7 – PERIFÉRICOS Y SERVICIOS DE GOOGLE

CAMARA

Para emplear la Cámara desde mi aplicación, puedo hacerlo de dos maneras:

- Directamente: A través de las clases del paquete/API android.hardware.camera2

- Indirectamente a través de un Intent (reusando apps ya hechas)

CAMARA

Lo más usual es el modo indirecto. Sólo sería apropiado hacerlo por uso directo si la cámara es algo fundamental o quiero hacer una app de captura de vídeo.

En tal caso, debería hacer uso del API referido anteriormente y seguir los pasos descritos en la documentación

https://developer.android.com/guide/topics/media/camera.html#custom-camera

CAM vía INTENT

Debemos lanzar un intent con la acción ACTION_IMAGE_CAPTURE

Indicando de forma opcional la ruta (expresada como uri) del fichero de destino

Y lanzando la actividad exigiendo retorno

CAM vía INTENT

Si quiero capturar vídeo, lo que debo cambiar es el nombre de la acción a ACTION_VIDEO_CAPTURE

Google recomienda el empleo de URIs en ambos casos (foto/video) por temas de compatibilidad

CAM vía INTENT

```
intent = new Intent (MediaStore.ACTION_IMAGE_CAPTURE);

fileUri = ObtenerURI();

intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

startActivityForResult(intent, CODIGO_PETICION)
```

CAMARA

Hay cierta controversia sobre los permisos debido al elevado número de modelos de cámara y fabricantes. Con lo que es mejor curarse en salud y agregar todos los permisos en el Manifest (de lecutra, escritura y uso de la cámara y sus caracerísticas)

CAMARA PERMISOS

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
```

DESTINO

Hay por defecto dos rutas que puedo emplear para guardar mis imágenes.

Una carpeta PÚBLICA

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)

Una asociada a mi APP

Context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)

StartActivityForResult

Es un mecanismo previsto por Android para intercomunicar actividades.

Para ello puedo emplear intents implícitos o explícitos.

StartActivityForResult

En la Actividad “Hija”, debo setear el restulado de la operación y un intent devuelto al que puedo incorporar información de retorno

```
setResult(RESULT_OK, intent);
finish();
```

StartActivityForResult

En la Actividad llamante, haré uso del método startActivityForResult y deberé sobreescribir el método onActivityResult, que será invocado por Android a modo de Callback o al finalizar la subactividad invocada

StartActivityForResult

En la clase Activity hay unas constantes predefinidas que podemos usar para expresar el resultado de la ejecución de la subactividad

RESULT_CANCELED (0)

RESULT_OK (-1)

Intent.createChooser()

Al lanzar un intent puedo pasarlo directamente o hacerlo dentro del método createChooser

```
startActivity (Intent.createChooser(myIntent))
```

Intent.createChooser()

La ventaja de envolver nuestro intent en el *Chooser,* es que en caso de *match*, saldrán todas las aplicaciones coincidentes con nuestro mensaje personal (aún habiendo preseleccionado una antes) y si no hay coincidencia, nos saldrá un mensaje del Sistema, informándonos del hecho.

Compartir (Enviar)

Para compatir algún documento desde mi app hacia otra que permita el envío de documentos, basta con lanzar un intent con la acción que incluya el tipo mime adecuado y la URI de los documentos

Compartir (Enviar)

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
Uri uri = Uri.fromFile (new File ("/storage/8.jpg"));
shareIntent.putExtra(Intent.EXTRA_STREAM, uri);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, "Envia foto ... "));
```

LOCAILIZACIÓN

Android trae incorporado un servicio (**LocationManager**) en una API, bajo el paquete **android.location** que permite gestionar el posicionamiento (**Location**) por diferentes proveedores de localización (**LocationProvider**): antenas, redes wifi y GPS

LOCALIZACIÓN - API

La guía básica de la API está presente en

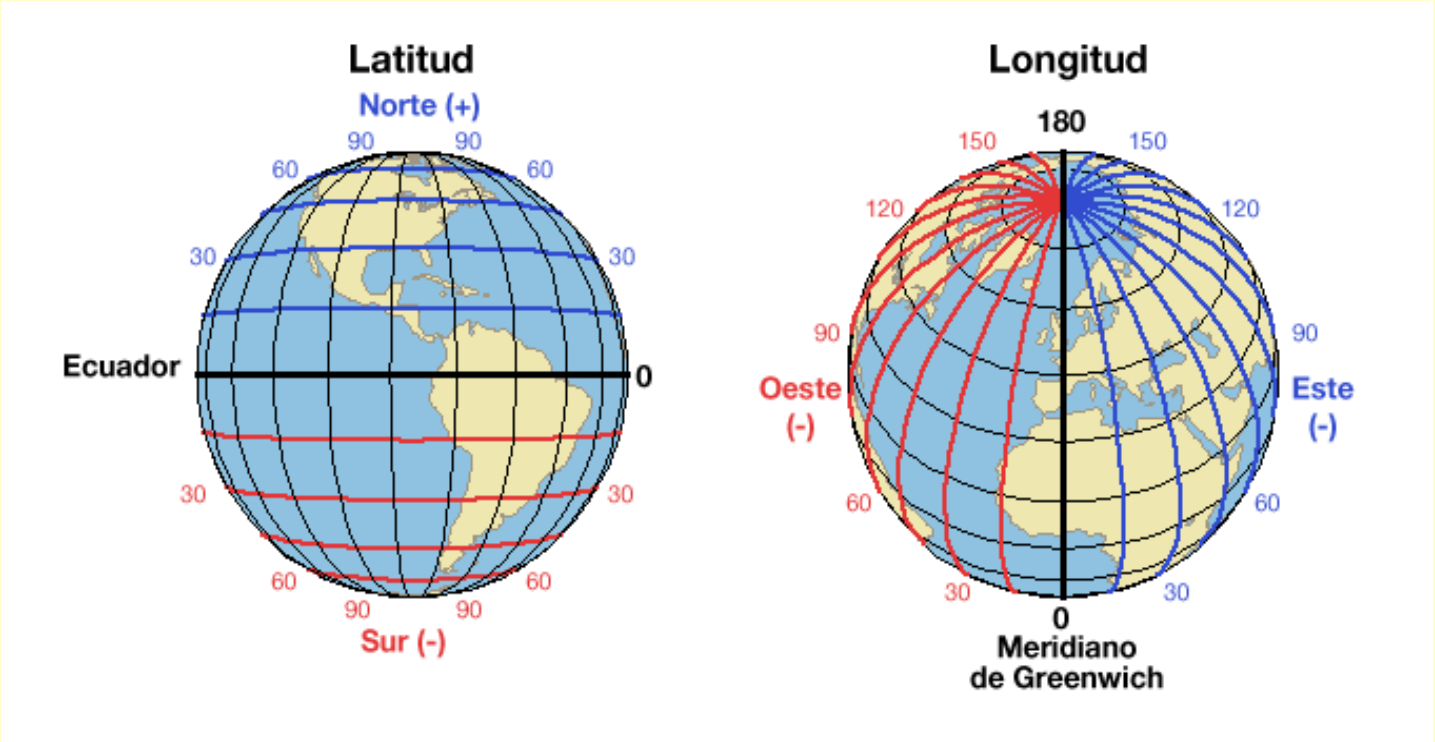
https://developer.android.com/reference/android/location/package-summary.html

LOCALIZACIÓN

Para situar un punto en el plano terrestre se emplean dos puntos: la latitud y la longitud.

Ambos se pueden expresar en grados, minutos y segundos. Un tercer parámetro es la altitud, medida en metros, es un atributo no siempre disponible (dependiendo del hardware)

LOCALIZACIÓN



LOCALIZACIÓN Permisos

Para poder acceder a los servicios de localización es necesario incluir en el Manifest

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

LOCALIZACIÓN Permisos

Además de conceder permisos al instalar -”normales”-, desde la versión 23 (A 6), existe el concepto de permisos “peligrosos”, que atañen a la privacidad del usuario y que requieren además una autorización expresa por parte del usuario en tiempo de ejecución

Permisos Peligrosos

Grupo: Permisos

Calendar: Read_Calendar / Write_Calendar

Camera: Camera

Contacts: Read_Contacts / Write_Contacts / Get_Accounts

Location: Access_Fine_Location / Access_Coarse_Location

Microphone: Record_audio

Permisos Peligrosos

Grupo: Permisos

Phone: Read_Phone_State / Call_Phone / Read_Call_Log / Write_Call_Log / Add_Voice_Mail / Use_Sip / Process_Outgoing_Calls

Sensors: Body_Sensors

Sms: Send_Sms / Recive_Sms / Read_Sms / Recieve_Wap_Push

Permisos Peligrosos

Grupo: Permisos

Sms: Receive_Mms

Storage: Read_External_Storage / Write_External_Storage

LOCALIZACIÓN Permisos

SI ESTOY EN UNA VERSIÓN DE ANDROID MAYOR O IGUAL A 6, EL CONJUNTO DE PERMISOS PELIGROSOS, DEBERÁ SER AL MENOS ACEPTADO UNA VEZ EN EJECUCIÓN (PESE HABERLO SIDO DURANTE LA INSTALACIÓN)

LOCALIZACIÓN Permisos

ADEMÁS, A LA HORA DE OBTENER LA LOCALIZACIÓN (Y NO OBTENER ASÍ VALORES NULOS), DEBO TENER ACTIVADA LA UBICACIÓN DESDE EL SETTING, LO QUE PUEDE REQUERIR LANZAR UN INTENT PARA DEMANDAR SU ACTIVACIÓN AL USUARIO

LOCALIZACIÓN

- LocationManager
- LocationProvider
- Location
- Address
- LocationListener

LocationManager

Esta clase me permite acceder al servicio de localización de Android. Se obtiene una instancia a través del contexto:

```
LocationManager locationmanager = (LocationManager)
Context.getSystemService(Context.LOCATION_SERVICE)
```

LocationManager

En ella están definidos los diferentes proveedores a modo de constantes

GPS_PROVIDER

NETWORK_PROVIDER

PASSIVE_PROVIDER

LocationManager

Y ofrece dos métodos para que nuestra app se mantenga a alerta de los cambios en la posición del dispotivo

requestLocationUpdates

removeUpdates

CLASE LocationProvider

Es la clase padre de los distintos tipos de acceso a la posición del dispositivo (proveedores).

Cada proveedor, presenta unas peculiaridades en cuanto a funcionamiento, precisión y consumo de batería.

LocationProvider GPS

Usa el Chip de GPS del dispositivo (hw)

Usa satélites

Alta precisión

Tarda varios segundos (lento)

Gasto alto de batería

En exteriores (no va en edificios altos)

LocationProvider Network

Usa las torres de telefonía (id celdas) y las redes Wi-fi (dirección MAC)

Menor precisión

Rápido

Menor gasto de batería

Funciona en interiores y exteriores

LocationProvider Passive

No determina la posición (al contrario que los anteriores): sólo obtiene la posición determinada por otros proveedores. Baja precisión. No requiere localización activada.

Android lo emplea para generar estadísticas y captar información que alimenta sus bases de datos de geolocalización

LOCALIZACIÓN Permisos

ACCESS_COARSE_LOCATION . - Sólo puedo usar el proveedor netowrk

ACCESS_FINE_LOCATION.- Puedo usar tanto network, como GPS y passive (passive puede usar indirectamente GPS)

CLASE Address

Es una clase incluida en la API que representa una dirección física e incluye atributos como Calle, Nombre, Número, Código postal, Provincia/Estado, País...

LocationListener

Es la clase que será notificada por Callbacks, cuando se de las condiciones establecidas en el seguimiento de la variación de la localización mediante el método

locationManager.requestLocationUpdates

LocationListener

requestLocationUpdates (String provider,
long minTime, //intervalo ms
float minDistance, //distancia m
LocationListener listener)

Este método debería usarse en onResume

LocationListener

removeUpdates (String provider)

Este método debería usarse cuando la actividad deja de estar visible (método onPause)

Alerta de Proximidad

addProximityAlert(latitud, longuitud, metros, tiempo, pendingIntent)

Es un método de LocationManager que me permite lanzar un componente (Avtivity, Servicio o Reciever) mediante PendingIntent cuando se dan las condiciones de distanciamento (metros) cada cierto tiempo (t)

Alerta de Proximidad

Además, no hace falta un Listener para mantener actualizada la posición; lo que en teoría, le convierte en un excelente método para generar avisos. Y sólo en teoría, porque en la práctica, no funciona con la precisión y exactitud esperadas (retardos, bloqueos, no dispara los avisos...)

Dibujar MAPA

La forma más sencilla de contar con un mapa al estilo GMaps en nuestra app, es incluir un layout con el siguiente fragment predefinido

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context=".MainActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

Dibujar MAPA

La forma más sencilla de contar con un mapa al estilo Google Maps en nuestra app, es incluir un layout donde se mencione un fragment predefinido. Basta asociar ese fragmente a una instancia de GoogleMap, para poder manejar los eventos del mapa dibujado

Dibujar MAPA

```
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context=".MainActivity"

android:name="com.google.android.gms.maps.SupportMapFrag
ment"/>
```

Google Play Services

Google provee librerías que pueden ser importadas a nuestro proyecto y ponen a nuestra disposición multitud de recursos de y servicios a modo de API's remotas

Google Play Services

La librería 'com.google.android.gms:play-services:8.4.0', incluye multitud de API's cuya lista completa puede cosultarse aquí:

https://developers.google.com/android/reference/packages

Google Play Services

Usando estas APIS, voy a realizar llamadas locales a mis librerías importadas, que a su vez, invocarán a los servidores de Google en muchos casos para obtener respuesta; por lo que se hace necesaria una autenticación.

Google Play Services

1. TENER CUENTA EN GOOGLE DEVELOPER
2. CREAR UN PROYECTO (o APP)
3. DECDIR QUÉ API necesitas usar
4. OBTENER UN CÓDIGO
5. INCLUIR ESE CÓDIGO EN TU Manifest

GoogleApiAviability

```
googleApiAvailability = GoogleApiAvailability.getInstance();
googleApiAvailability.isGooglePlayServicesAvailable(this);
```

Simplemente con una llamada, puedo comprobar si tengo conectividad con los servicios de google

GoogleApiClient

```
googleApiClient = new GoogleApiClient.Builder(this)
                    .addConnectionCallbacks(this)
                    .addOnConnectionFailedListener(this)
                    .addApi(LocationServices.API)
                    .build();
```

Además, debo instanciar el objeto GoogleApiClient que manejará las conexiones en background y al que deberé indicar mediante Scopes qué apis necesito usar (que servidores y servicios de Google) -esto último varía y no siempre es necesario para todoas las apis-

CLASE GEOCODER

Es una clase incluida en la API location que permite traducir unas coordenadas en una dirección y viceversa.

No debe emplearse en el hilo principal por tardar bastante en obtener la traducción

Hackeando Geoposición

SE PUEDE HACKEAR LA POSICIÓN DEL DISPOSITIVO (PENSADO PARA PRUEBAS) DEFINIENDO UN PROVEEDOR (LocationProvider) con el nombre “gps” y asociándole un Listener que se actulice usando parámetros de temporización del sistema (System.getCurrentTime())

Hackeando Geoposición

SE PUEDE HACKEAR LA POSICIÓN DEL DISPOSITIVO (PENSADO PARA PRUEBAS) DEFINIENDO UN PROVEEDOR (LocationProvider) CON EL NOMBRE DE GPS, ASOCIÁNDOLE UN LISTENER Y SETEANDO EL RELOJ DEL SISTEMA (System.getCurrentTime())

Hackeando Geoposición

Además se hace necesario duplicar en manifest en la subcarpeta src/debug e incluír en esa versión el permiso especial

<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION"/>

Esnifando Notificaciones

Se pueden tener acceso a todas las notificaciones que recibe el teléfono desarrollando un Servicio que herede NotificationListenerService y añadiendo en el Manifest la siguiente entrada

Esnifando Notificaciones

```
<service    android:name=".ServiNotis"
android:permission="android.permission.BIND_NOTIFICATION_
LISTENER_SERVICE">
            <intent-filter>
<action android:name=
"android.service.notification.NotificationListenerService" />
            </intent-filter>
        </service>
```

Esnifando Notificaciones

ADEMÁS DEBO DAR PERMISOS EN EL TELÉFONO A LA APP QUE DESARROLLO PARA RECOJER LOS AJUSTES → Sonidos y Notificaciones → Acceso a Notificaciones