

# Documentación APP Alerta Temprana

Julio 2022

## Contenido

1.	Ambiente de desarrollo.....	3
2.	Estructura del código fuente .....	4
3.	Páginas y lógica de la APP.....	5
3.1.	Splash.....	7
3.2.	Login .....	8
3.3.	Home .....	10
3.3.1.	Selección de Activo.....	10
3.3.2.	Operatividad y Daño.....	13
3.3.3.	Datos Adicionales .....	14
3.3.4.	Resumen .....	15
3.4.	Pendientes.....	20
3.5.	Ayuda.....	22
3.6.	Menú .....	23
4.	Mapas de flujo.....	24

## 1. Ambiente de desarrollo

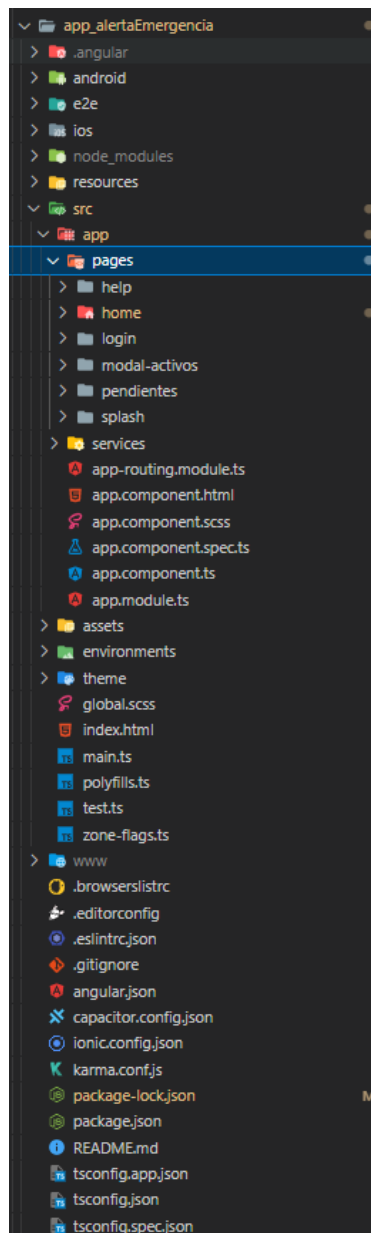
La aplicación está construida con Framework de desarrollo Ionic, en su versión 6.19.1. Este Framework permite desarrollar de manera híbrida, lo cual facilita realizar un solo desarrollo para sistemas operativos de iOS y Android bajo un mismo código fuente.

Para levantar el proyecto de manera local, es necesario instalar ciertos softwares, los cuales forman parte del cuerpo del proyecto

Software	Versión	Link
<b>Node JS</b>	v16.15.1	<a href="https://nodejs.org/es/">https://nodejs.org/es/</a>
<b>Angular CLI</b>	14.0.3	<a href="https://angular.io/guide/setup-local">https://angular.io/guide/setup-local</a>
<b>Ionic</b>	6.19.1	<a href="https://ionicframework.com/docs/intro/cli">https://ionicframework.com/docs/intro/cli</a>
<b>Capacitor</b>	3.6.0	<a href="https://capacitorjs.com/docs/getting-started">https://capacitorjs.com/docs/getting-started</a>
<b>Cordova</b>	11.0.0	<a href="https://www.npmjs.com/package/cordova">https://www.npmjs.com/package/cordova</a>
<b>Android Studio</b>	Bumblebee 2021.1.1	<a href="https://developer.android.com/studio">https://developer.android.com/studio</a>
<b>Xcode (En caso de usar Mac y compilar para iOS)</b>	13.4.1	<a href="https://apps.apple.com/cl/app/xcode/id497799835?mt=12">https://apps.apple.com/cl/app/xcode/id497799835?mt=12</a>
<b>Visual Studio Code</b>	1.69.1	<a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a>
<b>Gradle</b>	7.4.2	<a href="https://docs.gradle.org/current/userguide/installation.html">https://docs.gradle.org/current/userguide/installation.html</a>
<b>Git</b>	2.36.0	<a href="https://git-scm.com/downloads">https://git-scm.com/downloads</a>

## 2. Estructura del código fuente

El proyecto está desarrollado como se mencionó anteriormente en ionic y sus carpetas se distribuyen de la siguiente manera:



- **android**: carpeta que contiene la aplicación ya compilada para sistema operativo Android
- **ios**: carpeta que contiene la aplicación ya compilada para sistema operativo iOS
- **resources**: contiene el icono e imagen de inicio de la pantalla de carga de la aplicación
- **src**: carpeta que alberga todo el cuerpo de la aplicación
  - **app**: Esta carpeta contiene las vistas, servicios y archivos de enrutamiento de la aplicación
    - **pages**: Contiene todas las vistas de la aplicación y sus controladores individualmente
    - **services**: Archivos de conexión a servicios, en este caso a IBM MAXIMO
- **assets**: Carpeta que almacena todos los recursos estáticos, como imágenes, archivos de texto, entre otros.

Esos son los directorios con los cuales se trabaja mayoritariamente, ya que los otros documentos son en su gran parte estáticos sin alterar y quedan con su configuración original al crear el proyecto.

### 3. Páginas y lógica de la APP

Al ejecutar la aplicación, esta inicia desde el archivo **app.component.ts** el cual es archivo índice del aplicativo. Al desplegar la aplicación este ejecutara diferentes funciones

- Abrir página de splash
- Consultar si el usuario ya está logeado en la aplicación, con tal de activar el menú
- Consultar si hay conexión a internet
- Consultar si hay alertas pendientes por enviar en caso de tener conexión a internet

```
observadorConectado(){
  this.connectSubscription = this.network.onConnect().subscribe(() => {
    this._us.cargar_storage().then(()=>{
      if(this._us.conexion == 'no' || !this._us.conexion){
        this.presentToast('Conexión establecida').then(()=>{
          setTimeout(()=>{
            this.buscarAlertasPendientes()
          },4000)
        })
      }
    })
    this.storage.setItem('conexion', 'si');
    localStorage.setItem('conexion','si')
    this._us.cargar_storage().then(()=>{})
  })
});
}
```

Función que está a la escucha de si hay conexión a internet

```
buscarAlertasPendientes(){
  if(this.platform.is('capacitor')){
    this.sqlite.create({name:'mydbAlertaTemprana',location:'default',createFromLocation:1}).then((db:SQLiteObject)=>{
      this.db = db;
      this.db.transaction(async tx=>{
        this.db.executeSql('SELECT * FROM alerta', []).then((data)=>{
          if(data.rows.length > 0){
            this.pendientes = true;
            this.presentToast('Hay '+data.rows.length+' alertas pendientes por enviar')
          }else{
            let options: NativeTransitionOptions = {
              direction:'right',
              duration:500
            }
            this.nativePageTransitions.fade(options);
            this.navCtrl.navigateRoot('/home')
            this.pagina = 'home'
          }
        })
      })
    })
  }
}
```

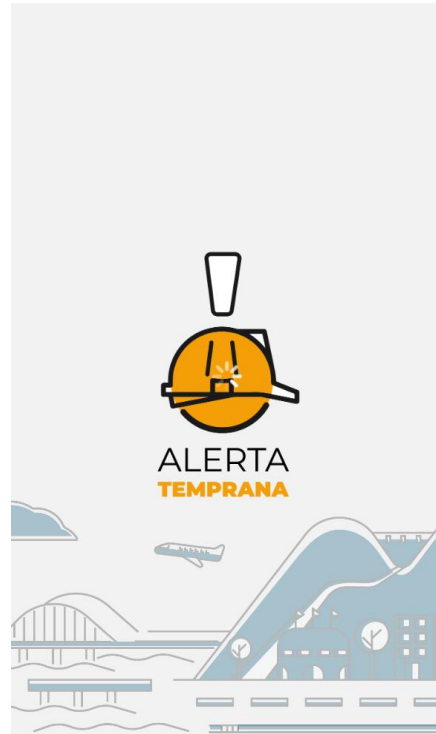
Función para buscar si hay alertas pendientes por enviar

### 3.1. Splash

Esta página es una pantalla de carga mientras se analizan datos de si existe un usuario logeado o no, en caso de existir lo redireccionara directamente al home de la aplicación, en caso de no, lo envía al login de la APP.

```
ngAfterViewInit(){
  setTimeout(()=>{
    if(this.platform.is("capacitor")){
      let options: NativeTransitionOptions = {
        direction:'left',
        duration:200
      }
      this.nativePageTransitions.fade(options);
    }
    this._us.cargar_storage().then(()=>{
      if(this._us.usuario){
        this._mc.enable(true,'first')
        this.navctrl.navigateRoot('/home')
      }else{
        this._mc.enable(false,'first')
        this.navctrl.navigateRoot('/login')
      }
    }).catch(()=>{
      this._mc.enable(false,'first')
      this.navctrl.navigateRoot('/login')
    })
  }, 7000);
}
```

Función para revisar si existe el usuario



Pantalla de carga de la APP

### 3.2. Login

Esta pantalla es para iniciar sesión en la aplicación, esto se realiza ingresando las credenciales del usuario registrado en los sistemas de MAXIMO. Si el usuario ingresado existe, el servicio devolverá datos importantes del usuario, como región a la cual pertenece o tiene acceso, dirección dentro del MOP, ID, grupo, estado, entre otros. Estos datos se almacenan en el storage del dispositivo, para hacer uso de ellos cada vez que se necesiten para consulta de datos o llamadas a servicios en los cuales se ocupen alguno de estos campos.

```

iniciar(){
  this.presentLoader().then(()=>{
    this.form.disable()
    if(this.platform.is('capacitor')){
      this._us.login(this.form.value).subscribe((res:any)=>{
        console.log('LOGIN ->>> ',res)
        if(res && res.status == '200'){
          this._us.xmlToJson(res).then((result:any)=>{
            let path = result['SOAPENV:ENVELOPE']['SOAPENV:BODY'][0].QUERYMOP_USUARIO_DOHRESPONSE[0].MOP_USUARIO_DOHSET[0].MAXUSER[0]
            var grupo = '';
            path.GROUPUSER.forEach((g,i)=>{
              grupo+=g.GROUPNAME[0]
              if((i+1) < path.GROUPUSER.length){
                grupo+=", "
              }
            })
            this._us.usuario = {
              DEFSITE:path.DEFSITE[0],
              GROUPUSER:grupo,
              LOGINID:path.LOGINID[0],
              PERSON:{
                CARGOCOMP:path.PERSON[0].CARGOCOMP[0],
                DFLTAPP:path.PERSON[0].DFLTAPP[0],
                DISPLAYNAME:path.PERSON[0].DISPLAYNAME[0],
                DPTOUNI:path.PERSON[0].DPTOUNI[0],
                INSTITUCION:path.PERSON[0].INSTITUCION[0],
                PERSONID:path.PERSON[0].PERSONID[0],
                PROFESION:path.PERSON[0].PROFESION[0],
                STATEPROVINCE:path.PERSON[0].STATEPROVINCE[0],
                TIPOBOD:Boolean(String(path.PERSON[0].TIPOBOD[0]['$']['XSI:NIL']).replace(/["']/gi,""))
              },
              STATUS:path.STATUS[0]['_'],
              USERID:path.USERID[0]
            }
            if(this._us.usuario.STATUS == 'ACTIVE'){
              this._us.saveStorage(this._us.usuario,this.form.value)
              this._us.cargar_storage().then(()=>{
                this._mc.enable(true,'first')
                this.loader.dismiss()
                let options: NativeTransitionOptions = {
                  direction:'left',
                  duration:500
                }
                this.nativePageTransitions.slide(options);
                this._us.nextmessage('usuario_logeado')
                this.navctrl1.navigateRoot('/home')
              })
            }else{
              this.presentAlert('¡Atención!', 'EL usuario esta inactivo')
            }
          })
        }
      })
    }
  })
}


```

Función de inicio de sesión





Bienvenido al sistema de ingreso de Emergencias



**ALERTA  
TEMPRANA**

Usuario

Contraseña 

**INGRESAR**

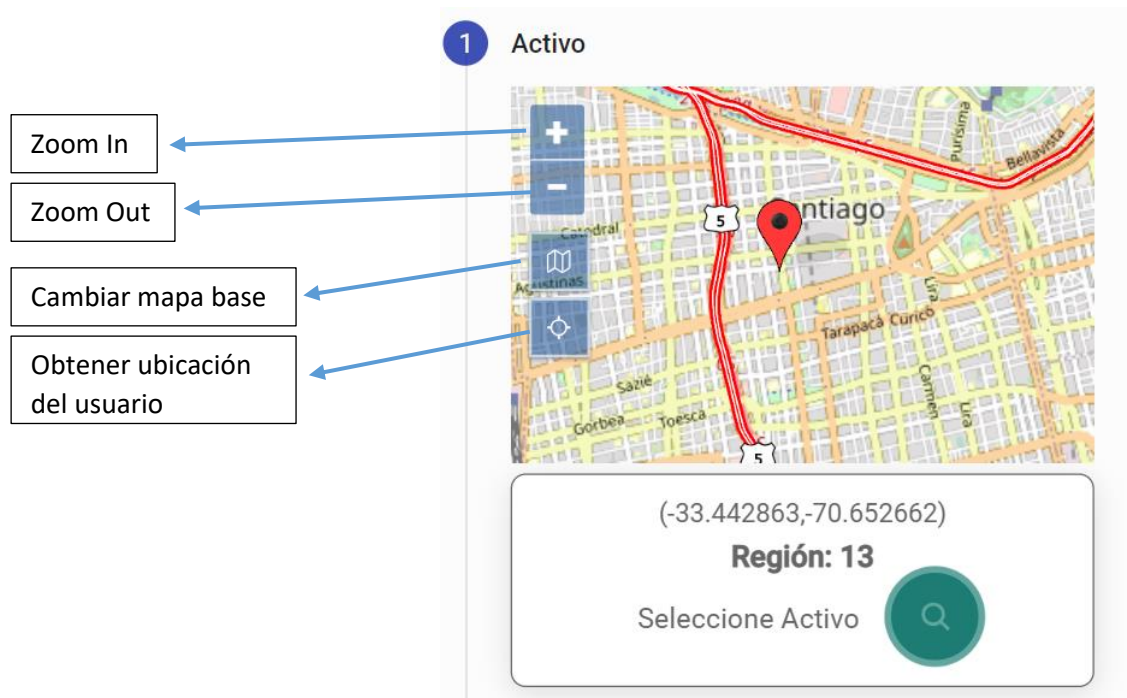
The bottom of the screen features a stylized illustration of a cityscape with a bridge, a train, and buildings.

Pantalla de inicio de sesión

### 3.3. Home

#### 3.3.1. Selección de Activo

Esta es la pantalla principal de la aplicación, contiene mapa base con activos, red vial u otros servicios que sean pertinentes o necesarios mostrar en el mapa. EL mapa, posee 4 botones de uso, los 2 primeros son para realizar Zoom Ino Zoom out, el tercero es para cambiar el tipo de mapa que se está visualizando (entre mapa de tipo satelital o base) y el ultimo es para obtener la ubicación del usuario.



Información de botonera en el mapa

```
geolocate(){
  this.presentLoader('Localizando ...').then(()=>{
    this.geolocation.getCurrentPosition().then((resp) => {
      this.view.setCenter(olProj.transform([resp.coords.longitude,resp.coords.latitude], 'EPSG:4326', 'EPSG:3857'))
      this.marker.getGeometry().setCoordinates(this.view.getCenter());
      this.view.setZoom(15)
      this.loader.dismiss();
      this.obtenerUbicacionRegion()
    }).catch((error) => {
      console.log('Error getting location', error);
    });
  });
}
```

Función para obtener la ubicación del usuario

```
changeMap(){  
  if(this.modo == 'osm'){  
    this.osm.setVisible(false)  
    this.baseLayer.setVisible(true)  
    this.modo = 'satelite'  
  }else{  
    this.osm.setVisible(true)  
    this.baseLayer.setVisible(false)  
    this.modo = 'osm'  
  }  
}
```

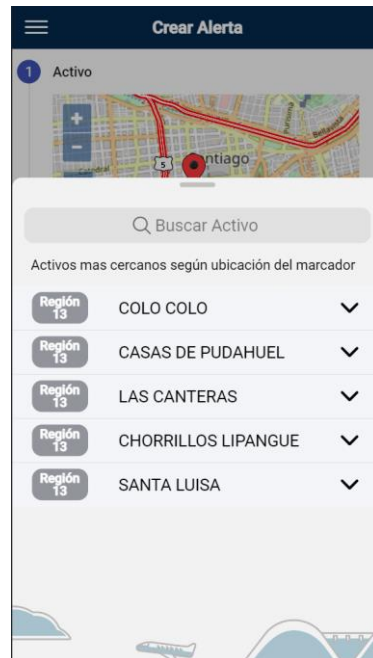
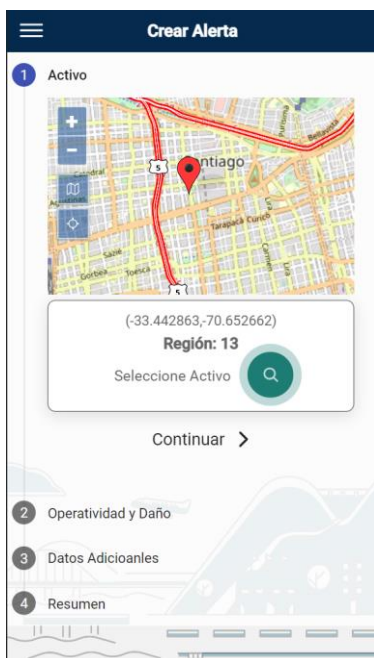
*Función para cambiar el tipo de mapa base*

Se cargan los **niveles de alerta** y **estados de operatividad**. En los casos que usuario sea de DOH, DAP y DOP, estos cargarán activos, los cuales podrán ser seleccionados, obteniendo así el código del activo. En caso de no ser ninguno de los anteriormente mencionados, no mostrará la opción de seleccionar activo, y solo bastara con la posición del marcador en el mapa. Al hacer click en el botón de seleccionar activo, se abrirá una pop-up con información de los 5 activos más cercanos al marcador, de igual manera, este pop-up contiene un buscador para buscar otros activos independientemente si no está entre los 5 más cercanos.

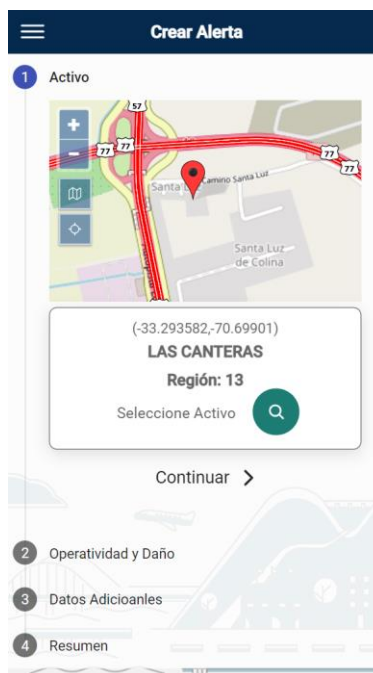
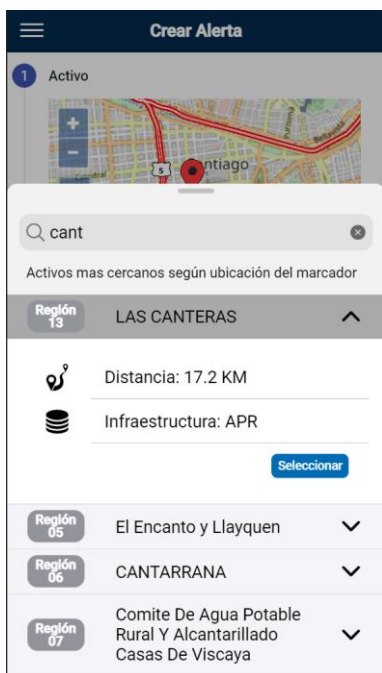
Al mover el mapa, el puntero se recalcula al centro del mapa y obtiene la región en la cual está posicionado, esto es por medio de una capa en geoJSON de Chile, la cual se intersecta con la posición del marcador para saber en qué región está posicionado.

```
obtenerUbicacionRegion(){  
  this.marker.getGeometry().setCoordinates(this.view.getCenter());  
  var curr = olProj.toLonLat(this.view.getCenter());  
  this.dataPosicion.lat = Number(curr[1].toFixed(6));  
  this.dataPosicion.lng = Number(curr[0].toFixed(6));  
  var region;  
  this.regiones = this.chile.getSource().getFeatures();  
  for (var i in this.regiones) {  
    var polygonGeometry = this.regiones[i].getGeometry();  
    var coords = olProj.toLonLat(this.marker.getGeometry().getCoordinates());  
    if (polygonGeometry && typeof polygonGeometry != "undefined") {  
      if (polygonGeometry.intersectsCoordinate(coords)) {  
        region = this.regiones[i].get("region") + "";  
        if (region.length == 1) region = "0" + region;  
        this.dataPosicion.region = region;  
      }  
    }else{  
      console.log('fuera de regiones')  
    }  
  }  
}
```

*Función para obtener el valor de la región en la cual se encuentra el marcador posicionado*



Home de la APP y pop-up de activos desplegado

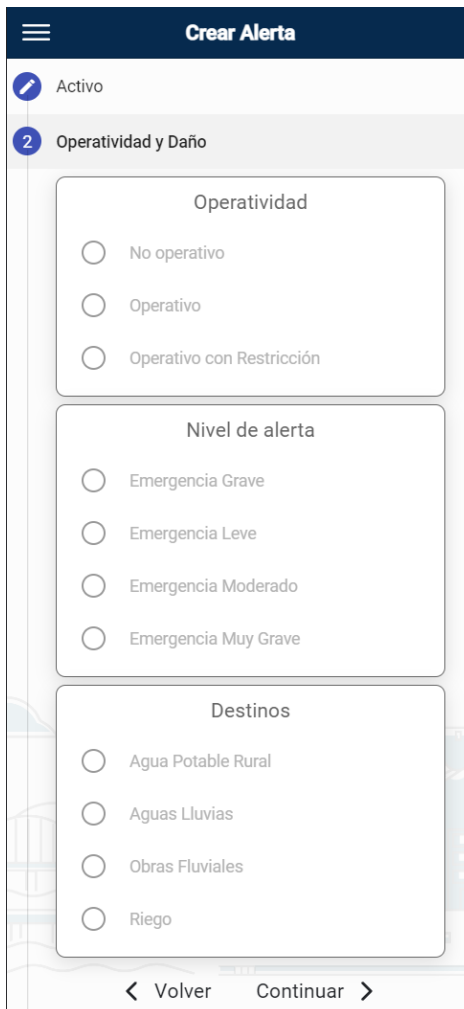


Selección de activo y posicionamiento del marcador en la posición del activo seleccionado

Una vez seleccionado el activo o posicionado el marcador en el punto deseado, se sigue con los otros ítems a completar para enviar la alerta.

### 3.3.2. Operatividad y Daño

El segundo ítem a completar, es el de Operatividad y Daño, el cual tiene 3 cajas con opciones, Operatividad, Nivel de Alerta y Destino. En cada uno de estos puntos, se consta con una lista de elecciones, en la cual, se debe seleccionar uno de cada uno según corresponda el caso de la alerta.



**Crear Alerta**

1 Activo

2 Operatividad y Daño

**Operatividad**

- ☐ No operativo
- ☐ Operativo
- ☐ Operativo con Restricción

**Nivel de alerta**

- ☐ Emergencia Grave
- ☐ Emergencia Leve
- ☐ Emergencia Moderado
- ☐ Emergencia Muy Grave

**Destinos**

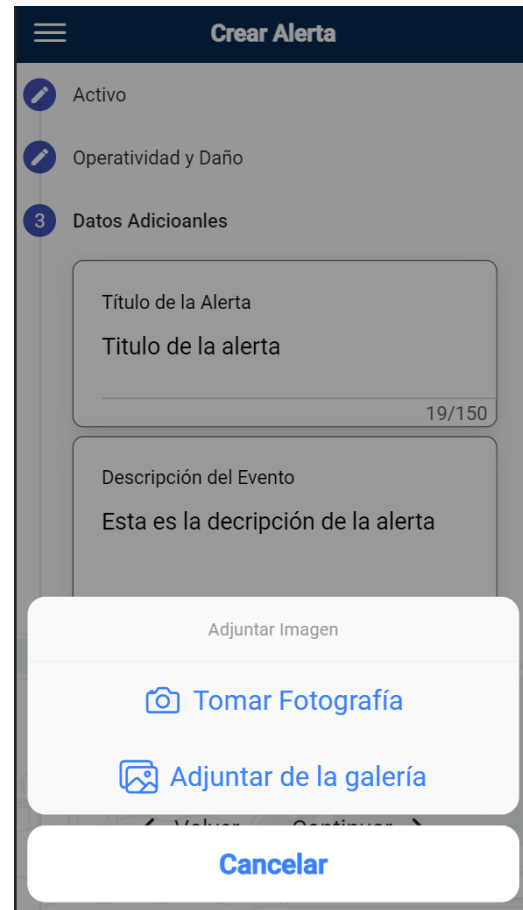
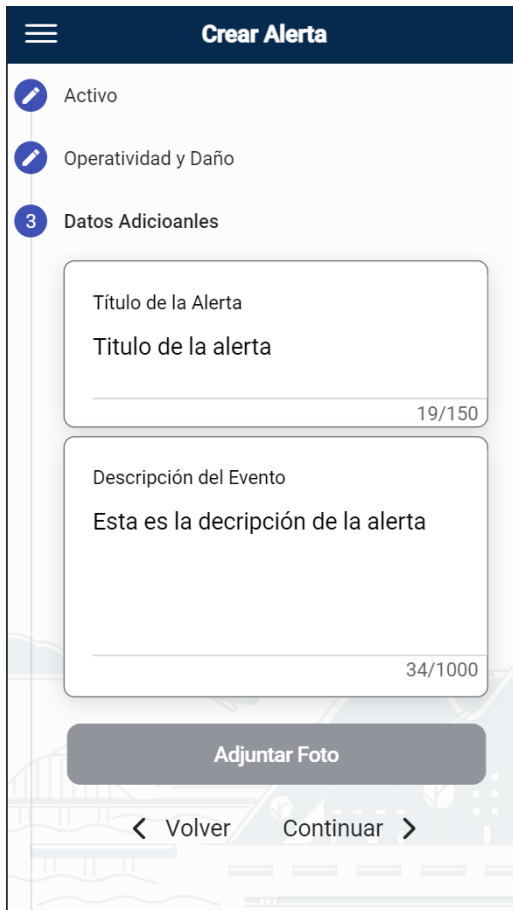
- ☐ Agua Potable Rural
- ☐ Aguas Lluvias
- ☐ Obras Fluviales
- ☐ Riego

< Volver Continuar >

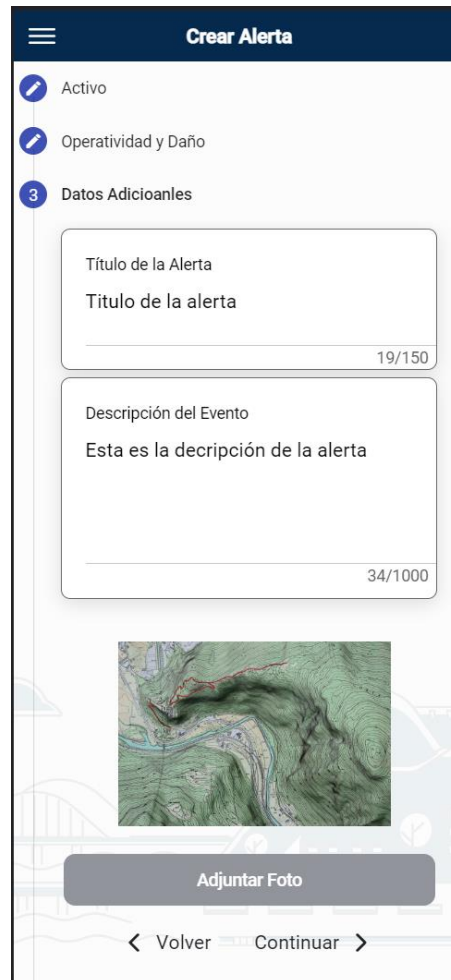
*Ítem de Operatividad y Daño*

### 3.3.3. Datos Adicionales

Como tercer ítem, se encuentra el de Datos Adicionales, el cual se ingresan los últimos datos de la alerta, los cuales son el título, descripción y una imagen opcionalmente. En caso de querer adjuntar una imagen, se consulta si se quiere tomar una fotografía o adjuntar una de la galería de fotos del usuario. La imagen se guardara en el storage del dispositivo y en base64, ya que es la codificación que debe tener para ser adjuntada en el servicio de MAXIMO.



*Ítem de Datos adicionales y selección de imagen a adjuntar*



*Imagen adjunta exitosamente*

#### 3.3.4. Resumen

Una vez listo este ítem, queda solo el último, el cual es un resumen de todo lo realizado, y de las opciones obligatorias que deben estar listas. En caso de estar todo “ok” se activara el botón para enviar la alerta, en caso contrario, el botón se desactivara e indicara en la tabla el dato faltante.

Crear Alerta

Activo

Operatividad y Daño

Datos Adicioanles

4 Resumen

Dato	Información	Estado
Título	Título de la alerta	✓
Destino	DOH-RIEG	✓
Operatividad	Operativo	✓
Nivel	Leve	✓
Mensaje	Esta es la decripción de la alerta	✓

< Volver

REPORTAR

Crear Alerta

Activo

Operatividad y Daño

Datos Adicioanles

4 Resumen

Dato	Información	Estado
Título		!
Destino	DOH-RIEG	✓
Operatividad	Operativo	✓
Nivel	Leve	✓
Mensaje	Esta es la decripción de la alerta	✓

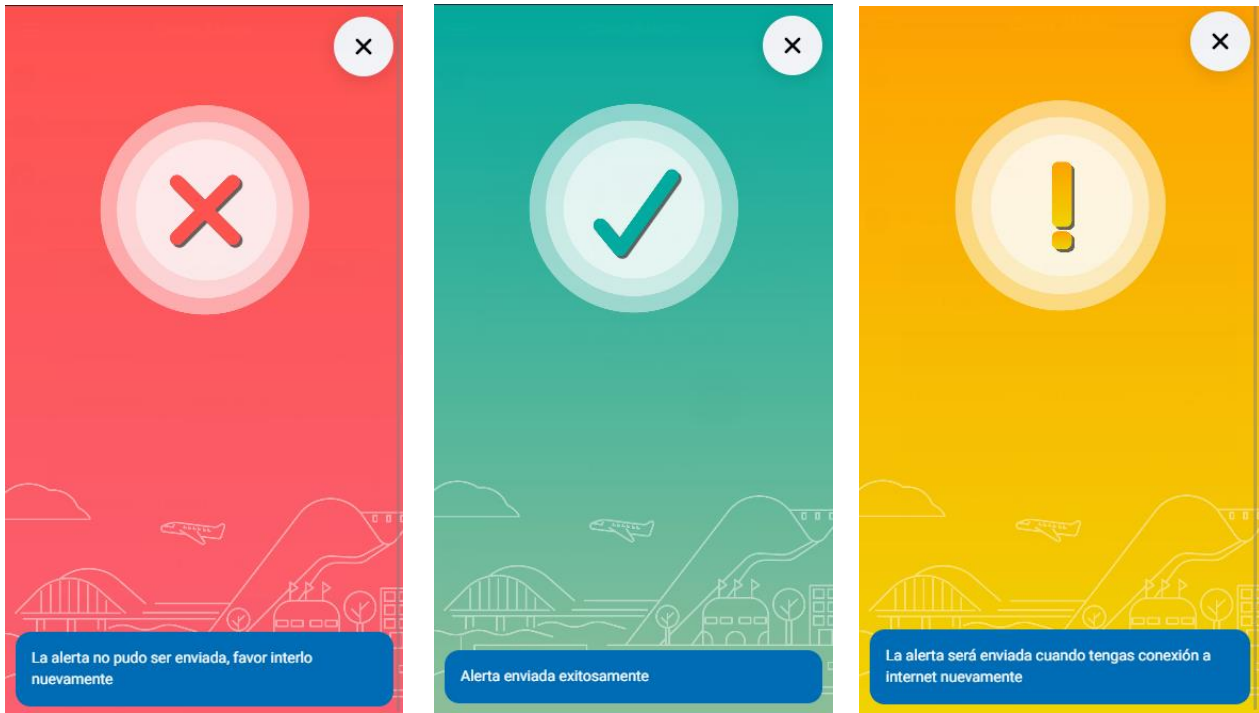
< Volver

REPORTAR

*Ítem de Resumen, caso exitoso y aso con dato faltante*

Al momento de enviar la alerta, se pueden dar 3 casos: **envío de alerta exitoso**, **envío de alerta fallido** y **envío de alerta pendiente**. En caso de ser exitoso, mostrará un mensaje que se ha enviado exitosamente la alerta y se reiniciará el formulario y mapa al punto de origen. En el caso de haber fallado el envío, mostrará un mensaje indicando que no se pudo enviar y que se intente nuevamente, y el último caso se da cuando al momento de enviar la alerta no se encuentra con internet, en este caso mostrará un mensaje que la alerta ha sido almacenada y será enviada cuando se vuelva a tener conexión. Solo se pueden almacenar un máximo de 7 alertas pendientes por enviar.





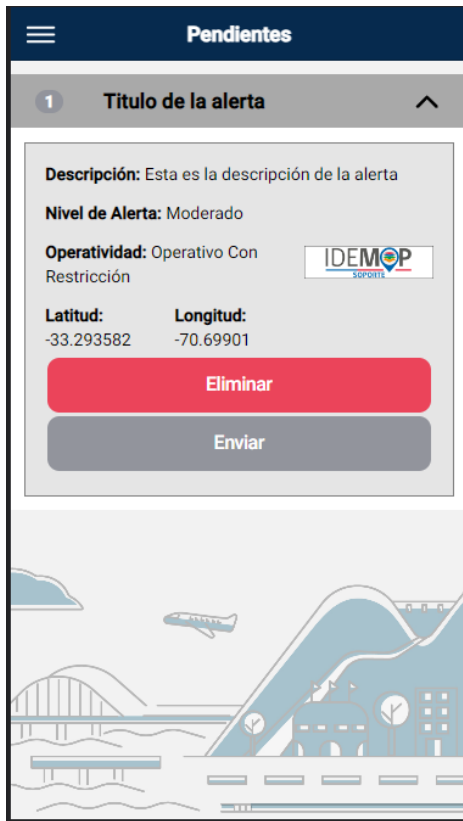
*Tipos de respuestas por envío de alerta*

### Envío de alerta con estado pendiente

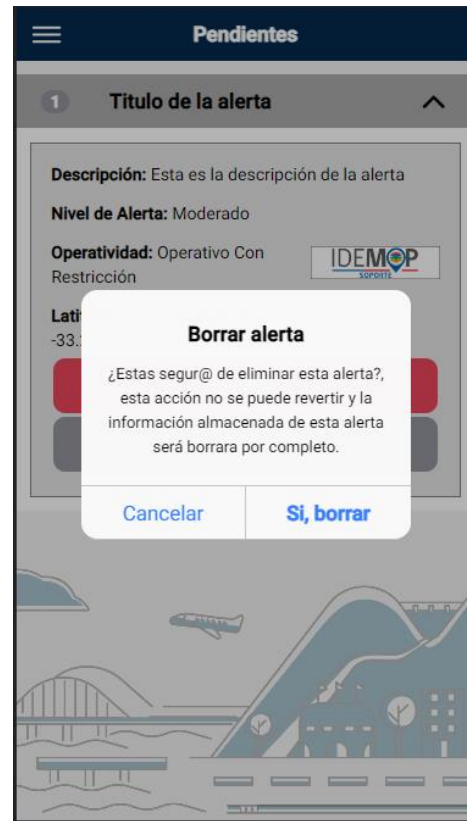
```
}else{
  this._us.enviarAlerta(data).subscribe(res=>{
    console.log('***** RESPUESTA AL ENVIAR FORMULARIO *****', res)
    this.estadoEnvioAlerta = 'exitoso'
    var modal = document.querySelector('ion-modal');
    this.deleteImage(this.images[0])
    this.volverInicio()
    modal.present().then(()=>{
      this.presentToast('Alerta enviada exitosamente');
    })
  },err=>{
    this.estadoEnvioAlerta = 'fallido'
    var modal = document.querySelector('ion-modal');
    modal.present().then(()=>{
      this.presentToast('La alerta no pudo ser enviada, favor interlo nuevamente');
    })
    console.log('***** ERROR ENVIAR ***** ',err)
  })
}
```

*Envío de alerta con caso exitoso o fallido*





*Pantalla de pendientes*

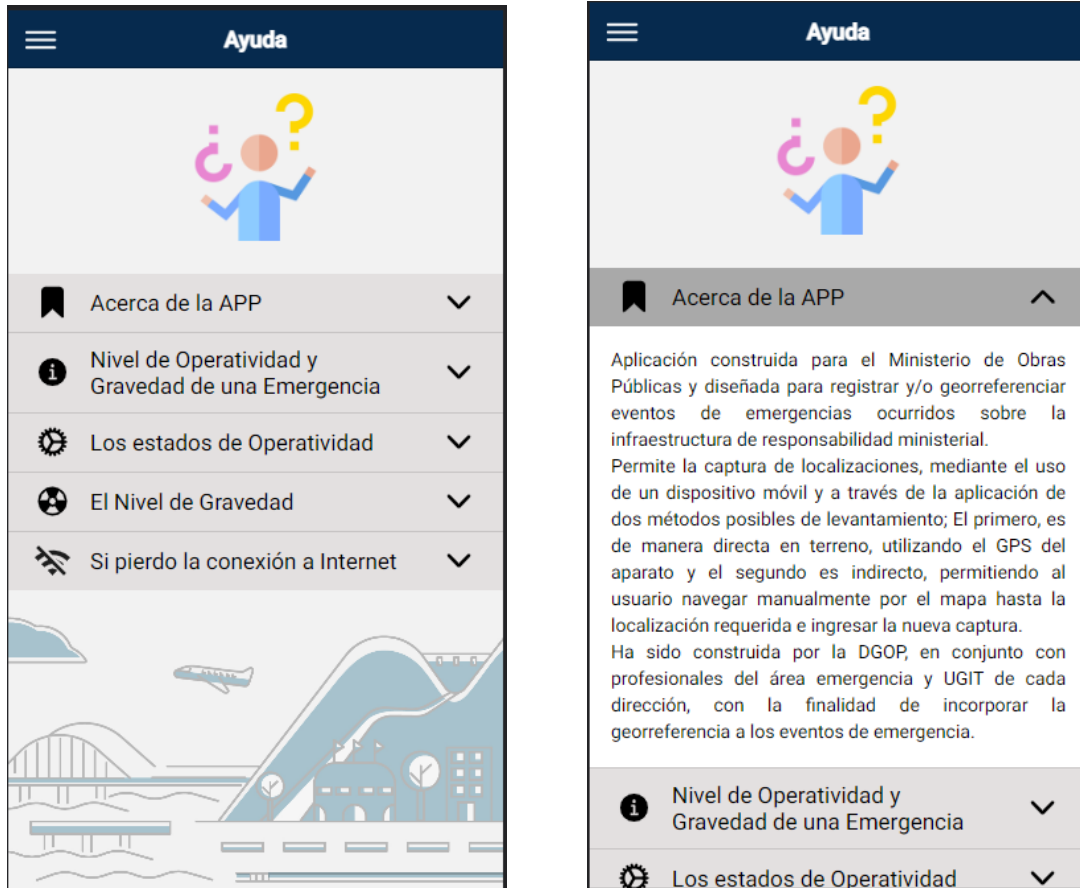


*Borrar alerta pendiente*

Automáticamente al tener conexión a internet se iniciará el envío de alertas pendientes, pero en caso que alguna de las alertas no haya sido enviada de forma automática, se puede hacer click en el botón enviar que posee cada alerta pendiente. De la misma forma, cada alerta posee la funcionalidad de poder ser eliminada de la base de datos de pendientes por enviar. Una vez hayan sido todas las alertas que están en pendientes enviadas, se redireccionara automáticamente al home de la aplicación, y la pantalla de pendientes desaparecerá del menú.

### 3.5. Ayuda

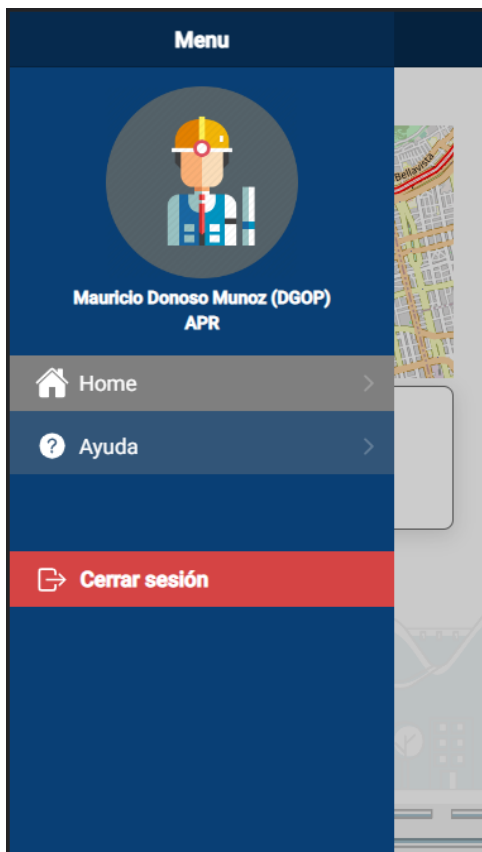
Esta pantalla consta con información relevante de la aplicación, como funciona, operatividad, que pasas sin se cae la conexión, entre otros.



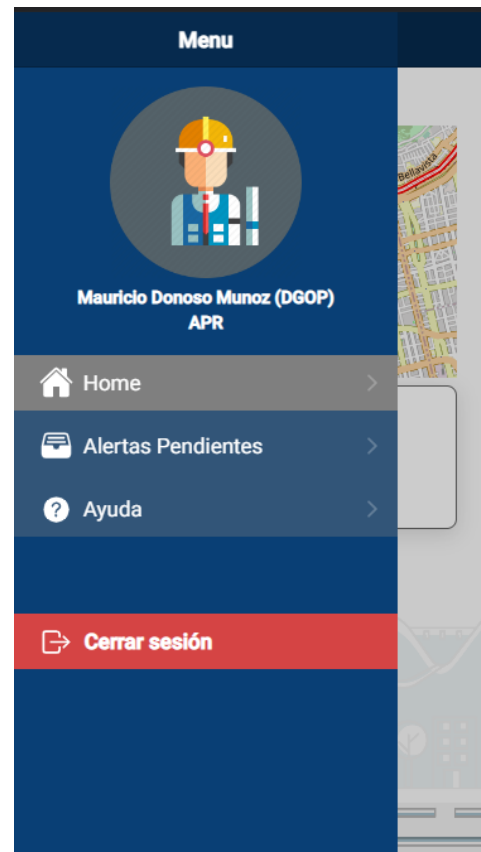
Pantalla de ayuda de la APP

### 3.6. Menú

La pantalla o página del menú, es accesible por todas las pantallas, si solo si, el usuario esta logeado. La lista de páginas a acceder desde el menú varía dependiendo si hay alertas pendientes o no.



*Página de menú normal*



*Página de menú con pendientes ingresados*

## 4. Mapas de flujo

