomLoading } from '../custom-modal/custom-loading';
import { ModalController, NavController, Events, Platform } from
'@ionic/angular';
import { iAccidente } from '../model/iAccident';
import { iMeteorology } from '../model/iMeteorology';
import { NavegacionService } from '../servicios/navegacion.service';
import { TranslateService } from '@ngx-translate/core';
import { NativeStorage } from '@ionic-native/native-storage/ngx'
import { NetworkService } from '../servicios/network.service';
import { Diagnostic } from '@ionic-native/diagnostic/ngx';
import { AndroidPermissions } from '@ionic-native/android-
permissions/ngx';









@Component({
  selector: 'app-tab2',
  templateUrl: 'tab2.page.html',
  styleUrls: ['tab2.page.scss']
})
export class Tab2Page {
  @ViewChild('map') mapContainer: ElementRef;
  ocultaA: Boolean;
  ocultaM: Boolean;
  map: any;
  latitud: any;
  longitud: any;
  listadoMarcaAccidente: iAccidente[] = [];
  listadoMarcaMeteorology: iMeteorology[] = [];
  markerGroupM = leaflet.featureGroup();
  markerGroupA = leaflet.featureGroup();
  manualGMarker = leaflet.featureGroup();
  manualMarker: any;
  colorB: any;
  disablebutton = false;

//////////////////////////////////////ICONS/////////////////////////
  lluvia = leaflet.icon({
    iconUrl: '../../assets/icon/weather/rainy.png',
    iconSize: [40, 40], // size of the icon
  });
```

```
  nieve = leaflet.icon({
    iconUrl: '../../assets/icon/weather/snow.png',
    iconSize: [40, 40], // size of the icon
  });
  viento = leaflet.icon({
    iconUrl: '../../assets/icon/weather/wind.png',
    iconSize: [40, 40], // size of the icon
  });
  olas = leaflet.icon({
    iconUrl: '../../assets/icon/weather/waves.png',
    iconSize: [30, 30], // size of the icon
  });
  accidente = leaflet.icon({
    iconUrl: '../../assets/icon/traffic/accident.png',
  });
  control = leaflet.icon({
    iconUrl: '../../assets/icon/traffic/control.png',
    iconSize: [30, 30], // size of the icon
  });
  radar = leaflet.icon({
    iconUrl: '../../assets/icon/traffic/radar.png',
    iconSize: [30, 30], // size of the icon
  });
  atasco = leaflet.icon({
    iconUrl: '../../assets/icon/traffic/atasco.png'
  });
  obras = leaflet.icon({
    iconUrl: '../../assets/icon/traffic/obras.png',
    iconSize: [25, 25], // size of the icon
  });
  cloudy = leaflet.icon({
    iconUrl: '../../assets/icon/weather/fog.png',
    iconSize: [40, 40], // size of the icon
  });




/**
 *
 * @param modalController objeto del controlador del modal
 * @param cmm objeto servicio del custom modal
 * @param cloudS objeto servicio de almacenamiento en la nube
 * @param navCtrl objeto servicio de navegacion
 * @param loading objeto servicio del loading
 * @param openM objejo del servicio de navegacion que establece variables
temporales
 * @param nativeStorage objeto del almacenamiento nativo
```

```
 * @param netwoekS objeto del servicio que comprueba la conexion a
internet o gps
 * @param diagnostic objeto del servicio que comprueba el estado de los
elementos nativos
 * @param toast objeto del servicio del custom toast
 * @param translate objeto del servicio de la traducción
 * @param eventCtrl objeto del servio de Events
 * @param androidPermissions objeto del servicio que administra los
permisos de android
 */
  constructor(
    public modalController: ModalController,
    private cmm: CustomModalModule,
    private cloudS: CloudserviceService,
    public navCtrl: NavController,
    private loading: CustomLoading,
    private openM: NavegacionService,
    private nativeStorage: NativeStorage,
    private netwoekS: NetworkService,
    private diagnostic: Diagnostic,
    private toast: CustomToast,
    private translate: TranslateService,
    public eventCtrl: Events,
    private androidPermissions: AndroidPermissions,
    private platform: Platform


  ) {
 /*
 Cambiamos el valor del color del boton
 En funcion de la calidad de gps que tengamos.
 Falta implementación
 */
    this.colorB = this.netwoekS.colorN;
  }


  /*este metodo se acciona antes de a entrar en la pagina
    carga el mapa y actualiza las marcas
    Volvemos a comprobar la conexion a internet
  */
  ionViewWillEnter() {
    this.loading.show('');
  /*Cambiamos el valor del color del icono de señal
    En funcion de la calidad de gps que tengamos.
 */
    this.colorB = this.netwoekS.colorN;
```

```
// Se comprueba la conexión a internet y se hacen las respectivas
operaciones segun tengamos o no activada la conexion
    if (this.netwoekS.previousStatus === 1) {
      this.loadmap();
    } else if (this.netwoekS.previousStatus === 0) {

      this.loadmap();
/*comprueba si existe la variable ocultaA y ocultaM en la memoria del
dispositivo
Si no es asi se crea la variable en la memoria y se inicializa en la
clase
Si ya existe se obtiene el valor y se asigna a la de la clase
Esta variable es la que nos permite ocultar o habilitar las marcas en el
mapa
*/
this.nativeStorage.getItem('ocultaA').then( d => {
      this.ocultaA = d.property;
      console.log(this.ocultaA);
      this.chargeAllMarkAccident(this.ocultaA);
}).catch(e => {

    this.ocultaA = false;
    this.chargeAllMarkAccident(this.ocultaA);
    this.nativeStorage.setItem('ocultaA', {property: false})
    .then(
      () => console.log('Stored item!'),
      error => console.error('Error storing item', error)
    );

});

this.nativeStorage.getItem('ocultaM').then((d) => {
      this.ocultaM = d.property;
      console.log(this.ocultaM);
      this.chargeAllMarkMeteorology(this.ocultaM);
}).catch( e => {

  this.ocultaM = false;
  this.chargeAllMarkMeteorology(this.ocultaM);
  this.nativeStorage.setItem('ocultaM', {property: false})
  .then(
    () => console.log('Stored item!'),
    error => console.error('Error storing item', error)
  );
});

// comprobamos si se ha navegado desde la tab 1
      if (this.openM.getAddM() === true) {
          this.addMark();
```

```
          this.openM.setAddM();
      }
    }
    this.loading.hide();


  }

  /*Este metodo se acciona cuando se va a cerrar la tab
    establecemos el valor de la variable del servicio a false
  */
  ionViewDidLeave() {
    this.openM.setCargaMapa();
    this.map.remove();
    // TODO cuando salga hay que mirar el valor de la variable del
booleano que cambia
    // las marcas para comprobar que valor tiene y cambiarlo al contrario
del cual esta almacenado;
    this.nativeStorage.setItem('ocultaM', {property: !this.ocultaM})
  .then(
    () => console.log('Stored item!'),
    error => console.error('Error storing item', error)
  );
  this.nativeStorage.setItem('ocultaA', {property: !this.ocultaA})
  .then(
    () => console.log('Stored item!'),
    error => console.error('Error storing item', error)
  );

  }


  /**
   * Funcion encargada de cargar el mapa cuando cambiamos a la tab2
   * Comprueba si se ha navegado desde el modal o desde las tabs
   *
   */
  loadmap() {
    // si el mapa es abierto desde el modal de ver mas informacion de la
alerta
    // se establecera la vista encima de la alerta pulsada
    if (this.openM.getCargarMapa() === true) {
      this.map =
leaflet.map('map').fitWorld().setView([this.openM.getLatitud(),
this.openM.getLongitud()], 15);
      // establecemos los valores a true para mostrar las marcas
    } else {
      // Si el mapa se ha abierto desde los tabs se establecera la vista
en general sobre el mapa de españa
```

```
      this.map = leaflet.map('map').fitWorld().setView([40.416665, -
3.705315], 6);
      this.map.on('click', (e)=>{
        // Este evento añade una marca a nuestro mapa cuando hacemos clic
sobre el
        this.addMark(e.latlng.lat, e.latlng.lng);
      });
    }
    // eliminamos el control del zoom
    this.map.removeControl(this.map.zoomControl);

leaflet.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attributions: 'Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a
href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,
Imagery © <a href="http://mapbox.com">Mapbox</a>',
      // Establecemos un maximo y un minimo de zoom
      maxZoom: 20,
      minZoom: 5,

    }).addTo(this.map);

    // se añade todo al mapa

  }

  /**
   * Esta funcion es activada cuado pulsamos el boton de localizar en el
mapa
   * Primero comprueba si tenemos permisos para acceder al gps del
dispositivo
   * Si no es asi no pide permisos
   * Comprueba si tenemos el gps activado o no lanzando un toast
   */
  locateme() {

    if (this.platform.is('android')) {
       // Comprobamos permisos

this.androidPermissions.checkPermission(this.androidPermissions.PERMISSIO
N.ACCESS_COARSE_LOCATION).then(
      result => {
        if (result.hasPermission === true) {

          // Comprobacion del GPS si no esta activado, lanza un toast
          // Si el gps esta activado nos situa en nuestra posicion
          this.diagnostic.isGpsLocationAvailable().then((verdad) =>{
            if(verdad === false){
              this.toast.show(this.translate.instant('NoGPS'));
```

```
            } else {
              this.map.locate({ setView: true, maxZoom:
15 }).on('locationfound', (e) => {}).on('locationerror', (err) => {
                this.toast.showTop(this.translate.instant('LSGPS'));
              });
            }
          }).catch(e => console.error(e))
        } else {
          // Pedimos permisos

this.androidPermissions.requestPermission(this.androidPermissions.PERMISS
ION.ACCESS_COARSE_LOCATION);
        }

      });


    } else {

      this.map.locate({ setView: true, maxZoom: 15 }).on('locationfound',
(e) => {}).on('locationerror', (err) => {
        this.toast.showTop(this.translate.instant('LSGPS'));
      });


    }

  }



  /**
   *
   * Este metodo es el encargado de mostrar en el mapa las ubicaciones de
los avisos por meteorologia
   * Se mostraran o no segun el valor del boton del menu de configuracion
del mapa
   * Recibe un boleano con esta informacion
   * @param hide boolean
   */
  chargeAllMarkMeteorology(hide: Boolean) {
    if (this.markerGroupM != null) {
      this.map.removeLayer(this.markerGroupM);
    }
    if (hide !== false) {

      // se obtienen las marcas de meteorologia
      this.cloudS.getMarkMeteorology().then(d => {
```

```
        this.listadoMarcaMeteorology = d;
        // se crea un grupo de marcas
        this.markerGroupM = leaflet.featureGroup();

        this.listadoMarcaMeteorology.forEach(element => {

          let marker: any;
          let circle: any;
          switch (element.icon) {
            case 'rainy':

           // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker = leaflet.marker([element.latitud, element.longitud],
{ icon: this.lluvia }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;
            case 'snow':
           // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
           marker = leaflet.marker([element.latitud, element.longitud],
{ icon: this.nieve }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;
            case 'swap':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker = leaflet.marker([element.latitud, element.longitud],
{ icon: this.viento }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;
            case 'boat':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker = leaflet.marker([element.latitud, element.longitud],
{ icon: this.olas }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);


              break;
              case 'cloudy':
```

```
              // Se crea la marca y se le asigna a esa marca un pop up
con la descripcion
              marker = leaflet.marker([element.latitud,
element.longitud], { icon: this.cloudy }).on('click', () => {
              this.map.setView([element.latitud, element.longitud], 15);
            }).bindPopup(element.descripcion);

                break;
            default:
              break;
          }
          // se le asigna un radio de precaucion a la marca
          circle = leaflet.circle([element.latitud, element.longitud],
{ radius: 2000 }, { color: 'green', opacity: .5 });
          this.markerGroupM.addLayer(marker);
            this.markerGroupM.addLayer(circle);

        });
        // se le añade todas las marcas al mapa
        this.map.addLayer(this.markerGroupM);


      });
    }

    // cambia el valor del booleano recibido
    this.ocultaM = !this.ocultaM;
    this.nativeStorage.setItem('ocultaM', {property: this.ocultaM} );
  }



  /**
   *
   * Este metodo es el encargado de mostrar en el mapa las ubicaciones de
los avisos por meteorologia
   * Se mostraran o no segun el valor del boton del menu de configuracion
del mapa
   * Recibe un boleano con esta información
   * @param hide boolean
   */
  chargeAllMarkAccident(hide: Boolean) {

    if (this.markerGroupA != null) {
      this.map.removeLayer(this.markerGroupA);
    }

    if (hide !== false) {
```

```
      // se obtienen las marcas de accidente
      this.cloudS.getMarkAccident().then(d => {

        this.listadoMarcaAccidente = d;

        this.markerGroupA = leaflet.featureGroup();


        this.listadoMarcaAccidente.forEach(element => {
          let marker: any;
          switch (element.icon) {
            case 'car':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker =leaflet.marker([element.latitud, element.longitud],
{ icon: this.accidente }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;

            case 'hand':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker =leaflet.marker([element.latitud, element.longitud],
{ icon: this.control }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;

            case 'speedometer':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker =leaflet.marker([element.latitud, element.longitud],
{ icon: this.radar }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);

              break;

            case 'hourglass':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker =leaflet.marker([element.latitud, element.longitud],
{ icon: this.atasco }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);
```

```
              break;
            case 'hammer':
            // Se crea la marca y se le asigna a esa marca un pop up con
la descripcion
          marker =leaflet.marker([element.latitud, element.longitud],
{ icon: this.obras }).on('click', () => {
            this.map.setView([element.latitud, element.longitud], 15);
          }).bindPopup(element.descripcion);


              break;
            default:
              break;
          }

          this.markerGroupA.addLayer(marker);
          // se añade todo al grupo de marcas

        });
        // Se añade al mapa
        this.map.addLayer(this.markerGroupA);

      });

    }

    // Cambiamos el valor del booleano recibido
    this.ocultaA = !this.ocultaA;
    this.nativeStorage.setItem('ocultaA', {property: this.ocultaA} );

  }




  /**
   * Este metodo es el encargado de crear una marca en el lugar señalado
por el usuario o
   * añade una marca al pulsar el botón añadir marca en la ubicación del
usuario.
   * Comprueba al inicio el estado de los permisos para acceder al GPS
   * Comprueba que el gps este activado
   * Recibe la longitud y la latitud de la pulsación en el mapa
   *  @param mapMark1 latitud de la marca
   *  @param mapMark2 longitud de la marca
   */
  addMark(mapMark1?,mapMark2? ) {
    this.nativeStorage.getItem('user').then(user => {
      this.disablebutton = true;
```

```
      if (mapMark1 != null || mapMark2 != null) {

        if (this.manualMarker != null) {
          this.map.removeLayer(this.manualMarker);

        }
        this.manualMarker = leaflet.marker([mapMark1,
mapMark2]).on('click', async () => {
        // se llama al metodo que recibe la marca, la latitud y la
longitud
           this.touchMark(this.manualMarker,mapMark1 ,mapMark2 );
           });
           this.map.addLayer(this.manualMarker);
      } else {

        if (this.platform.is('android')) {


         // Comprueba permisos

this.androidPermissions.checkPermission(this.androidPermissions.PERMISSIO
N.ACCESS_COARSE_LOCATION).then(
        result => {

          if(result.hasPermission == true){
            // Comprobacion del GPS si no esta activado, lanza un toast
            // Si el gps esta activado nos añade la marca en la ubicacion
actual
            this.diagnostic.isGpsLocationAvailable().then(verdad => {
              if (verdad == false) {
                this.toast.show(this.translate.instant('NoGPS'));
              } else {
                  // te localiza a traves del uso del GPS
                this.map.locate({
                  setView: true, maxZoom: 15
                }).on('locationfound', (e) => {
                  // crea una marca en la localizacion que te encuentras
                  let markerGroup = leaflet.featureGroup();
                  let marker = leaflet.marker([e.latitude, e.longitude]);
                  markerGroup.addLayer(marker);
                  this.map.addLayer(markerGroup);
                  // se llama al metodo que recibe la marca, la latitud y
la longitud
                  this.touchMark(marker, e.latitude, e.longitude);
                }).on('locationerror', (err) => {
                  this.toast.showTop(this.translate.instant('LSGPS'))
                });
                  }
```

```
            }).catch(e => {
              console.error(e);
              this.disablebutton = false;
            });
          } else {

            // Pedimos permisos

this.androidPermissions.requestPermission(this.androidPermissions.PERMISS
ION.ACCESS_COARSE_LOCATION).then(result => {

              if ( result.hasPermission === true) {

                this.diagnostic.isGpsLocationAvailable().then(verdad => {
                  if (verdad == false) {
                    this.toast.show(this.translate.instant('NoGPS'));
                  } else {
                      // te localiza a traves del uso del GPS
                    this.map.locate({
                      setView: true, maxZoom: 15
                    }).on('locationfound', (e) => {
                      // crea una marca en la localizacion que te
encuentras
                      let markerGroup = leaflet.featureGroup();
                      let marker = leaflet.marker([e.latitude,
e.longitude]);
                      markerGroup.addLayer(marker);
                      this.map.addLayer(markerGroup);
                      // se llama al metodo que recibe la marca, la
latitud y la longitud
                      this.touchMark(marker, e.latitude, e.longitude);
                    }).on('locationerror', (err) => {

this.toast.showTop(this.translate.instant('LSGPS'));
                    });
                      }
                }).catch(e => {
                    console.error(e);
                    this.disablebutton = false;
                  });
              } else {
                this.toast.show(this.translate.instant('GPSS'));
                this.disablebutton = false;
              }
            });

          }
        });
      } else {
```

```
          // te localiza a traves del uso del GPS
          this.map.locate({
            setView: true, maxZoom: 15
          }).on('locationfound', (e) => {
            // crea una marca en la localizacion que te encuentras
            let markerGroup = leaflet.featureGroup();
            let marker = leaflet.marker([e.latitude, e.longitude]);
            markerGroup.addLayer(marker);
            this.map.addLayer(markerGroup);
            // se llama al metodo que recibe la marca, la latitud y la
longitud
            this.touchMark(marker, e.latitude, e.longitude);

          }).on('locationerror', (err) => {
            this.toast.showTop(this.translate.instant('LSGPS'));
            this.disablebutton = false;
          });
      }

        }
    }).catch(e => {

      if(!this.platform.is('android')){
        this.toast.show(this.translate.instant('browser'));
      }else{
        this.toast.show(this.translate.instant('noUser'));
      }


    });

  }


  /**
   * Metodo encargado de devolver el estado inicial
   * de los componentes de la vista de la vista cuando
   * el modal se cierra
   */
  onModalClose() {
    // abrimos el loading
    this.loading.show('');
    // eliminamos el mapa
    this.map.remove();
    // lo volvemos a cargar
    this.loadmap();
    // establecemos el valor  del boton en funcion del valor establecido
anteriormente
    // y cargamos las marcas correspondientes
```

```
    if (this.ocultaA === true) {

      this.ocultaA = false;
      this.nativeStorage.setItem('ocultaA', {property: this.ocultaA} );
    } else {
      this.ocultaA = true;
      this.nativeStorage.setItem('ocultaA', {property: this.ocultaA} );
    }
    this.chargeAllMarkAccident(this.ocultaA);

    if (this.ocultaM === true) {

      this.ocultaM = false;
      this.nativeStorage.setItem('ocultaM', {property: this.ocultaM} );
    } else {
      this.ocultaM = true;
      this.nativeStorage.setItem('ocultaM', {property: this.ocultaM} );
    }
    this.chargeAllMarkMeteorology(this.ocultaM);

    // llamamos al metodo para que nos localice
    this.locateme();
    // cerramos el modal
    this.loading.hide();

  }

  /**
   * Metodo que recibe la marca, la longitud y la latitud de la marca
creada vacia
   * Y lanza el modal para añadir un aviso en esa posición
   * @param mark marca que ha sido pulsada en el mapa
   * @param lat latitud de la ubicación de la marca
   * @param lng longitud de la ubicación de la marca
   */
  touchMark(mark: any, lat: any, lng: any) {

     this.cmm.show(AddAlertComponent, lat, lng, this);
     this.disablebutton = false;
     console.log(this.disablebutton);
 }
```

10.13 Tab3

- HTML

```
<ion-header *ngIf="usuario" no-border>
  <!--Todo esto es necesario para el cambio de tema de la toolbar-->
  <ion-toolbar>
    <ion-avatar  >
```

```
        <img [src]="imgV " style="height:200px; width:200px"
style="border:5px solid #fafafa">
    </ion-avatar>
    <div class="ion-text-center" style="margin-top:15px;">
        <ion-label style=" font-size:20pt; color:var( --ion-color-
dark);">
            {{ this.perfil.user}}</ion-label>
      </div>
  </ion-toolbar>
  </ion-header>

  <ion-header *ngIf="!usuario">
    <ion-toolbar color="primary">
        <ion-title>
              ZeroD
        </ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
      <ion-item *ngIf="!usuario" (click)="loginRegister()">
          <ion-label style="color:var( --ion-color-
dark);">{{ "loginregister" | translate }}</ion-label>
      </ion-item>
      <div *ngIf="usuario">
        <ion-item>
                <ion-label style="margin-top:30px;">
                    <h2 style="color:var( --ion-color-
dark);">Alertas</h2>
                </ion-label>
            <ion-range min="50" pin="true" step="10" max="1000"
id="valorKM" [(ngModel)]="valueKM" (ionChange)="setValueKM(valueKM)"
color="secondary">
                <ion-label style="color:var( --ion-color-dark);"
slot="start">50Km</ion-label>
                <ion-label style="color:var( --ion-color-dark);"
slot="end">1000Km</ion-label>
            </ion-range>
        </ion-item>
        <ion-item>
            <ion-label style="color:var( --ion-color-
dark);">{{ "activealert" | translate }} </ion-label>
            <ion-badge>{{this.activeAlert}}</ion-badge>
        </ion-item>

        <ion-item>
                <ion-label class="sc-ion-label-ios-h sc-ion-label-ios-s
ios hydrated">
```

```
                    <h2 style="color:var( --ion-color-
dark);">{{ "nightmode" | translate }}</h2>
                    <p style="color:var( --ion-color-tertiary-
tint);">{{ "descripnightmode" | translate }}</p>
                </ion-label>
            <ion-toggle [(ngModel)]="night" checked="{{this.night}}"
disabled="{{this.toggle}}" (ionChange)="manualMode(night)"></ion-toggle>
        </ion-item>

        <ion-item>
                <ion-label class="sc-ion-label-ios-h sc-ion-label-ios-s
ios hydrated">
                    <h2 style="color:var( --ion-color-
dark);">{{ "autonightmode" | translate }}</h2>
                    <p style="color:var( --ion-color-tertiary-
tint);">{{ "descripauto" | translate }}</p>
                </ion-label>
            <ion-toggle [(ngModel)]="autoNight"
checked="{{this.autoNight}}" (ionChange)="autoMode(autoNight)"></ion-
toggle>
        </ion-item>

        <ion-item (click)="logout()">
            <ion-label style="color:var( --ion-color-dark);">{{ "logout"
| translate }}</ion-label>
        </ion-item>
    </div>


  </ion-content>
```

- SCSS

```
ion-avatar{

    margin: auto;
    height: 200px;
    width: 200px;
    margin-top: 15px;
    margin-bottom: 15px;

}

ion-toolbar{
    --background:linear-gradient(to bottom,  var(--ion-color-primary)
0%,var(--ion-color-primary) 65%,var(--ion-color-light-shade) 65%, var(--
ion-color-light-shade) 100%);
}
```

```
ion-item{
    --background: none;
}

ion-badge{
    margin-right:20px;
    --background:var(--ion-color-primary);
    --color: #ffffff;
}

ion-range{
    --bar-background-active:var(--ion-color-primary);
}
```

- TS

```
import { Component} from '@angular/core';
import { Router } from '@angular/router';
import { CustomLoading } from '../custom-modal/custom-loading';
import { NativeStorage } from '@ionic-native/native-storage/ngx';
import { Camera, CameraOptions } from '@ionic-native/camera/ngx';
import { CloudserviceService } from '../servicios/cloudservice.service';
import { DomSanitizer } from '@angular/platform-browser';
import { ThemingService } from '../servicios/theming.service';



@Component({
  selector: 'app-tab3',
  templateUrl: 'tab3.page.html',
  styleUrls: ['tab3.page.scss']
})
export class Tab3Page {
  valueKM: number;
  night: boolean;
  autoNight:boolean;
  usuario: boolean = false;
  imgV: any;
  perfil: any;
  activeAlert;
  toggle:boolean;

  constructor(
    private loading: CustomLoading,
```

```
    private router: Router,
    private nativeStorage: NativeStorage,
    private camera: Camera,
    private sanitizer: DomSanitizer,
    private cloudS: CloudserviceService,
    private themeS: ThemingService

  ) {

  }
/**
 * Iniciamos variables del usuario si existe
 */
  ionViewWillEnter() {
    // si el usuario esta creado
    this.nativeStorage.getItem('user').then(user => {

        this.usuario = true;

        this.perfil = user;

        this.imgV = this.sanitizer.bypassSecurityTrustUrl(user.img);

        this.cloudS.getNumberOfAlert(this.perfil.email).then(n => {

          this.activeAlert = n ;

        });

        this.nativeStorage.getItem('distance').then(distance => {

          this.valueKM = distance.km;
          }).catch(e => {

          this.valueKM = 1000;

          });

        this.nativeStorage.getItem('themeAuto').then(au => {

              this.autoNight = au.autoNight;
              }).catch(e => {

              this.autoNight = false;

              });

        this.nativeStorage.getItem('toggle').then(d => {
```

```
                this.toggle = d.toggle;
                }).catch(e => {

                this.toggle = false;

                });
        if(!this.toggle){
          this.nativeStorage.getItem('themeManual').then(n => {

            if (n.night === true) {

              this.themeS.changeSkin(9);
              this.night = true;

            } else{

              this.themeS.changeSkin(11);
              this.night = false;

            }
          }).catch(e => {

            console.log('Aun no hay usuario o no se ha cambiado de
tema');
              });

        }
    // si el usuario no existe
    }).catch(error => {

      console.log('No hay usuario aun')

    });
  }

  /**
   * Metodo que navega hasta la pagina de registro/login de la aplicacion
   */
  loginRegister() {
    this.router.navigate(['register-login']);
  }


  /**
   * Establece el rango en Km de la distancia max que
   * nos cargara en las listas las alertas
   * @param valueKM Number
   */
  setValueKM(valueKM){
```

```
    this.nativeStorage.setItem('distance', {km: valueKM}).then(
      () => console.log('Stored item!'),
      error => console.error('Error storing item', error)
    );

  }

  /**
   * Metodo que controla el modo noche manual
   * Si es verdadero se cambia a modo noche
   * Si es falso se cambia a modo dia
   * @param night Boolean
   */
  manualMode(night) {
    if (night === true) {
      this.nativeStorage.setItem('themeManual', {night: night}).then(d =>
{
        console.log('Tema establecido a dark en manual');
        this.themeS. changeSkin(9);
        this.night= true;
      }
      ).catch(e => {
        console.log(e);
      });
    } else if (night === false ) {
      this.nativeStorage.setItem('themeManual', {night: night}).then(d =>
{
        console.log('Tema establecido a light en manual');
        this.themeS. changeSkin(11);
        this.night= false;
      }
      ).catch(e => {
        console.log(e);
      });

    }


  }

  /**
   * Metodo que controla el modo atomatico del modo noche
   * Tambien bloquea el boton de modo manual si este esta activado
   * @param autoNight Boolean
   */
  autoMode(autoNight) {
    // almacenamos el valor que recibimos del toggle en la bd
    this.nativeStorage.setItem('themeAuto', {autoNight: autoNight}).then(
```

```
      () => console.log('Boton tema automático'),
      error => console.error('Error storing item', error)
    );
    // si es falso
    // guardamos el valor del toggle en falso en la bd
    // establecemos su valor
    // comprobamos el valor del modo manual para establecer el tema
    if (autoNight === false) {
      this.toggle = false;
      this.nativeStorage.setItem('toggle', {toggle: false}).then(
        () => console.log('No esta deshabilitado el modo manual'),
        error => console.error('Error storing item', error)
      );
      this.nativeStorage.getItem('themeManual').then(r => {

        if (r.night === true) {

          this.themeS.changeSkin('9');

        } else if (r.night === false ) {
          this.themeS.changeSkin('11');


        }
});
    } else {
      // si es verdadero
      // almacenamos el valor en la bd del valor true del toggle
      this.toggle = true;
      this.nativeStorage.setItem('toggle', {toggle: true}).then(
        () => console.log('Esta deshabilitado el modo manual'),
        error => console.error('Error storing item', error)
      );
    }
  }


  /**
   * Cierra sesion en la aplicación
   */
  logout() {

    this.nativeStorage.clear().then(e => {
      console.log(e);
      this.usuario = false;
      this.router.navigateByUrl('/tabs/tab3');

    }).catch(error =>{
      console.log(error);
```

```
    });
  }

  getImagen(img) {
    if (img) {
      return;
    } else {
      return img;
    }
  }

}
```

10.14 Tab

- HTML

```
<ion-tabs>

  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="md-alert"></ion-icon>
      {{ "TAB1" | translate }}

    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="md-map"></ion-icon>
      {{ "TAB2" | translate }}
    </ion-tab-button>

    <ion-tab-button *ngIf="plataforma" tab="tab3">
      <ion-icon name="md-settings"></ion-icon>
      {{ "TAB3" | translate }}
    </ion-tab-button>


  </ion-tab-bar>

</ion-tabs>
```

- SCSS

```
ion-tab-bar{

    height: 50px;;
    background-color: var(--ion-color-primary);
```

```
}

ion-tab-button{
    background-color: var(--ion-color-primary);
    --color-selected: var(--ion-color-light);
    --color: var(--ion-color-dark-shade);

}
```

- TS

```
import { Component } from '@angular/core';
import { Platform } from '@ionic/angular';

@Component({
  selector: 'app-tabs',
  templateUrl: 'tabs.page.html',
  styleUrls: ['tabs.page.scss']
})
export class TabsPage {

  plataforma: Boolean = true;
    constructor(
      private platform: Platform
     ) {
       if(!this.platform.is('android')){
         this.plataforma = false;

       }


  }

}
```

- ROUTER

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TabsPage } from './tabs.page';

const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
```

```
      {
        path: 'tab1',
        children: [
          {
            path: '',
            loadChildren: '../tab1/tab1.module#Tab1PageModule'
          }
        ]
      },
      {
        path: 'tab2',
        children: [
          {
            path: '',
            loadChildren: '../tab2/tab2.module#Tab2PageModule'
          }
        ]
      },{
        path: 'tab3',
        children: [
          {
            path: '',
            loadChildren: '../tab3/tab3.module#Tab3PageModule'
          }
        ]
      },
      {
        path: '',
        redirectTo: '/tabs/tab1',
        pathMatch: 'full'
      }
    ]
  },
  {
    path: '',
    redirectTo: '/tabs/tab1',
    pathMatch: 'full'
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class TabsPageRoutingModule {}
```

10.15 App component

- TS

```
import { BackbuttonService } from './servicios/backbutton.service';
import { StatusBar } from '@ionic-native/status-bar/ngx';
import { ThemingService } from './servicios/theming.service';
import { NetworkService } from './servicios/network.service';
import { Component } from '@angular/core';
import { Platform, Events} from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { TranslateService } from '@ngx-translate/core';
import { Network } from '@ionic-native/network/ngx';
import { CustomToast } from './custom-modal/custom-toast';
import { Diagnostic } from '@ionic-native/diagnostic/ngx';
import { Sensors, TYPE_SENSOR} from '@ionic-native/sensors/ngx';
import { NativeStorage } from '@ionic-native/native-storage/ngx';




@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html'
})
export class AppComponent {
  light: number;



  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private translate: TranslateService,
    public events: Events,
    public network: Network,
    public networkS: NetworkService,
    public toast: CustomToast,
    public diagnostic: Diagnostic,
    public themeS: ThemingService,
    public sensor: Sensors,
    public bar: StatusBar,
    private nativeStorage: NativeStorage,
    private back: BackbuttonService
  ) {
    this.light = 0;
    this.initializeApp();
  }
```

```
  initializeApp() {

    this.platform.ready().then(() => {

      // Establecemos el color de inicio de la barra de estado

      this.nativeStorage.getItem('theme').then(d =>{
        if(d.theme === 'dark'){
          this.themeS.setTheme('dark')
        }else{
          this.themeS.changeSkin(11);
        }
      }).catch(e =>{
        this.themeS.changeSkin(11);
      });

      /*Comprobamos el idioma del dispositivo
      Si el dispositivo esta en español, la aplicacion se inicia en
español
      Si el dispositivo esta en otro idioma, por defecto se inicia en
ingles
      */


      if (this.translate.getBrowserLang() === 'es') {
        this.translate.setDefaultLang('es');
      this.translate.use('es');
      } else {
        this.translate.setDefaultLang('en');
        this.translate.use('en');
      }

      // Comprobamos la conexion a internet
      this.networkS.initializeNetworkEvents();

      // Evento que se ejecuta cuando la conexion esta inactiva
      this.events.subscribe('network:offline', () => {
        this.toast.show(this.translate.instant('noNetwork'));
      });

      // Evento que se  ejecuta cuando la conexion esta activa
      this.events.subscribe('network:online', () => {
        this.toast.show(this.translate.instant('Network') + ': ' +
this.network.type);
      });
    });

    /**
     * Comprobamos la calidad del  GPS
```

```
     * Falta mejora del servicio que
     * administra este evento
     *  */
    setInterval(() => { this.networkS.publicShowGPSEvent();
   }, 10000);

    this.events.subscribe('High accuracy', () => {
      this.networkS.colorSignalGPS('green');
    });
    this.events.subscribe('Battery saving', () => {
      this.networkS.colorSignalGPS('yellow');
    });
    this.events.subscribe('Device only', () => {
      this.networkS.colorSignalGPS('orange');
    });
    this.events.subscribe('Location off', () => {
      this.networkS.colorSignalGPS('red');
    });

    this.initSensor();

    this.splashScreen.hide();


  }


/**
 * Metodo encargado de inicializar el sensor
 * Si se ha activado la opción de modo noche automático
 * Se ejecuta cada 15 segundos comprobando la
 * Luz ambiental que recibee nuestro telefono
 * Y cambia el tema segun la intensida de luz
 */
initSensor() {

  setInterval(() => {

    this.nativeStorage.getItem('themeAuto').then(e => {

      if (e.autoNight === true) {

      this.sensor.enableSensor(TYPE_SENSOR.LIGHT).then(d => {
        console.log(d);
      }).catch( e => {
        console.log(e);
      });
      this.sensor.getState().then(d => {
      this.light = d[0];
```

```
      this.themeS.changeSkin(this.light);
      });
    }
    }).catch(e => {
        console.log('aun no se ha establecido el tema automático')
    });
  }, 13000);

}

}
```